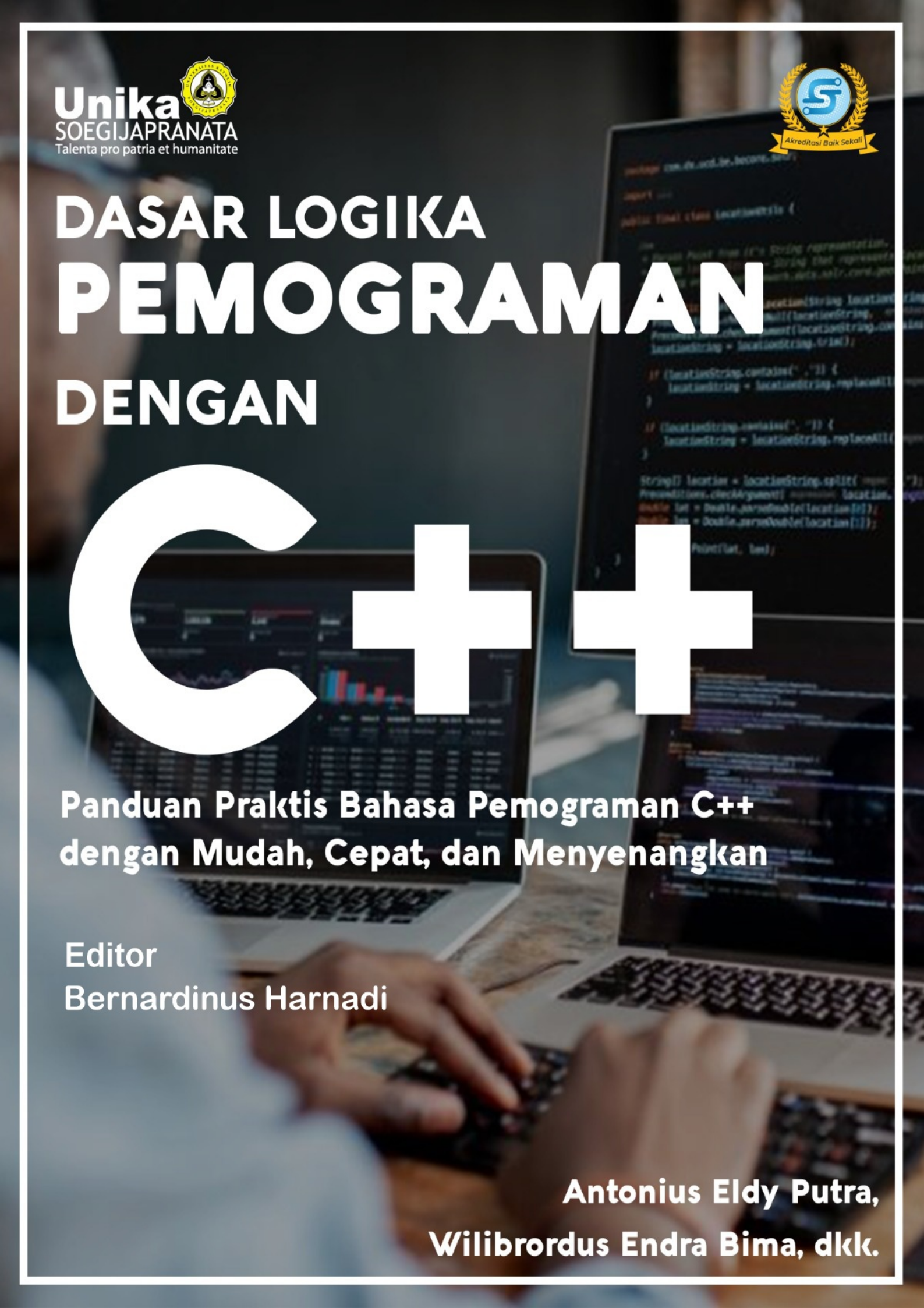


Unika SOEGIJAPRANATA
Talenta pro patria et humanitate

Akreditasi Baik Sekali

DASAR LOGIKA PEMOGRAMAN DENGAN C++

Panduan Praktis Bahasa Pemograman C++ dengan Mudah, Cepat, dan Menyenangkan

Editor
Bernardinus Harnadi

Antonius Eldy Putra,
Wilibrordus Endra Bima, dkk.

Dasar Pemrograman Logika Dengan C++

BERNARDINUS HARNADI,
ANTONIUS ELDY PUTRA,
WILIBRORDUS ENDRA BIMA, DKK

SIEGA PUBLISHER

SEMARANG

Dasar Logika Pemrograman Dengan C++

Bernardinus Harnadi, Antonius Eldy Putra, Wilibrordus Endra Bima, dkk

2 Januari 2025

Tim Penulis

1. Bernardinus Harnadi
2. Antonius Eldy Putra
3. Wilibrordus Endra Bima
4. Alfonso Praditya Galuh Mahesa
5. Ignatius Yogyawan Dwi
6. Mikha Eka Saputra
7. Tegar Heru Saputra
8. Irwan Wirawan Gulo

Table of Contents

Kata Pengantar

Kepada para pembaca yang terhormat, Dalam dunia yang semakin terhubung secara digital ini, pemrograman dan algoritma telah menjadi bahasa universal yang memungkinkan kita untuk berkomunikasi dengan mesin. Seiring dengan perkembangan teknologi, pemahaman dasar-dasar algoritma dan pemrograman telah menjadi keterampilan yang sangat berharga, bahkan menjadi keharusan, bagi siapa pun yang ingin bersaing di pasar kerja yang semakin kompetitif.

Dimulai dari fondasi yang kokoh, kami akan membimbing Anda melalui konsep-konsep dasar yang perlu dipahami setiap pemrogram, mulai dari struktur data hingga pengendalian aliran program. Bagi mereka yang baru mengenal dunia pemrograman, buku ini akan menjadi panduan yang sempurna untuk memahami konsep-konsep dasar. Sementara itu, bagi mereka yang telah memiliki pengalaman, buku ini akan menjadi sumber referensi yang berharga untuk memperdalam pemahaman mereka dan mengasah keterampilan mereka. Kami percaya bahwa pemrograman bukanlah sekadar keterampilan teknis, tetapi juga seni dalam memecahkan masalah dan mengungkapkan ide-ide kreatif.

Oleh karena itu, buku ini tidak hanya akan mengajarkan Anda “bagaimana” melakukan sesuatu, tetapi juga “mengapa” dan “kapan” harus melakukannya. Dalam

proses pembelajaran, kami juga menghargai pentingnya praktik dan latihan. Oleh karena itu, setiap konsep yang dijelaskan dalam buku ini akan disertai dengan contoh konkret dan latihan yang dirancang untuk menguji pemahaman Anda. Kami yakin bahwa dengan ketekunan dan dedikasi. Terakhir, kami ingin mengucapkan terima kasih kepada Anda, pembaca setia, karena telah memilih buku ini sebagai mitra Anda dalam perjalanan pembelajaran Anda. Semoga buku ini tidak hanya memberi Anda pengetahuan yang berharga, tetapi juga inspirasi untuk terus menjelajahi dunia yang luas dan menarik dari algoritma dan pemrograman.

Semarang,2 Januari 2025

Pendahuluan

Selamat datang dalam dunia pemrograman C++! Dalam buku ini, Anda akan diajak untuk menjelajahi dan memahami dasar-dasar serta konsep-konsep lanjutan dalam bahasa pemrograman C++. C++ telah menjadi salah satu bahasa pemrograman yang paling banyak digunakan di dunia, digunakan oleh para pengembang perangkat lunak, insinyur sistem, ilmuwan data, dan pengembang game untuk menciptakan aplikasi dan sistem yang kuat dan efisien.

Buku ini dirancang untuk memberikan pemahaman yang kokoh tentang sintaksis, struktur data, pemrograman berorientasi objek, dan teknik-teknik pemrograman lanjutan dalam C++. Kami akan memulai dengan memperkenalkan Anda pada dasar-dasar bahasa ini, termasuk tipe data, struktur kendali, dan fungsi dasar. Selanjutnya, Anda akan mempelajari konsep-konsep pemrograman berorientasi objek seperti kelas, pewarisan, polimorfisme, dan abstraksi.

Buku ini tidak hanya menawarkan penjelasan teoritis, tetapi juga berbagai contoh kode dan latihan praktis yang akan membantu Anda memperdalam pemahaman Anda dan meningkatkan keterampilan pemrograman C++ Anda. Selamat menikmati perjalanan Anda dalam mempelajari bahasa pemrograman C++ melalui buku ini!

1

Pengenalan C++

A. Pendahuluan

1. Pengertian dasar tentang bahasa pemograman c++

Pada saat ini sudah banyak pemograman yang di jadikan keterampilan yang tujuannya untuk mencari pekerjaan dan mendapat penghasilan. Nah, pada bab kali ini kita akan membahas tentang pemograman bahasa c++. Sebelum kita masuk lebih dalam tentang c++, sebelumnya kita harus mengetahui apa itu bahasa pemograman C++ ?. C++ adalah salah satu bahasa pemograman tingkat tinggi yang pada umumnya banyak digunakan untuk pengembangan perangkat lunak. C++ juga merupakan salah satu pemograman yang sangat populer karna dapat mengembangkan berbagai jenis aplikasi web. Aplikasi web yang dapat di kembangkan oleh pemograman c++ yaitu perangkat lunak destop hingga permainan, berbagai aplikasi web, dan sitem yang tertanam.

2. Asal Mula Dan Sejarah C++

Pada awalnya sebelum muculnya pemograman bahasa c++ ada bahasa pemograman C, yang dimana pemograman C ini diciptakan oleh tokoh yang bernama Dennis Ritchie yang lahir di Bronxville, New york, Amerikat serikat pada tanggal 9 September 1941. Dia dibesarkan di daerah Bronxville dan dikenal sebagai ilmuan komputer dan salah satu tokoh penting dalam

sejarah teknologi informasi karena kontribusi besar terhadap pemograman C. Bahasa pemograman C secara luas digunakan karna flektabilitasnya, portabilitas, dan sangat efesien dalam memanipulasi sumber daya sistem.

Pada tahun 1980-an ,Bjarne Stroustrup seorang ilmuan komputer dari Bell laps, mulai merangcang bahasa baru berbasis C,(C with classes). Tujuan utamanya merangcang pemograman berbasis C untuk memperluas kemampuan bahasa C dengan memasukan konsep-konsep pemograman beriontasi objek. Bjarne Stroustrup menggabungkan fitur baru seperti kelas, abstraksi, dan polimofirme yang tujuanya agar penggunaan bahasa C lebih sederhana. Asal mula c++ berawl dari kebutuhan untuk mengatasi beberapa kekurangan pemograman bahasa C dalam pengembangan perangkat lunak yang komplek dan besar. C++ pertama kali di perkenalkan kepada publik pada tahun 1983 yang di perkenalkan melalui sebuah artikel yang ditulis oleh Stroustrup. Pada 1985 Stroustrup menerbitkan buku yang berjudul " The C++ Programming Language". Sejak saat itu, C++ terus berkembang dan menjadi salah satu bahasa pemograman yang sangat populer dan sangat berpengaruh di dunia. C++ juga menjadi dasar bagi banyak bahasa pemograman dan fitur-fitur inovatifnya terus memengaruhi perkembangan teknologi perangkat lunak secara keseluruhan.

B.Kelebihan Dan Kekurangan C++

Nah setelah mengetahui pengertian dan sejarah

c++, kali ini kita akan membahas tentang kelebihannya serta kekurangan yang ada pada bahasa pemograman C++.

1. Kelebihan C++

Ada beberapa fitur menarik yang terdapat pada bahasa pemograman c++ ini diantaranya sebagai berikut :

1. Kinerja Tinggi: C++ adalah bahasa yang dikompilasi, yang berarti bahwa program-program yang ditulis dalam bahasa ini dapat berjalan dengan kinerja yang sangat tinggi. Bahasa ini memberikan akses langsung ke perangkat keras komputer, memungkinkan pengembang untuk mengoptimalkan performa aplikasi mereka secara maksimal.
2. Kontrol Langsung terhadap Hardware: C++ memberikan kontrol langsung terhadap perangkat keras komputer melalui fitur-fitur seperti pointer dan referensi. Ini memungkinkan pengembang untuk mengakses dan memanipulasi memori secara langsung, yang merupakan keuntungan besar dalam pengembangan perangkat lunak yang membutuhkan kinerja tinggi.
3. Bahasa Multi-paradigma: C++ adalah bahasa pemrograman multi-paradigma yang mendukung paradigma pemrograman prosedural, berorientasi objek, dan generik. Ini memberikan fleksibilitas yang besar kepada pengembang untuk memilih pendekatan pemrograman yang paling sesuai dengan kebutuhan proyek mereka.

4. Portabilitas: C++ telah diimplementasikan di banyak platform yang berbeda, sehingga dapat dijalankan di berbagai sistem operasi dan arsitektur perangkat keras. Ini memberikan portabilitas yang baik untuk aplikasi yang ditulis dalam bahasa C++.

5. Perpustakaan dan Ekosistem yang Kuat: C++ memiliki ekosistem yang sangat luas dan beragam perpustakaan yang dapat digunakan oleh pengembang untuk mempercepat proses pengembangan perangkat lunak. Ini termasuk perpustakaan standar seperti STL (Standard Template Library) dan perpustakaan pihak ketiga seperti Boost, yang menyediakan berbagai macam fungsionalitas siap pakai.Kesesuaian untuk Pengembangan Permainan: Karena kinerja tinggi dan kontrol langsung terhadap hardware, C++ menjadi salah satu pilihan utama untuk pengembangan permainan video. Bahasa ini memungkinkan pengembang untuk mengoptimalkan permainan mereka secara maksimal dan mengakses sumber daya perangkat keras dengan efisien.

6. Fleksibilitas dan Kontrol: C++ memberikan tingkat fleksibilitas dan kontrol yang tinggi kepada pengembang, terutama dalam hal manajemen memori dan penggunaan pointer. Ini memungkinkan pengembang untuk mengoptimalkan kinerja aplikasi mereka dan mengimplementasikan solusi khusus yang sesuai dengan kebutuhan proyek mereka.

7. Kompatibilitas dengan Bahasa C: C++ kompatibel dengan bahasa C, yang berarti bahwa kode C dapat diintegrasikan secara langsung ke dalam kode C++ dan sebaliknya. Ini membuat C++ menjadi pilihan yang baik

untuk pengembangan perangkat lunak yang memerlukan integrasi dengan kode C yang sudah ada.

8. Dukungan untuk Pemrograman Sistem: C++ banyak digunakan dalam pengembangan sistem operasi dan perangkat lunak sistem lainnya karena kemampuannya untuk berinteraksi langsung dengan perangkat keras dan mengakses fitur-fitur sistem secara langsung.

9. Kebutuhan Sumber Daya Rendah: C++ memiliki kebutuhan sumber daya yang relatif rendah dalam hal memori dan CPU, sehingga cocok untuk pengembangan perangkat lunak yang membutuhkan efisiensi sumber daya yang tinggi, seperti aplikasi embedded dan sistem real-time.

10. Kesesuaian untuk Pengembangan Permainan: Karena kinerja tinggi dan kontrol langsung terhadap hardware, C++ menjadi salah satu pilihan utama untuk pengembangan permainan video. Bahasa ini memungkinkan pengembang untuk mengoptimalkan permainan mereka secara maksimal dan mengakses sumber daya perangkat keras dengan efisien.

2. Kekurangan Yang Terdapat Pada Pemograman C++

Setelah mengetahui kelebihan dan fitur menarik yang terdapat pada C++ tentu saja suatu aplikasi memmiliki kekurangan didalamnya. Adapun fitur yang menjadi kelemahan suatu bahasa pemograman C++ antara lain sebagai berikut :

1. Pemrosesan String yang Rumit: Bahasa pemrograman C++ memiliki dukungan string yang kurang kuat dibandingkan dengan bahasa modern

lainnya. Ini dapat menyulitkan pemrosesan string, manipulasi teks, dan operasi terkait.

2. Keterbatasan dalam Manajemen Memori: Meskipun memberikan fleksibilitas dalam manajemen memori, pengelolaan manual memori dalam C++ dapat menyebabkan kesalahan, kebocoran memori, dan bahkan kerentanan keamanan.

3. Kompleksitas Template: Penggunaan template dalam C++, meskipun kuat, sering kali kompleks dan sulit dipahami, terutama untuk pemula.

4. Kompatibilitas Versi: Pemutakhiran standar C++ kadang-kadang menyebabkan masalah kompatibilitas dengan kode yang ada, terutama ketika menggunakan fitur eksperimental atau spesifik versi.

5. Kode Boilerplate yang Tinggi: Penggunaan C++ sering kali memerlukan penulisan kode boilerplate yang besar, yang dapat meningkatkan kompleksitas proyek dan mengurangi produktivitas.

6. Kurangnya Standarisasi Implementasi: Meskipun ada standar C++ yang ditetapkan oleh ISO, implementasi dari kompilator dan perpustakaan mungkin bervariasi antara platform dan penyedia, menyebabkan masalah portabilitas.

7. Keterbatasan Dukungan untuk Konkurensi dan Paralelisme: Meskipun ada beberapa fitur untuk konkurensi dan paralelisme dalam C++, namun penggunaannya cenderung lebih rumit dan kurang ekspresif dibandingkan dengan bahasa pemrograman lainnya.

8. Penggunaan Pointers yang Berpotensi Berbahaya:

Penggunaan pointer dalam C++ dapat menyebabkan kesalahan memori yang berpotensi berbahaya, terutama ketika tidak dikelola dengan benar.

9. Siklus Pembelajaran yang Curam: Kurva pembelajaran C++ cenderung curam, terutama bagi pemula, karena kompleksitas bahasa dan banyaknya fitur yang perlu dipelajari.

10. Kurangnya Dukungan untuk Pembangunan Modern: C++ tidak selalu memiliki dukungan yang kuat untuk paradigma pengembangan modern seperti pemrograman fungsional atau reaktif.

Setelah mengetahui kelebihan dan kekurangan dari bahasa pemograman C++ tentu dalam menjalankan pemograman kita tahu kesulitan apa saja yang dapat diperhatikan jika ingin menguasai bahasa pemograman C++. Jadi dengan mengetahui kelebihan pemograman bahasa C++ ini, kita dapat memanfaatkan untuk dijadikan motivasi jika ingin menjaadi pemograman yang baik.

C. Teks Editor Dan Cara Menjalankan Bahasa Pemograman C++

1. Teks Editor

Tekst editor adalah program komputer yang digunakan untuk mengedit teks, baik itu dalam bentuk dokumen, kode program, atau berkas teks lainnya. Perangkat lunak ini umumnya memiliki antarmuka sederhana yang memungkinkan pengguna untuk menulis, mengedit, dan menyimpan teks.

Teks editor sering digunakan oleh programmer untuk menulis kode program dalam berbagai bahasa pemrograman seperti Python, Java, C++, dan sebagainya. Namun, tekst editor juga sering digunakan dalam pekerjaan sehari-hari untuk mengedit dokumen teks biasa seperti dokumen Word, catatan, atau berkas konfigurasi. Beberapa contoh tekst editor populer termasuk Notepad (di Windows), TextEdit (di macOS), Vim, Emacs, Sublime Text, Visual Studio Code, dan Atom. Setiap tekst editor memiliki fitur dan fungsionalitas yang berbeda, dan beberapa di antaranya mendukung fitur lanjutan seperti penyorotan sintaksis, pemeriksaan ejaan, dan dukungan untuk berbagai bahasa pemrograman.

Text editor digunakan dalam pengembangan perangkat lunak dengan bahasa pemrograman C++ untuk mengedit kode program. Berikut adalah beberapa fungsi dan kegunaan text editor dalam konteks pengembangan C++:

a. Menulis Kode: Text editor memungkinkan programmer untuk menulis, mengedit, dan mengatur kode program C++. Ini adalah tempat di mana programmer menghasilkan logika program mereka, menambahkan sintaks, dan mengatur struktur program.

b. Penyorotan Sintaksis: Text editor sering menyediakan fitur penyorotan sintaksis yang membedakan berbagai elemen dalam kode program, seperti variabel, fungsi, komentar, dan kata kunci bahasa. Ini membantu programmer untuk dengan mudah membaca dan memahami kode yang mereka tulis.

c. Pemeriksaan Ejaan: Beberapa text editor memiliki fitur pemeriksaan ejaan yang membantu dalam mengidentifikasi kesalahan pengejaan dalam komentar, pesan, atau bahkan dalam kode program itu sendiri.

d. Integrasi dengan Compiler: Beberapa text editor dapat diintegrasikan dengan compiler C++ sehingga programmer dapat dengan cepat mengompilasi dan menjalankan program dari dalam text editor.

e. Snippet dan Template: Text editor sering menyediakan fitur untuk menyimpan dan menggunakan potongan kode atau template yang sering digunakan. Ini membantu dalam meningkatkan produktivitas dengan memungkinkan pengguna untuk dengan cepat menyisipkan kode yang sering digunakan.

f. Integrasi dengan Sistem Kontrol Versi: Banyak text editor dapat diintegrasikan dengan sistem kontrol versi seperti Git, SVN, atau Mercurial. Ini memudahkan pengelolaan kode sumber dalam proyek pengembangan yang lebih besar.

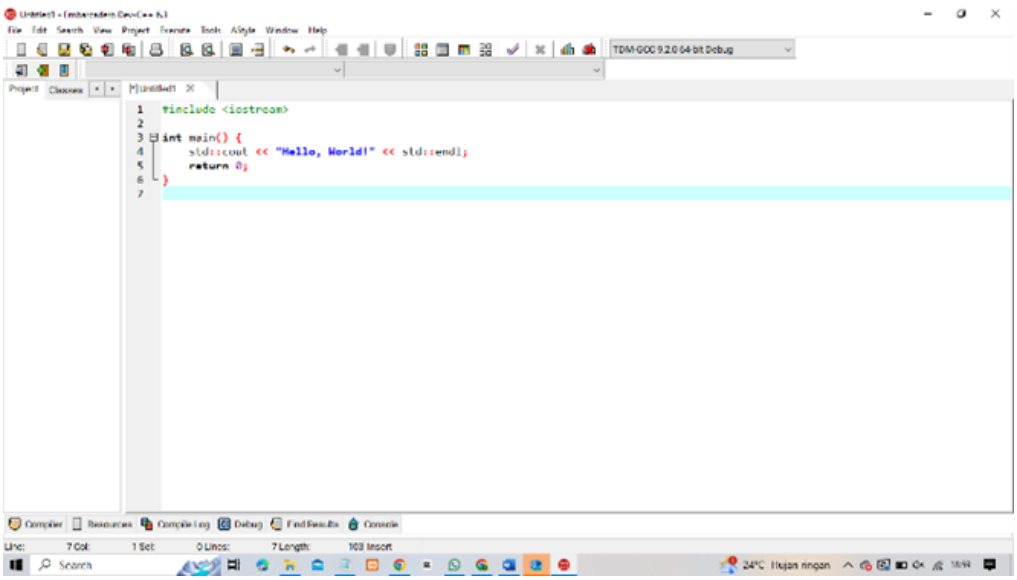
g. Ekstensibilitas: Beberapa text editor bersifat ekstensibel, yang berarti pengguna dapat menginstal plugin atau ekstensi tambahan untuk menambahkan fungsionalitas tambahan sesuai kebutuhan mereka, seperti debugging, integrasi dengan alat pengujian, atau peningkatan produktivitas lainnya.

h. Portabilitas: Text editor sering tersedia di berbagai platform, sehingga memungkinkan programmer untuk konsisten dalam pengalaman pengkodean mereka terlepas dari sistem operasi yang mereka gunakan.

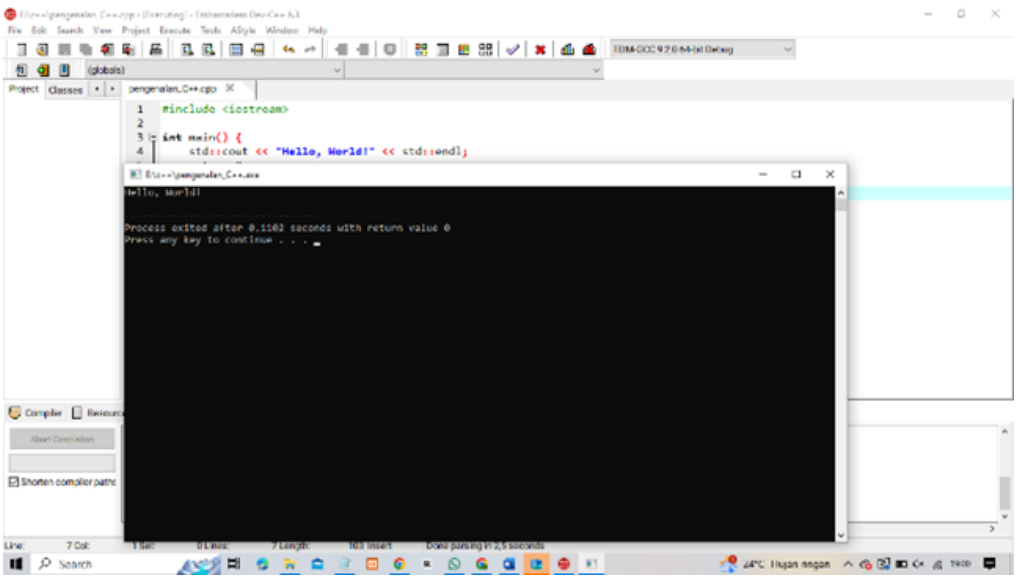
2. cara menjalankan C++

Cara menjalankan pemograman bahasa C++ dilakukan dengan beberapa tahap dimulai dari penginstalan aplikasi, tulis kode program, simpan kode program, kompilasi program(jalankan program), dan tampilkan program yang sudah dibuat.

Contoh cara menjalankan pemograman C++



Gambar 1.1 cara menulisan helloworld



Gambar 1.2 hasil dari penulisan helloworld

Gambar di atas adalah contoh menjalankan program C++ dengan memasukan kode program dan merupakkan hasil dari menjalankan pemograman C++.

D. Struktur kode program C++

Kode Program adalah rangkaian pernyataan atau deklarasi yang ditulis dalam bahasa pemograman yang ditulis dalam bahasa pemograman yang terbaca manusia.

Kode program biasanya sebagai kompilasi dan eksekusi program.

Preposesor = Lybrary yang digunakan

Kompiler = Menerjemahkan kedalam bahasa assemble

Assembler = menerima keluaran dari kopiler dan akan membuat sebuah kode objek

Link editor = mengkombinasikan kode objek dan

lybrary yang lain menjadi file exe

cout << “Hello, world!” << endl;: Baris ini mencetak string “Hello, world!” ke layar menggunakan objek cout dari namespace std. endl digunakan untuk menambahkan newline (baris baru) setelah mencetak.Berikut ini adalah struktur dasar kode program c++:

1. #include <iostream>: Direktif praprosesor yang memasukkan file header iostream yang berisi definisi objek cin (untuk input) dan cout (untuk output) dari C++ Standar Library. File include ini juga sering disebut sebagai header file.

2. using namespace std; Menggunakan namespace std untuk menghindari penulisan std:: sebelum cout dan cin.

3. Struktur main() pada dasarnya merupakan sebuah fungsi (function). Isi dari function ini diawali dan diakhiri dengan tanda kurung kurawal ” { ” dan ” } “. Di dalam tanda kurung inilah “isi” dari kode program penyusun function main() ditulis. Function main() merupakan kode program utama dalam mayoritas aplikasi bahasa C++. Di sinilah kita akan banyak menulis kode program.

4. cout << “Hello, world!” << endl;: Baris ini mencetak string “Hello, world!” ke layar menggunakan objek cout dari namespace std. endl digunakan untuk menambahkan newline (baris baru) setelah mencetak.

5. Perintah return 0; berhubungan dengan kode int main() sebelumnya. Disinilah kita menutup function

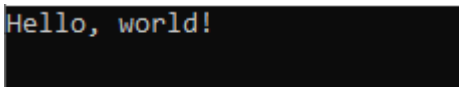
main() yang sekaligus mengakhiri kode program bahasa C++. Return 0 : Mengembalikan nilai 0 ke sistem operasi sebagai tanda bahwa program telah berjalan dengan sukses. Nilai ini akan ditangkap oleh sistem operasi.

Dengan mengetahui struktur kode program maka kita dapatmelakukanpemograman sederhana c++ dengan menggunakan seluruh struktur kode program.Contoh penggunaan kode program:

```
#include <iostream> // Direktif untuk masukan dan keluaran standar
using namespace std; // Menggunakan namespace std untuk menghindari penulisan std:: sebelum objek dari namespace std

// Fungsi utama
int main() {
    // Penulisan kode program di sini

    // Menampilkan output ke layar
    cout << “Hello, world!” << endl;

    // Mengembalikan nilai 0 yang menandakan program berjalan dengan sukses
```

```
    return 0;
}
```

Hasilnya:



Gambar 1.3 tampilan hasil penulisan helloworld

a. Hello World

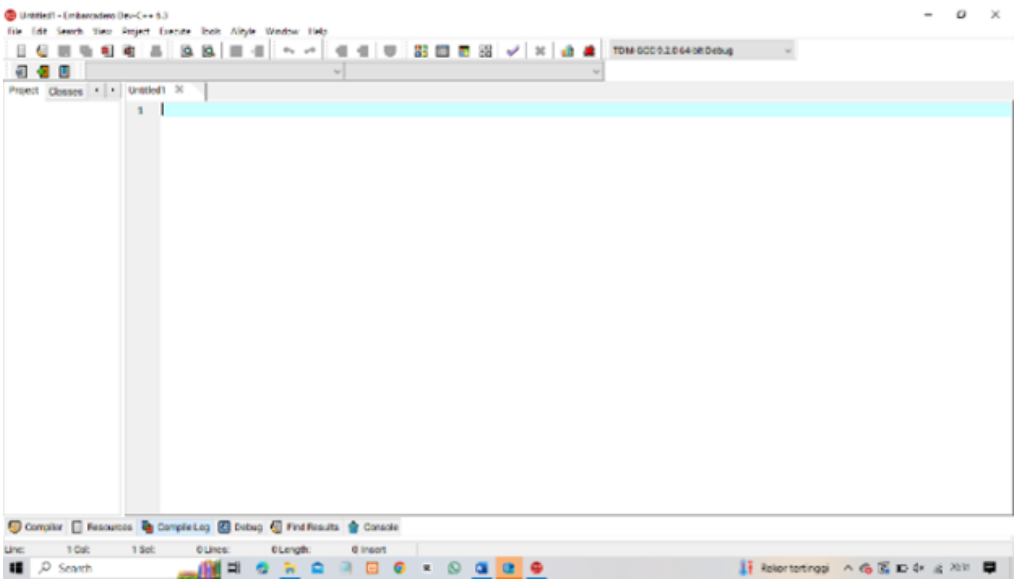
Hello world merupakan program yang pertama kali di ajarkan saat belajar pemograman sederhana c++. Pada umumnya program hello ini di merupakan program komputer yang menampilkan teks “Hello World” ke layar. Biasanya program Hello World merupakan pemograman dasar sebelum membuat pemograman yang lebih sulit

Membuat Program Hello World menggunakan C++

- Pastikan sudah mendowload aplikasi Dev-C++, setelah mendowload buka aplikasi C++.

Gambar 1.4 logo aplikasi Dev C++

- Setelah membuka program c++ kemudian untuk memulai tekan (Ctrl+N) untuk memulai membuat program. Maka tampilan akan seperti gambar di bawah ini.

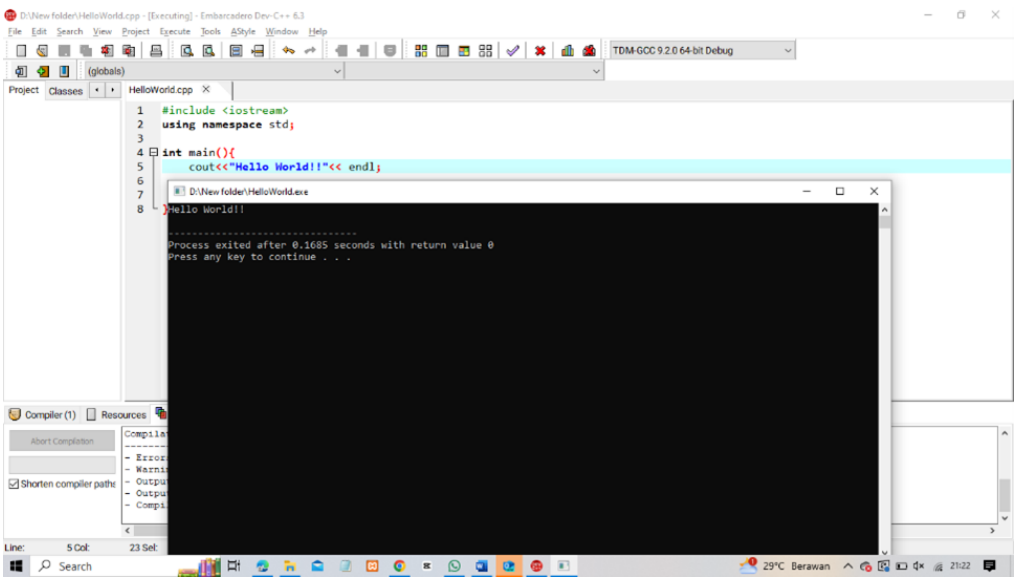


Gambar 1.5 tampilan awal aplikasi Dev C++

- Mulai dengan mengetikan kode program seperti

berikut.

```
#Include<iostream>
using namespace std;

int main(){
    cout<<"Hello world!!"<<endl;

    return 0;
}
```



Gambar 1.6 hasil menjalankan program

Menganalisis program:

- include <iostream> Baris ini memasukkan

pustaka input-output standar C++ yang memungkinkan kita menggunakan cout dan endl.

- Using namespace std adalah fitur penamaan dalam Bahasa pemrograman yang menerapkan konsep OOP (Object Oriented Programming). Tujuan : berbagai perintah tidak saling bentrok
- int main() { ... }adalah fungsi utama dari program C++. Setiap program C++ harus memiliki fungsi main().
- cout << “Hello, World!” <<endl Ini adalah pernyataan yang mencetak teks “Hello, World!” ke layar.
- return 0, Mengembalikan nilai 0 kepada sistem operasi, menandakan bahwa program selesai dijalankan tanpa masalah.

Hal-hal yang perlu diingat:

1. Kita menggunakan cout untuk mencetak output di layar.
2. Harus menyertakan #include <iostream> agar kita dapat menggunakan cout.
3. Eksekusi kode dimulai dari fungsi main(), yang wajib ada dalam program C++.

2. komentar

Komentar dalam c++ biasanya digunakan oleh para programmer untuk memberi tanda atau keterangan pada kode program yang dia kembangkan, agar mudah dibaca, dipahami dan dimengerti terutama untuk programmer lainya jika kita sedang bekerja dalam suatu

team, karena tidak semua programmer mempunyai jalan pikiran yang sama dalam pembuatan algoritma, hal ini sangat beguna untuk memberitahu dengan menggunakan bahasa manusia dan mempermudahkan pada orang untuk mempelajari apa yang ada di dalam kode program yang telah kita tulis.

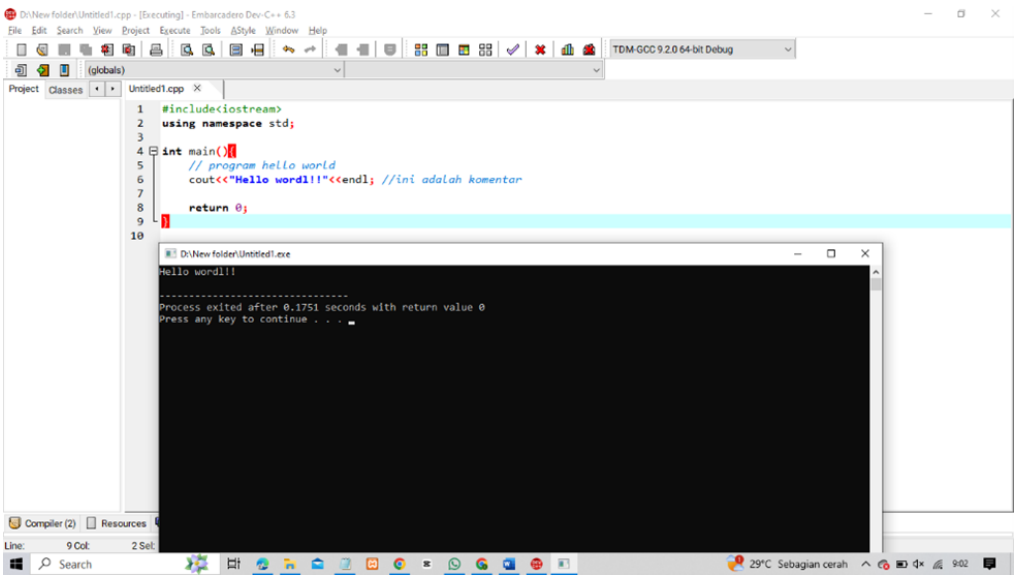
Dalam program C++ terdapat dua jenis komentar yang digunakan dalam melakukan pemograman yaitu (sigle-line dan multi-line).

a. Single-line(menggunakan tanda //)

Komentar ini sering digunakan sebagai penjelas satu baris. Berikut contoh penggunaan komentar sigle-line. (// ini adalah komentar).

Cara menggunakan komentar di bahasa pemograman

```
#include<iostream>
using namespace std;

int main(){
    // program hello world
    cout<<“Hello wordl!!”<<endl; //ini adalah komentar

    return 0;
}
```



Gambar 1.7 hasil menjalankan program

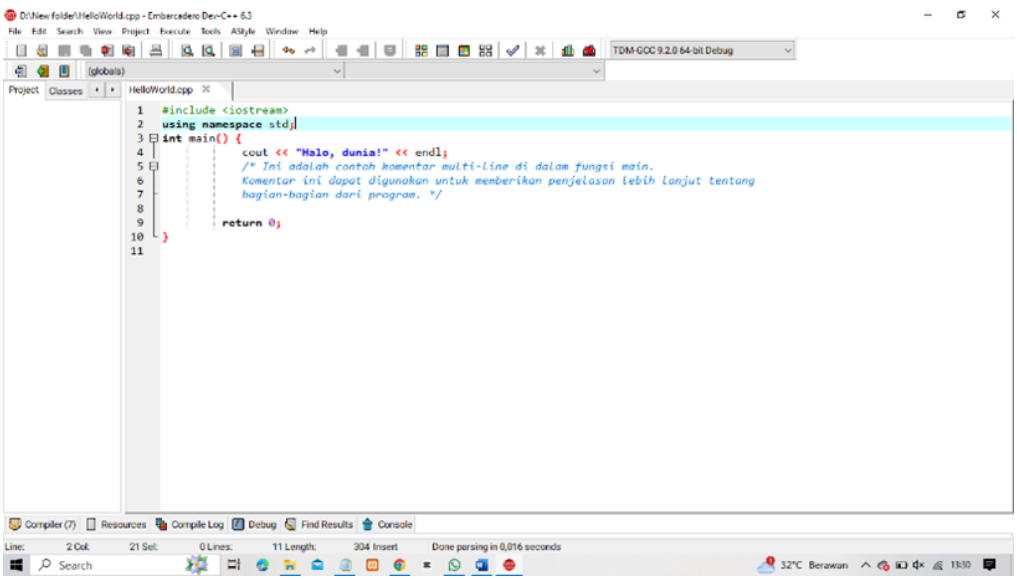
Komentar satu baris dimulai dengan dua garis miring (//). Semua teks setelah tanda // hingga akhir baris dianggap sebagai komentar dan tidak diproses oleh kompiler. Komentar ini juga bisa digunakan dibagian samping kode program untuk memperjelas kepada pembaca dan membantu suatu pemograman mengingat kode yang dibuat.

b. Multi-Line(menggunakan tanda /* dan */)

Komentar Multi-Line merupakan komentar multi baris yang diawali dengan tanda /* dan di akhiri dengan tanda */. Suatu kalimat yang berada di antara /* dan */ di anggap sebagai komentar.

Contoh penggunaan komentar multi line.

```
#include <iostream>
        using namespace std;
int main() {
        cout << “Halo, dunia!” << endl;
/* Ini adalah contoh komentar multi-line di dalam fungsi main.
Komentar ini dapat digunakan untuk memberikan penjelasan lebih lanjut tentang     bagian-bagian dari program. */
        return 0;
}
```



Gambar 1.8 penulisan komentar di Dev C++

E. Mengenal Variabel Dan Tipe Data Dalam C++

a.Pengertian Variabel

Variabel adalah penanda identitas yang digunakan untuk menampung suatu nilai yang sudah di ketahuai atau belum diketahui. Variabel merupakan lokasi memori yang diberi nama untuk menyimpan data yang dapat diubah selama eksekusi program. Setiap variabel memiliki tipe data tertentu yang menentukan jenis nilai yang dapat disimpan di dalamnya, seperti bilangan bulat, pecahan, karakter, atau nilai kebenaran. Jika dilihat dari nilai tipe data yang dimasukkan kedalam sebuah variabel, variabel bisa dibedakan menjadi dua yaitu:

1) Variabel Numerik

Variabel numerik ini selanjutnya dapat dibagi menjadi menjadi 3 macam, yaitu:

- Bilangan Bulat atau Integer
- Bilangan Desimal Berpresisi Tunggal / Floating Point.
- Bilangan Desimal Berpresisi Ganda / Double Precision

2) Variabel Text

Variabel teks ini selanjutnya dapat dibagi menjadi menjadi 2 macam, yaitu:

- Character (Karakter Tunggal)
- String (Untuk Rangkaian Karakter)

Berikut aturan penamaan sebuah variabel.

a) Nama WAJIB dimulai dengan huruf atau underscore. Dengan kata lain, sebuah nama variabel tidak dapat dimulai dengan angka. Contoh: nama, umur, _nim dan sebagainya.

b) Nama tidak boleh mengandung simbol-simbol khusus, seperti !, @, #, $, %, ^, &, *, (,), +, -, /, ~, =, [,], |, \, dan sebagainya kecuali underscore (‘_’).

c) Nama yang lebih dari satu kata tidak boleh dipisahkan dengan spasi. Gunakan underscore untuk memisahkan kata-kata tersebut (contoh: nilai_huruf, data_pegawai, nomor_induk_pegawai, nomor_telepon, rata_rata) atau gunakan huruf besar untuk setiap huruf pertama dari kata-kata tersebut (contoh: nilaiAngka, namaMahasiswa, nomorIndukMahasiswa).

d) Maksimum panjang nama sebuah variabel adalah 31 karakter. Nama yang terlalu panjang sebaiknya disingkat. Contoh nim, nip dan sebagainya.

e) Kata-kata kunci atau keywords yang disediakan bahasa C, seperti switch, if, else, int, char dan lain sebagainya, dan nama-nama fungsi TIDAK DAPAT digunakan sebagai nama variabel, kecuali kata-kata tersebut digabungkan dengan karakter lain sehingga membentuk kata-kata baru.

f) Bahasa C++ merupakan bahasa yang case sensitive, artinya huruf besar dan huruf kecil akan diperlakukan berbeda. Contoh namamahasiswa dan namaMahasiswa akan diperlakukan berbeda.

1. Tipe data variabel

Variabel dapat dikelompokkan menjadi berbagai jenis berdasarkan sifatnya, seperti variabel kategorikal (atau variabel kualitatif) dan variabel numerik (atau variabel kuantitatif). Variabel kategorikal dapat terdiri dari kategori atau label, sedangkan variabel numerik terdiri dari nilai numerik.

Tipe variabel adalah cara untuk menentukan jenis data yang akan disimpan dalam jenis variabael. Berikut adalah beberapa tipe variabel yang biasanya digunakan dalam pemograman C++

1) Integer(int): digunakan untuk menyimpan bilangan bulat, misalnya 1,2,10 dan sebagainya. Contoh penggunaan:

```
contoh;

int umur = 25;
```

Gambar 1.9 penulisan tipe data integer

2) Floating Point (float dan double): Digunakan untuk menyimpan bilangan pecahan. Tipe data float menyimpan bilangan pecahan dengan presisi yang lebih rendah dibandingkan dengan double. Contoh penggunaan:

```
contoh;

float tinggi = 175.5;
double harga = 99.99;
```

Gambar 1.10 penulisan tipe data float dan double

3) Character (char): Digunakan untuk menyimpan karakter tunggal, seperti huruf, angka, atau simbol. Contoh penggunaan:

```
contoh;

char jenisKelamin = 'L';
```

Gambar 1.11 penulisan tipe data char

4) Boolean (bool): Digunakan untuk menyimpan nilai kebenaran (true atau false). Contoh penggunaan:

```
contoh;

bool flag = true;
```

Gambar 1.12 penulisan tipe data boolean

5) String: Digunakan untuk menyimpan kumpulan

karakter atau teks. Di C++, tipe data string didefinisikan sebagai bagian dari library string. Contoh penggunaan:

```
#include <string>
using namespace std;

string nama = "Iwg Gul";
```

Gambar 1.13 penulisan tipe data string

6) Array: Digunakan untuk menyimpan sejumlah nilai yang serupa dalam satu variabel. Contoh penggunaan:

```
#include <iostream>
using namespace std;

struct Point {
    int x;
    int y;
};

int main() {
    // Inisialisasi struktur Point
    Point p1 = {10, 20};

    // Menampilkan nilai x dan y
    cout << "Point p1: (" << p1.x << ", " << p1.y << ")" <<endl;

    return 0;
}
```

Gambar 1.14 penulisan tipe data array

2. Deklarasi variabel

Deklarasi variabel adalah proses pengalokasian

memori untuk variabel dan menentukan tipe data yang akan disimpan di dalamnya.

Cara umum deklarasi variabel:

```
contoh:

tipe_data nama_variabel1, nama_variabel2, nama_variabel3;
```

Gambar 1.15 penulisan deklarasi variabel

```
contoh;

int BilanganBulat;
```

Gambar 1.16 penulisan deklarasi variabel

Contoh ini merupakan deklarasi variabel berupa int. Variabel di atas sudah dapat digunakan untuk menampung nilai-nilai berupa bilangan bulat (sesuai rentang yang ada pada tipe data int).

Apabila mendeklarasikan variabel bertipe sama maka dapat menyingkat penulisannya dengan cara seperti berikut.

```
contoh;

int nilai1, nilai2, nilai3;
```

Gambar 1.17 penulisan deklarasi variabel tipe data integer

Penulisan di atas adalah contoh mendeklarasikan variabel bertipe int yaitu nilai1, nilai2, dan nilai3.

Ketentuan dalam membuat variabel. Berikut ketentuan dalam membuat variabel:

- Nama vaariabel tidak boleh diawali dengan simbol, angka. Nama variabel harus diawali dengan huruf misalnya.

```
contoh;

int kuliah1;     //penulisan yang benar
int 1kuliah      //penulisan yang salah
```

Gambar 1.18 penulisan nama variabel yang benar dan salah

- Variabel tidak boleh menggunakan spasi, tetapi dapat mengganti spasi dengan menggunakan underscore “–“. Misalnya.

```
contoh;

int nama_mahasiswa;      //penulisan yang benar
int nama mahasiswa;      //penulisan yang salah
```

Gambar 1.19 penulisan nama variabel yang benar dan salah

- Jangan menggunakan kata kunci yang sudah ada dalam bahasa C (seperti if, int, void, dll.).

```
contoh;

int break;       //penulisan yang salah
int return;      //penulisan yang salah
int case;        //penulisan yang salah
```

Gambar 1.20 penulisan nama variabel yang benar dan salah

- Case sensitive. Yang artinya huruf besar dan kecil dibedakan. Misalnya variabel bilanganBulat akan berbeda dengan variabel bilanganbulat

3. Inisialisasi Variabel Dalam c++

Inisialisasi variabel dalah tugas pemberian nilai awal yang dilakukan saat deklarasi variabel atau obyek.

Inisialisasi sangat disarankan untuk dilakukan setiap mendirikan sebuah deklarasi.

Inisialisasi disarankan untuk dilakukan karena ketika suatu program melepaskan memori, sebenarnya memori tersebut hanya dilepaskan dan tidak dikosongkan, dengan arti ketika kita memesan memori (mendirikan variabel) tanpa inisialisasi, kemungkinan kita akan mendapatkan data sampah pada memori yang telah kita pesan itu.

Contoh.

```
int umur = 25; // inisialisasi variabel umur dengan nilai 25
float pi = 3.14; // inisialisasi variabel pi dengan nilai 3.14
char grade = 'A'; // inisialisasi variabel grade dengan karakter 'A'
```

Gambar 1.21 penulisan inisialisasi variabel

Ada beberapa jenis inisialisasi variabel antara lain sebagai berikut.

- Inisialisasi satu baris

Inisialisasi variabel satu baris merupakan variabel yang di inisialisasikan saat deklarasi menggunakan tanda sama dengan “=”.

Contoh.

```
int x = 10;
float y = 5.5;
```

Gambar 1.22 penulisan inisialisasi satu baris

- Inisialisasi multi baris

Inisialisasi multi baris merupakan variabel yang diinisialisasikan dengan menggunakan beberapa baris kode.

Contoh.

```
int a = 1,
    b = 2,
    c = 3;
```

Gambar 1.23 penulisan inisialisasi multi baris

- Inisialisasi kurung kurawal

Inisialisasi kurung kurawal biasanya digunakan untuk array dan struktur. Contoh.

```
#include <iostream>
using namespace std;

struct Point {
    int x;
    int y;
};

int main() {
    // Inisialisasi struktur Point
    Point p1 = {10, 20};

    // Menampilkan nilai x dan y
    cout << "Point p1: (" << p1.x << ", " << p1.y << ")" <<endl;

    return 0;
}
```

Gambar 1.24 penulisan inisialisasi kurung kurawal

- Inisialisasi tanpa nama variabel

Variabel tanpa nama dapat diinisialisasi menggunakan inisialisasi tanpa nama. Contoh:

```
int a = {}; // a diinisialisasi dengan nilai default 0
```

Gambar 1.25 penulisan inisialisasi tanpa nama variabel

F. Input Dan Output Dalam Bahasa Pemograman Bahasa C++

1.mengenal input dan output pemograman c++

Input merupakan suatu data yang kita masukan kedalam program komputer. Input sangat penting dalam sebuah pemograman karena memungkinkan program menjalankan informasi yang diperlukan untuk menjalankan tugas tertentu.

Proses adalah langkah-langkah program untuk menghasilakan output. Output merupakan infomasi yang dihasilkan setelah dilakukan proses. Fungsi output untuk menampilkan program yang dibuat ke layar komputer

a. Menampilkan teks ke layar menggunakan ‘cout’

Cout adalah objek dari kelas ostream yang digunakan untuk menampilkan output ke layar. Dalam pemograman bahasa sederhana C++ kita dapat menggunakan cout untuk menampilkan teks atau nilai ke layar.

Contoh penggunaan cout.

```
cout<<"Hello World"<<endl;
cout<<"hello C++\n";
```

Gambar 1.26 penulisan output pada C++

Fungsi cout untuk menampilkan output ke layar. Setelah tanda “<<” kita dapat menuliskan teks yang ingin di tampilkan ke layar dan teks yang ingin di tampilkan ke layar harus di apit oleh tanda petik. Pada contoh penggunaan cout di atas terdapat endl dan \n yang fungsinya untuk membuat baris baru. Hasil output diatas yaitu:

Hello World

Hello C++

Jika pada program di atas tidak menuliskan endl atau \n maka hasil yang ditampilkan ke layar adalah seperti berikut.

Hello World Hello C++.

Jadi jika ingin menampilkan teks tiap baris kita harus menggunakan simbol endl atau \n.

Setiap simbol “<<” maka akan di tampilkan ke layar output contoh sebagai berikut.

```
#include<iostream>
using namespace std;

int main(){
    string nama = "Program C++";
    string umur = "20 tahun";

    cout<<"Nama anda adalah "<< nama <<" Dan anda berusia "<< umur;


}
```

Gambar 1.27 simbol untuk menampilkan output di C++

Hasilnya :

```
Nama anda adalah Program C++ Dan anda berusia 20 tahun
--------------------------------
```

Gambar 1.28 hasil output pada C++

Jika ingin membuat program di atas di buat menggunakan dengan tiap baris maka kita harus menggunakan simbol “endl” atau “\n”. contoh:

```cpp
#include<iostream>
using namespace std;

int main(){
    string nama = "Program C++";
    string umur = "20 tahun";

    cout<<"Nama anda adalah "<< nama <<endl;
    cout<<"Dan anda berusia "<< umur<<endl;


}
```

Gambar 1.29 penulisan endl dan \n pada C++

Hasilnya :

```
Nama anda adalah Program C++
Dan anda berusia 20 tahun
```

Gambar 1.30 hasil penulisan endl dan \n

Contoh output lainnya :

```cpp
#include <iostream>
using namespace std;

int main(){

  cout << "=========== PROGRAM OUTPUT ===========" << endl;
  cout << "Program ini adalah program untuk\n";
  cout << "Menampilkan output ke layar komputer.\n";
  cout << "Ini adalah contoh Output sederhana\n";
  cout << "=========== TERIMA KASIH =============" << endl;


  return 0;
}
```

Gambar 1.31 penulisan endl dan \n pada C++

Hasilnya :

```
=========== PROGRAM OUTPUT ===========
Program ini adalah program untuk
Menampilkan output ke layar komputer.
Ini adalah contoh Output sederhana
=========== TERIMA KASIH =============
```

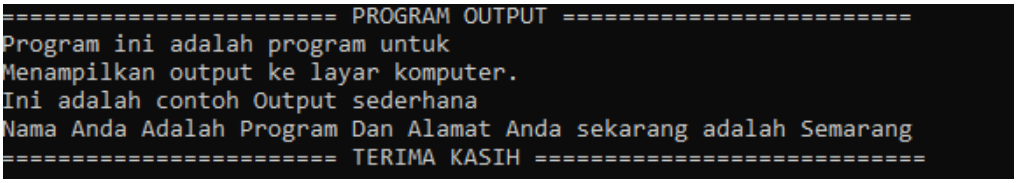
Gambar 1.32 hasil penulisan endl dan \n pada C++

Contoh :

```cpp
#include <iostream>
using namespace std;

int main(){

  cout << "======================== PROGRAM OUTPUT ========================" << endl;
  cout << "Program ini adalah program untuk\n";
  cout << "Menampilkan output ke layar komputer.\n";
  cout << "Ini adalah contoh Output sederhana\n";
  string nama = "Program";
  string alamat = "Semarang";

  cout<<"Nama Anda Adalah "<<nama<< " Dan Alamat Anda sekarang adalah "<< alamat<<endl;


  cout << "======================== TERIMA KASIH ==========================" << endl;


  return 0;
}
```

Gambar 1.33 penulisan endl dan \n pada C++

Hasilnya :



Gambar 1.34 hasil penulisan endl dan \n

b. output dengan menggunakan printf

printf adalah fungsi pemograman bahasa pemograman C, tetapi juga dapat digunakan dalam pemograman bahasa C++. Fungsi printf digunakan untuk mencetak nilai-nilai dengan format tertentu, seperti string, bilangan bulat, bilangan pecahan, dan sebagainya.

Contoh penggunaan printf:

```
#include <iostream>
using namespace std;

int main(){
    printf("Hello, World\n");
    printf("Nama saya %s\n", "Isi nama Anda"); //bagian isi nama anda, anda bisa mengganti dengan nama sendiri
    printf("Usia saya %d\n", 18);
    return 0;
}
```

Gambar 1.35 penulisan output menggunakan printf

Hasilnya :

```
Hello, World
Nama saya Isi nama Anda
Usia saya 18
```

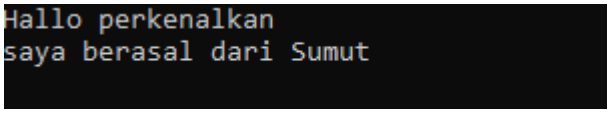
Gambar 1.36 hasil output menggunakan printf

Contoh :

```
#include <iostream>
using namespace std;

int main(){
    printf("Hallo perkenalkan\n");
    printf("saya berasal dari %s\n", "Sumut");

    return 0;
}
```

Gambar 1.37 penulisan output menggunakan printf

Hasilnya :



Gambar 1.38 hasil output menggunakan printf

Adapun simbol yang dapat kita gunakan dalam fungsi printf antara lain:

- %s adalah simbol untuk menampilkan nilai string.
- %d adalah simbol untuk menampilkan nilai angka atau bilangan desimal.
- \n adalah simbol untuk membuat baris baru.
- %c adalah simbol untuk menampilkan karakter.
- %d, % adalah simbol untuk menampilkan bilangan desimal.
- %fa adalah simbol untuk menampilkan bilangan pecahan.
- %o adalah simbol untuk menampilkan bilangan oktal.
- %x adalah simbol untuk menampilkan bilangan

heksadesimal.

- \t adalah simbol untuk membuat tab.

c. Input menggunakan “cin”

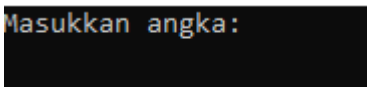
Cin merupakan bagian dari C++ yang digunakan untuk membaca input dari aliran input standar, yang biasanya keyboard. Cin memiliki kemampuan untuk membaca berbagai jenis data, seperti bilangan bulat, bilangan pecahan, karakter, dan string. Cin biasanya meminta pengguna memasukan data yang diminta dan setelah memasukan data maka inputan atau teks yang dimasukan akan di tampilkan ke layar komputer. Cout menggunakan simbol “<<” sedangkan cin menggunakan kebalikan simbol dari cout yaitu “>>”. Untuk menggunakan cin kita membutuhkan Extraction Operator dengan tanda >> yang diletakan di antara object cin dan ekspresi. Jadi penggunaan simbol ini sangat berpengaruh karna program c++ merupakan program sensitiv. Agar lebih jelas berikut ini kita akan mengenal bagaimana cara input dalam pemograman c++ dengan menggunakan cin.

Contoh input menggunakan integer:

```
#include <iostream>
using namespace std;

int main() {
    int angka;
    cout << "Masukkan angka: ";
    cin >> angka;
    cout << "Anda telah memasukan angka : " << angka << endl;

    return 0;
}
```

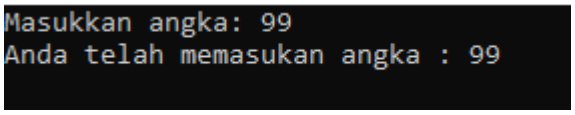
Gambar 1.39 penulisan input menggunakan cin

Codingan di atas meminta pengguna memasukan angka.



Gambar 1.40 hasil input menggunakan cin

hasil dari meminta input, jika misalnya kita memasukan angka 99 maka akan hasilnya akan menjadi.



Gambar 1.41 hasil input menggunakan cin

Ini merupakan program yang mencontohkan bagaimana penggunaan input output dengan cout dan cin. Pada baris ke empat didirikan variable dengan nama angka, dengan tipe data integer yang akan berfungsi

untuk penyimpanan data masukan dari pengguna program.

Contoh input menggunakan string;

```
#include <iostream>
using namespace std;

int main() {
    string nama,alamat, hobi;
    cout << "Masukkan Nama anda: ";
    cin >> nama;
    cout<<"masukan Alamat anda :";
    cin>> alamat;
    cout<<"Masukan Hobi anda   :";
    cin>>hobi;
    cout<<"Hallo "<< nama << " Alamat anda berada di "<< alamat << " hobi kamu adalah bermain "<< hobi<<endl;

    return 0;
}
```

Gambar 1.42 penulisan input menggunakan cin

Hasilnya :

```
Masukkan Nama anda: irwan
masukan Alamat anda :sumatra
Masukan Hobi anda   :futsal
Hallo irwan Alamat anda berada di sumatra hobi kamu adalah bermain futsal

--------------------------------
```

Gambar 1.43 hasil input menggunakan cin

Codingan ini meminta pengguna memasukan nama, alamat, dan hobi. Jika pengguna telah menginput semua data yang diminta maka akan menampilkan hasil seperti

gambar di atas. Untuk lebih jelas lagi kita akan membahas beberapa contoh input dan output pada c++.

Contoh input dan output:

```cpp
#include <iostream>
#include <string>

using namespace std;

int main() {
    // Menggunakan string untuk input dalam C++
    string nama;

    // C++: Menggunakan cout untuk output
    cout << "Masukkan nama Anda: ";

    // C++: Menggunakan getline untuk input string
    getline(cin, nama);

    // C++: Menampilkan hasil input
    cout << "Halo, " << nama << "! Selamat datang.\n";

    return 0;
}
```

Gambar 1.44 penulisan input dan output di C++

Hasilnya :

```
Masukkan nama Anda: irwan
Halo, irwan! Selamat datang.
```

Gambar 1.45 hasil input dan output di C++

Contoh input menggunakan karakter :

```
#include <iostream>
using namespace std;

int main() {
    char huruf;
    cout << "Masukkan sebuah huruf: ";
    cin >> huruf;
    cout << "Huruf yang dimasukkan: " << huruf << endl;
    return 0;
}
```

Gambar 1.46 cara input dengan char

Hasilnya :

```
Masukkan sebuah huruf: L
Huruf yang anda masukan adalah : L
```

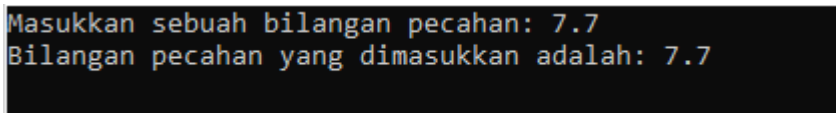
Gambar 1.47 hasil input menggunakan char

Contoh input menggunakan float :

```cpp
#include <iostream>
using namespace std;

int main() {
    float bilangan;
    cout << "Masukkan sebuah bilangan pecahan: ";
    cin >> bilangan;
    cout << "Bilangan pecahan yang dimasukkan adalah: " << bilangan << endl;
    return 0;
}
```

Gambar 1.48 penulisan input menggunakan float

Hasilnya :



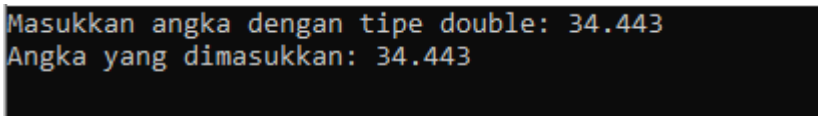
Gambar 1.49 hasil input menggunakan float

Contoh input menggunakan array :

```
#include <iostream>
using namespace std;
int main() {
    double angka;

    cout << "Masukkan angka dengan tipe double: ";
    cin >> angka;

    cout << "Angka yang dimasukkan: " << angka <<endl;

    return 0;
}
```

Gambar 1.50 penulisan input menggunakan array

Hasilnya :



Gambar 1.51 hasil input menggunakan array

e. Input menggunakan Scanf

Scanf digunakan untuk membaca input dari keyboard. Fungsi scanf terdiri dari dua paramter, yaitu format specifier dan variabel yang akan menampung input dari keyboard. Cara penulisan variabel untuk menampung input adalah didahului oleh tanda ampersand (‘&’) kemudian diikuti nama variabel. Tanda

ampersand artinya yang dilewatkan ke parameter adalah alamat memori variabel untuk diisi dengan nilai yang input diinput melalui keyboard.

Terdapat beberapa simbol yang digunakan untuk menentukan tipe data yang akan di scan. Berikut adalah beberapa di antaranya:

- %d: Untuk memindai nilai sebagai bilangan bulat (integer).
- %f: Untuk memindai nilai sebagai angka pecahan (float).
- %lf: Untuk memindai nilai sebagai angka pecahan dengan presisi ganda (double).
- %c: Untuk memindai satu karakter.
- %s: Untuk memindai sebuah string.
- %x, %X: Untuk memindai nilai dalam format heksadesimal.
- %o: Untuk memindai nilai dalam format oktal.

Contoh penggunaan scanf:

```
#include <cstdio>

int main() {
    int angka;
    printf("Masukkan sebuah angka: ");
    scanf("%d", &angka);
    printf("Angka yang Anda masukkan adalah: %d\n", angka);
    return 0;
}
```

Gambar 1.52 cara penulisan scanf

Hasilnya :

```
Masukkan sebuah angka: 78
Angka yang Anda masukkan adalah: 78
```

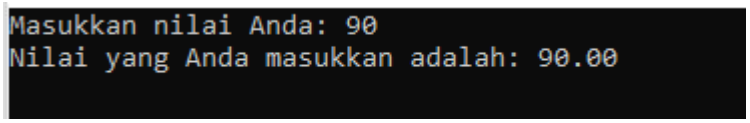
Gambar 1.53 hasil penulisan scanf

Contoh 2 :

```
#include <iostream>
#include <cstdio>
using namespace std;

int main() {
    float nilai;
    printf("Masukkan nilai Anda: ");
    scanf("%f", &nilai);

    printf("Nilai yang Anda masukkan adalah: %.2f\n", nilai);

    return 0;
}
```

Gambar 1.54 cara penulisan scanf

Hasilnya :



Gambar 1.55 hasil penulisan scanf

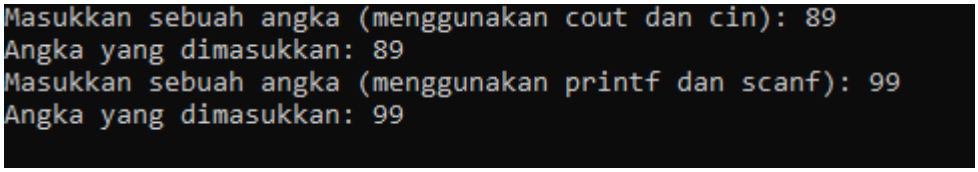
Setelah mengetahui cara input dan output dengan menggunakan cout, cin, printf, scanf kali ini kita akan menggunakan semua input dan output dalam membuat satu program c++.

Contoh penggunaan menggunakan integer sebagai berikut;

```
#include <iostream>
#include <cstdio>
using namespace std;

int main() {

 // Menggunakan cout dan cin
   int angka1;
   cout << “Masukkan sebuah angka (menggunakan cout dan cin): “;
   cin >> angka1;
   cout << “Angka yang dimasukkan: ” << angka1 << endl;
```

```
    // Menggunakan printf dan scanf
    int angka2;
    printf("Masukkan sebuah angka (menggunakan printf dan scanf): ");
    scanf("%d", &angka2);
    printf("Angka yang dimasukkan: %d\n", angka2);

    return 0;
}
```

Hasil:



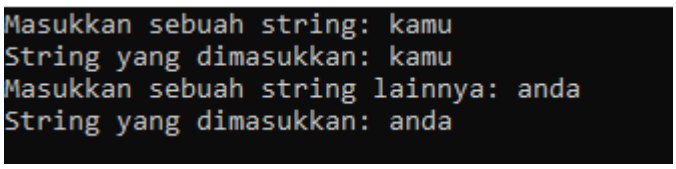
Gambar 1.56 penulisan scanf dan printf

Contoh penggunaan dengan string:

```
#include <iostream>

using namespace std;

int main() {
```

```
    // Menggunakan cout dan cin untuk input dan output string
    cout << “Masukkan sebuah string: “;
    string input_string;
    getline(cin, input_string);
    cout << “String yang dimasukkan: ” << input_string << endl;

    // Menggunakan printf dan scanf untuk input dan output string
    printf(“Masukkan sebuah string lainnya: “);
    char string_lain[100];
    scanf(“%s”, string_lain);
    printf(“String yang dimasukkan: %s\n”, string_lain);

    return 0;
}
```

Hasilnya:



Gambar 1.57 hasil penulisan scanf dan printf

2

File Header

FILE HEADER

File Header pada C++ adalah sebuah file yang berisi deklarasi fungsi, kelas variable global, dan mungkin juga definisi konstanta yang digunakan dalam program C++. File Header biasanya memiliki ekstensi ‘.h’ atau ‘.hpp’. File Header ini disertakan menggunakan preprocessor directive ‘#include’ di dalam file sumber utama program untuk menyertakan definisi-definisi yang diperlukan.

Dalam prakteknya, file header sering digunakan untuk menyimpan deklarasi fungsi, variable, dan makro yang digunakan diberbagai bagian dari program. Ini memungkinkan pengguna untuk memisahkan antara deklarasi dan implementasi, yang membantu dalam membagi kode program menjadi unit yang lebih kecil, lebih mudah dikelola, dan lebih mudah dipahami.

Berikut adalah beberapa fungsi penting dari file header pada C++ :

- Deklarasi Fungsi : File Header pada

umumnya berisi deklarasi dari semua fungsi yang digunakan di dalam program. Ini mencakup nama fungsi, tipe pengembalian, dan daftar parameter, Dengan menyertakan deklarasi fungsi dalam file header, program dapat dipanggil melalui fungsi-fungsi tersebut dari sumber lain tanpa perlu mendefinisikan ulang fungsi-fungsi tersebut.

- Deklarasi Kelas : File Header juga berisi deklarasi kelas, yang mencakup definisi kelas, anggota kelas, dan fungsi anggota kelas, Dengan menyertakan file header yang berisi deklarasi kelas, program dapat menggunakan kelas tersebut tanpa perlu mengetahuni detail implementasinya.

- Deklarasi Konstanta : Konstanta-konstanta yang digunakan diseluruh program juga dapat dideklarasikan dalam file header. Ini termasuk konstanta matematis seperti 'PI', konstanta enumerasi, atau konstanta-konstanta lain yang digunakan secara global dalam program.

- Deklarasi Variabel Global : Jika terdapat variable global yang akan digunakan diberbagai program, deklarasinya bisa dimasukan ke dalam file header. Ini memungkinkan akses ke variable global tersebut dari berbagai bagian program tanpa perlu mendefinisikan ulang.

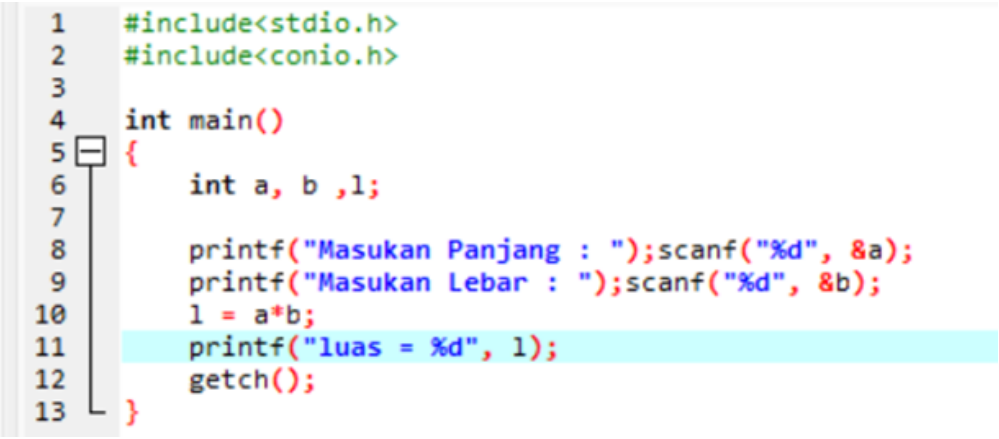
- Struktur Data dan Tipe Definisi : File Header juga dapat berisi deklarasi struktur data

seperti struct, class, atau union. Selain itu tipe data kustom yang didefinsikan oleh pengguna seperti typedef atau enum juga dapat dimasukan ke dalam file header. Ini mempermudah program untuk menggunakan tipe data kustom tersebut tanpa harus mendeklarasikannya ulang di setiap file sumber.

- Komentar dan Dokumentasi : File Header sering digunakan untuk memberikan dokumentasi tentang pengguna dan fungsi-fungsi yang dideklarasikan di dalamnya. Komentar dan dokumentasi ini membantu pengembang lain yang akan menggunakan file header tersebut untuk memahami cara menggunakan fungsi-fungsi atau struktur data yang didefinsikan

- Preprocessor Directives : Meskipun preprocessor directives seperti '#include' umumnya digunakan untuk menyertakan file header didalam file sumber, file header itu sendiri juga dapat menggunakan preprocessor directives untuk kondisional kompilasi atau untuk menyertakan file header lain yang diperlukan

Berikut adalah contoh file header sederhana dalam C++



```
#include<stdio.h>
#include<conio.h>

int main()
{
    int a, b ,l;

    printf("Masukan Panjang : ");scanf("%d", &a);
    printf("Masukan Lebar : ");scanf("%d", &b);
    l = a*b;
    printf("luas = %d", l);
    getch();
}
```

Gambar 2.1 penulisan file header sederhana

Dalam Bahasa pemrograman C++, file header memiliki peran penting dalam memisahkan deklarasi (decatation) dan implementasi (definition). Fungsi utama dari deklarasi-deklarasi yang terdapat di dalam file header adalah memberikan informasi kepada kompiler tentang fungsi,kelas,variable, atau konstanta sebelum implementasinya didefinisikan.

Berikut adalah penjelasan dan pengertian tentang fungsi dari deklarasi-deklarasi dalam file header :

- Memberikan Informasi tentang Entitas yang Dideklarasikan : Deklarasi-delkarasi pada file header memberikan informasi kepada kompiler tentang keberadaan fungsi, kelas,variable, atau konstanta sebelum implementasinya didefinisikan. Ini memungkinkan kompiler untuk mengetahuni tipe data yang akan digunakan dan bagaimana cara menggunakan sebelum melihan implementasinya.

- Mendukung Pemisahan Deklarasi dan Implementasi : File Header memungkinkan pemisahan yang jelas antara deklarasi dan implementasi. Deklarasi ditempatkan di. Ini file header, sedangkan implementasi ditempatkan yang berberbeda di dalam file sumber terpisah. Ini membantu dalam mengelola program yang lebih besar dengan membagi kode menjadi unit-unit yang lebih kecil dan terorganisir.

- Memudahkan Kompilasi Bersama : Dengan menggunakan file header, kompiler hanya perlu memeriksa deklarasi yang terdapat pada file header saat kompilasi. Ini menghindar kompiler harus mengompilasi ulang file sumber implementasi setiap ada perubahan. Sebagai hasilnya waktu kompilasi lebih cepat.

- Menghindar Duplikasi Deklarasi : Dengan menyertakan file header menggunakan preprocessor directives '#include', deklarasi-deklarasi tersebut hanya perlu ditulis sekali dalam file header. Ini menghindari duplikasi deklarasi dan memeastikan bahwa semua bagian program memiliki akses ke entitas yang sama.

- Memfasilitasi Pengguna Kode yang Reusable : File Header juga memfasilitasi pengguna kode yang reusable. Deklarasi-deklarasi yang ada di dalam file header dapat digunakan diberbagai bagian program atau bahkan diberbagai proyek yang berbeda dengan cara menyertakan file header yang sama.

Dengan demikian, file header dan deklarasi-deklarasinya memainkan peran penting dalam pengembangan perangkat lunak dengan C++. Mereka membantu dalam memisahkan deklarasi dari implementasi, memfasilitasi pemisahan kode, menghindari duplikasi, mempercepat waktu kompliasi, dan memungkinkan pengguna kode yang resuable.

File Header merupakan file-file yang berisi berbagai deklarasi, seperti fungsi, variable dan lain sebagainya. Di C++ file header diikutin dengan kata #include didepannya yang mengintruksikan kepada kompiler untuk menyiapkan file library nya.

Berikut adalah fungsi File Header beserta pengertiannya :

- Manajement memori
- Memanggil routin ROM BIOs
- Fungsi matematika kompleks
- Memanggil console DOS Input Output
- Routin basic di C++
- Menguakan manipulator
- Operasi string dan karakter

Manajement Memori

Fungsi file header dalam manajement memori dalam konteks pemrograman C++ dapat

bervariasi tergantung pada tujuan dan kompleksitas dari program yang dikembangkan. Namun secara umum, file header dalam manajement memori berperan penting dalam menyediakan deklarasi dan definisi fungsi-fungsi serta struktur data yang digunakan untuk mengelola memori secara efisien dalam sebuah program. Berikut adalah beberapa fungsi umum dari file header dalam manajement memori :

- Deklarasi Fungsi dan Struktur Data : File header biasanya akan mendefinisikan fungsi-fungsi dan struktur data yang digunakan dalam manajement memori. Termasuk fungsi-fungsi untuk alokasi, dealokasi, dan engelolaan memori lainnya seperti ‘malloc’, ‘free’, ‘new’, ‘delete’, serta struktur data seperti linked list, tree, atau queue yang digunakan untuk menyimpan Alamat memori yang dialokasikan.

- Mendefinisikan Konstanta dan Makro : File header dapat mendefinisikan konstanta dan makro yang digunakan dalam manajemen memori, seperti ukuran blok memori default, bilai-nilai spesifikasi yang digunakan dalam algoritma pengelolaan memori, atau makro untuk melakukan operasi operasi tertentu secara efisien.

- Dokumentasi : File header berisi komentar dan dokumentasi yang menjelaskan cara penggunaan fungsi-fungsi dan struktur data yang didefinisikan di dalamnya. Dokumentasi ini akan membantu pengembang lain yang menggunakan file header tersebut, untuk

memahami cara mengguakan fungsi-fungsi dan struktur data yang disediakan.

- Preprocessor Directives : File header juga dapat menggunakan preprocessor directives seperti ‘#ifndef’, ‘define’, dan ‘#endif’ untuk mencegah duplikasi dan memastikan bahwa file header hanya disertakan satu kali dalam suatu program.

- Mengelompokkan Fungsi dan Struktur Data Terkait : File header biasanya akan mengelompokan fungsi dan struktur data yang terkait dengan manajement memori ke dalam satu unit yang teripsah. Hal ini membantu dalam memperjelas organisasi kode dan memudahkan pemeliharaan.

Memanggil routin ROM BIOs

File header tidak secara langsung terlibat dalam pemanggilan rutin BIOs (Basic Input dan Output System) pada system. BIOs adalah perangkat lunak tingkat rendah yang berjalan pada hardware computer dan menyediakan fungsi dasar untuk mengakses dan menontrol perangkat keras, seperti keyboard, layer monitor, dan disk. BIOs biasanya terintergasi langsung dengan firmware hardware dan diakses oleh system operasi dan aplikasi melalui antarmuka yang ditentukan oleh spesifikasi perangkat keras seperti ACPI,UEFI, atau Legancy BIOs.

Namun dalam beberapa kasus, file header

dapat digunakan dalam pengembangan perangkat lunak yang berinteraksi dengan BIOs atau dengan perangkat keras secara langsung. Sebagai contohnya dalam pengembangan perangkat lunak tertentu, file header dapat mengandung deklarasi fungsi-fungsi yang memfasilitasi komunikasi langsung dengan perangkat keras melalui antarmuka yang disediakan oleh BIOs.

Berikut adalah beberapa fungsi memanggil routin ROM BIOs :

- Deklarasi Fungsi-fungsi Khusus : File header dapat menyertakan deklarasi fungsi-fungsi yang menghubungkan aplikasi dengan BIOs yang diperlukan untuk mengakses perangkat keras tertentu. Misalnya fungsi untuk membaca atau menulis ke sektor disk menggunakan layanan BIOs.

- Konstanta dan Struktur Data : File header dapat mendefinisikan konstanta dan struktur data yang diperlukan untuk berkomunikasi dengan BIOs atau untuk memformat atau mengelola data yang dikirim atau diterima BIOs.

- Dokumentasi dan Komentar : Seperti yang disebutkan sebelumnya, file header dapat berisi dokumentasi dan komentar yang menjelaskan cara menggunakan fungsi-fungsi dan struktur data yang didefinisikan didalamnya. Ini dapat membantu pengembang dalam memahami dan menggunakan interaksi dengan BIOs.

Namun, penting untuk dicatat bawha interaksi langsung dengan BIOs dalah praktik yang jarang dilakukan dalam pengembangan perangkat lunak modern, terutama pada system yang menjalankan system operasi modern seperti Windows,macOS, atau Linux. Sebagian besar pengembangan perangkat lunak berinteraksi dengan perangkat keras melalui antarmuka standar yang disediakan oleh system operasi, perangkat keras Tingkat tinggi seperti DirectX, Vulkan, atau OpenGL. Penggunaan BIOs secara langsung biasanya terbatas pada kasus-kasus Dimana akses langsung ke perangkat keras diperlukan, seperti dalam pengembangan firmware atau system operasi tertanam.

Matematika Kompleks

File header dalam matematika kompleks dapat digunakan untuk mendefinisikan struktur data dan fungsi-fungsi yang berhubungan dengan operasi pada bilangan kompleks. Bilangan kompleks adalah bagian yang terdiri dari bagian real dan bagian imajiner, direpresentasikan dalam bentuk a + bi, di mana a adalah bagian dari real, sedangkan b adalah bagian imajiner, dan i adalah unit imajiner (yang merupakan akar kuadrat dari -1).

Berikut adalah beberapa fungsi file header pada metematika kompleks :

- Deklarasi Struktur Data Kompleks : File

header mendefinisikan struktur data untuk merepresentasikan bilangan kompleks. Stuktur data ini biasanya akan memiliki dua baigan yang pertama bagian real dan bagian imajiner.

- Operasi Matematika : File header menyediakan deklarasi untuk fungsi-fungsi yang digunakan untuk melakukan operasi matematika pada bilangan kompleks, seperti penjumlahan, pengurangan, perkalian, dan pembagian.

- Fungsi-Fungsi Utilitas : Selain operasi matematika dasar, file header dapat menyertakan fungsi-fungsi untuk menghitung nilai mutlak bagian kompleks, menghitung konjugat, dan mengkonversi dari dan ke bntuk polar, dan sebagainya.

- Konstanta Matematika : File header juga bisa menyertakan definisi konstanta matematika yang berkaitan dengan bilangan kompleks, seperti unit imajiner dan nilai-nilai yang digunakan dalam matematika kompleks.

- Fungsi-fungsi Analisis : Jika aplikasi memerlukan analisis lebih lanjut dari bilangan kompleks, file header dapat menyediakan deklarasi untuk fungsi-fungsi seperti eksponensial kompleks, logaritma kompleks, fungsi trigonometrti kompleks, dan sebagainya.

Dengan menggunakan file header untuk matematika kompleks, pengembang dapat memisahkan deklarasi dari implementasi, yang memungkinkan operasi pada bilangan kompleks

diberbagai bagian program tanpa perlu mengetahui detail implementasinya. Ini dapat membantu dalam mengorganisir kode secara lebih baik, memfasilitasi pemeliharaan, dan memudahkan pengguna kode yang reusable untuk aplikasi yang memerlukan bagian kompleks.

Memanggil console DOS Input Output

Dalam konteks pemrograman C++ untuk console DOS (Disk Operating System), file header dapat digunakan untuk menyediakan fungsi-fungsi yang mempermudah input dan output data. Fungsi-fungsi ini sering digunakan untuk berinteraksi dengan pengguna dan menampilkan informasi ke layer. Berikut adalah beberapa fungsi umum yang bisa dimasukan ke dalam file header untuk memeprmudah penggunaan input dan output pada console DOS :

1. Fungsi- Fungsi Input : Fungsi-fungsi input digunakan untuk menerima input dari pengguna memalui console DOS. Beberapa adalah contoh fungsi input yang umum digunakan :

- ‘cin’ : Fungsi ini digunakan dalam C++ untuk merima input dari keyboard. Berikut contoh penggunaannya :

```
int angka;

std::cin >> angka;
```

- ‘scanf’ : Fungsi-fungsi ini dalah fungsi dari library C yang digunakan untuk mnerima

input dari pengguna. Berikut contoh penggunaannya :

Int angka;

scanf(“%d”, angka);

2. Fungsi-Fungsi Output : Fungsi-fungsi output digunakan untuk menampilkan informasi ke layer pengguna penggunaan di console DOS. Beberapa contoh fungsi output yang umum digunakan adalah :

- ‘cout’ : Fungsi ini digunakan dalam C++ untuk menampilkan teks atau nilai ke layer. Berikut adalah contoh penggunaannya :

std::cout << “Hello, Word!” <<std::endl;

- ‘printf’ : Fungsi ini adalah fungsi dari library C yang digunakan untuk menampilkan teks ke layer. Berikut contoh penggunaannya :

printf(“Hello, Word!\n”);

3. Manipulasi Teks : File header juga dapat menyertakan fungsi-fungsi untuk manipulasi teks, sperti pemformatan teks, pencarian, penggantian, atau pemotongan string. Berikut adalah contohnya :

- ‘strlen’ : Fungsi dari library C yang menghitung panjang sebuah string. Contoh penggunaannya :

char teks[] = “Hello”;

int panjang = strlen(teks);

Berikut Contoh File Header untuk Input/Output di Console DOS :

```
#ifndef CONSOLE_IO_H
#define CONSOLE_IO_H

#include <iostream> // Untuk cout dan cin
#include <cstdio>   // Untuk printf dan scanf
#include <cstring>  // Untuk strlen

// Fungsi untuk menampilkan teks ke layar menggunakan cout
void tampilkanPesan(const std::string& pesan) {
    std::cout << pesan << std::endl;
}

// Fungsi untuk menerima input menggunakan cin
void terimaInput(int& nilai) {
    std::cin >> nilai;
}

// Fungsi untuk menampilkan teks ke layar menggunakan printf
void tampilkanPesan(const char* pesan) {
    printf("%s\n", pesan);
}

// Fungsi untuk menerima input dari pengguna menggunakan scanf
void terimaInput(int* nilai) {
    scanf("%d", nilai);
}

// Fungsi untuk menghitung panjang string
int hitungPanjang(const char* teks) {
    return strlen(teks);
}

#endif
```

Gambar 2.2 penulisan file header untuk input dan output

Dalam contoh diatas, file header ‘CONSOLE_IO_H’ menyediakan fungsi-fungsi untuk menampilkan pesan ke layer dan menerima input dari pengguna menggunakan kedua metode C++ (menggunakan cout, dan cin) dan metode C (menggunakan printf dan scanf). Sealin itu, fungsi dari

hitungPanjang juga disertakan untuk mengintruksikan penggunaan fungsi manipulasi teks.

Dengan menggunakan file header seperti ini, pengguna dapat mempermudah proses input dan output di console DOS dalam program mereka. Ini sangat berguna untuk membuat program interaktif yang membutuhkan masukan dari pengguna dan memberikan respons yang sesuai.

Routin basic di C++

File header dalam rutin dasar di C++ memiliki peran yang sangat penting dalam pengorganisasian dan pengelolaan kode. Berikut penjelasan lengkap tentang fungsi file header dalam rutin dasar di C++ :

1. Deklarasi Fungsi :

• Antarmuka Publik : File header adalah tempat yang tapat untuk mendeklarasikan fungsi-fungsi yang akan digunakan oleh berkas sumber lain dalam program. Ini memberikan antarmuka public dari suatu modul atau Pustaka.

• Pemisahan Deklarasi dan Implementasi ; Dengan mendeklarasikan fungsi di file header dan mengimplementasikannya di berkas sumber yang terpisah, kita memisahkan antara yang dapat digunakan oleh pengguna dan bagimana fungsi tersebut diimplementasikannya.

2. Deklarasi Konstanta :

• Nilai Tetap : Konstanta yang dideklarsikan kedalam file header menyediakan nilai-nilai tetap yang dapat digunakan secara global dalam program.

- Mudah Pemeliharaan : Dengan mendefinisikan konstanta-konstanta di satu tempat, jika nilai konstanta perlu diubah kita hanya perlu memperbarui nilai di file header tanpa perlu mencari setiap pengguna konstanta tersebut diseluru kode.

3. Deklarasi Struktur dan Kelas :

- Abstrak Data : File header memungkinkan penggunaan struktur data dan kelas tanpa perlu mengetahui detail implementasinnya.

- Pemisahan Antarmuka dan Implementasi : Dengan mendeklarasikan struktur data dan kelas di file header, kita dapat memisahkan antara antarmuka public (bagaimana struktur dan kelas digunakan) dan implementasi privat (bagaimana struktur dan kelas diimplementasikan).

4. Deklarasi Fungsi Inline :

- Optimasi Performa : Fungsi-fungsi pendek yang didefinisikan dalam file header dan dinyatakan sebagai inline dapat dioptimalkan oleh kompiler untuk meningkatkan performa.

- Menghindari Overhead Panggilan Fungsi ; Dengan menjadikan fungsi-fungsi tersebut sebagai inline, kita dapat menghindari overhead panggilan fungsi yang terjadi saat memanggil fungsi secara konvesional.

5. Guard Clause :

- Mencegah Inklusi Ganda : Guard clause (biasanya menggunakan #ifndef, #define, dan #endif) digunakan untuk mencegah inklusi ganda dari file header yang sama dalam satu unit kompliasi.

- Mencegah Kesalahan Kompilasi : Dengan mencegah inkluasi ganda, kita dapat menghindari masalah penggandaan definisi yang dapat menyebabkan kesalahan kompilasi.

Manfaat Penggunaan File Header dalam Routin Dasar di C++ :

- Organisasi Kode yang Baik : File header memungkinkan pengelompokan deklarasi bersama untuk fungsi-fungsi terkait, konstanta-konstanta, struktur data, dan kelas-kelas.

- Keterbacaan Kode yang Ditingkatkan ; Dengan memisahkan deklarasi dari implementasi, file header membuat kode menjadi lebih mudah dimengerti.

- Penggunaan Kembali Kode yang Efisien : File header memungkinkan kode untuk digunakan Kembali dalam proyek-proyek lain dengan mudah.

- Pemeliharaan yang Mudah : Memisahkan deklarasi dari implementasi memudahkan pemeliharaan dari penguji kode, serta meminimalkan kesalahan.

Dengan memahami peran dan fungsi file header dalam rutin dasar di C++, pengembang dapat menginkatkan organisasi, keterbacaan, dan kegunaan kode, memudahkan kolaborasi dalam proyek tim.

Manipulator

Manipulator adalah fungsi-fungsi yang digunakan dalam C++ untuk mengontrol perilaku input/output pada (stream) seperti std::cin dan std::cout. Fungsi manipulator bekerja dengan operator << dan >> untuk mengatur format output dan mengubah input. File

header dapat digunakan untuk mendefinisikan dan mendeklarasikan manipulator kustom yang akan digunakan dalam program. Berikut adalah penjelasan lengkap tentang fungsi manipulator dalam file header pada C++ :

1. Manipulator Output (<<) :

Manipulator output digunakan untuk mengontrol bagaimana data ditampilkan ke aliran output misalnya std::cout.

- Manipulator Standar :

std::setw(int n) : Menentukan lebar output menjadi n karakter.

std::left dan std::right : Mengatur perataan output menjadi kiri atau kanan.

std::fixed dan std::scientific : Mengontrol cara bilangan pecahan ditampilkan (tetap atau ilmiah)

- Manipulator Kustom :

Mendefinisikan manipulator kustom untuk menyesuaikan tampilan output sesuai kebutuhan program. Contohnya adalah manipulator yang menambah symbol mata uang, mengatur presisi decimal, atau mengubah format Tunggal.

2. Manipulator Input (>>) :

Manipulator input digunakan untuk mengontrol cara data dibaca dari aliran input seperti std::cin.

- Manipulator Standar :

std::hex, std::oct, dan std::dec. : Mengontrol format pembacaan angka (heksadesimal, oktal, decimal).

std::skipws dan std::noskipws. : Mengatur apakah sepasi dan karakter whitespace lainnya diabaikan atau tidak saat membaca.

- Manipulator Kustom :

Mendefinisikan manipulator kustom untuk memodifikasi perilaku pembacaan input. Contohnnya adalah manipulator yang memfilter atau memvalidasi input, atau manipulator yang mengabaikan baris tertentu dalam input.

3. Definisi Manipulator :

Setelah deklarasi pada file header, manipulator perlu didefinisikan diberkas sumber terpisah.

Definisi manipulator kustom

```
std::ostream& currency(std::ostream& os) {
   // Implementasi logika manipulator
   os << “$”;
   return os;
}
```

4. Penggunaan di Berkas Sumber :

Manipulator dapat digunakan diberkas sumber setelah di-include file header yang berisi deklarasi dan definisi manipulator tersebut.

```
#include “manipulators.h”


int main() {
   int value = 1000;
   std::cout << “Value: ” << currency << value;
```

```
    return 0;
}
```

Manfaat Penggunaan Manipulator dalam File Header :

- Abstraksi Tingkat Tinggi : Manipulator memungkinkan penggunaan kode yang lebih bersih dan ekspresif dengan menyembunyikan detail implementasi.
- Reusabilitas : Manipulator yang didefinisikan di file header dapat digunakan kembali di berkas sumber lain dalam proyek yang sama.
- Keterbacaan Kode : Penggunaan manipulator membuat kode lebih mudah dimengerti dan dikelola dengan memisahkan logika tampilan dari logika bisnis.

Dengan menggunakan manipulator dalam file header, pengembang dapat mengingkatkan fleksibilitas dan kejelasan kode, serta memfasilitasi penggunaan yang lebih efisien dari aliran input/output dalam program C++

Macam-macam file header yang sering digunakan pada pemrograman C++ sebagai sebikut :

iostream

iostream adalah header file dalam C++ yang memfasilitasi operasi input/output (IO) stander. Ini adalah salah satu file header yang sering digunakan dalam pemrograman C++ karena menyediakan objek input dan output standar, seperti cin, dan cout, yang memungkinkan program untuk berinteraksi dengan

pengguna dan melakukan input/output standar ke konsol.

Berikut adalah beberapa komponen utama iostream :

1. Objek Input dan Output Standar :

- cin : Objek standar input cin digunakan untuk membaca data diri dari pengguna melalui keyboard. Data yang dimasukan bisa berupa tipe data, seperti interger, float, double, atau string.

- cout : Objek standar output cout digunakan untuk mencetak data ke layar. Data yang dicetak bisa berupa teks, nilai numerik, atau hasil operasi lainnya.

- cerr : Obejk standar output eror, cerr digunakan untuk mencetak suatu pesan kesalahan kedalam layar. Biasanya digunakan untuk mencetak pesan-pesan terkait dengan kegagalan operasi atau eror laiinya.

- clog : Objek standar output log clog digunakan untuk mencetak pesan log ke layar. Digunakan untuk mencatat kegiatan atau informasi tambahan selama program berjalan.

2. Operator Input dan Output :

- << (left shift operator): Digunakan untuk mengirim data ke objek output (cout, cerr, clog). Misalnya, cout << “Hello, Word!”.

- >> (right shift operator): Digunakan untuk menerima data dari objek input (cin). Misalnya, cin >> cum.

3. Manipulator :

Manipulator adalah fungsi-fungsi yang digunakan bersama dengan objek output untuk mengatur format output. Beberapa manipulator yang sering digunakan adalah :

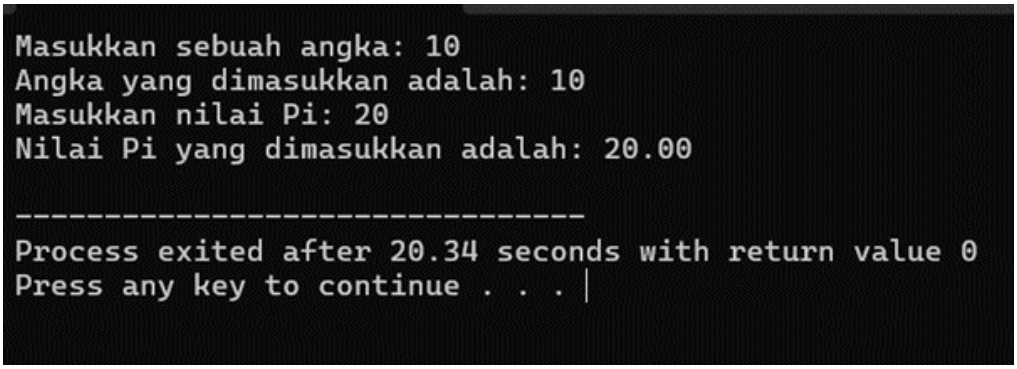
- std::endl : Digunakan untuk membuat baris baru dalam output.
- std::setw(int) : Digunakan untuk menetapkan lebar output.
- std::setprecision(int) : Digunakan untuk menetapkan presisi angka decimal dalam output.

Contoh program menggunakan iostream :

```cpp
#include <iostream>
#include <iomanip>

int main() {
    int num;
    double pi;

    std::cout << "Masukkan sebuah angka: ";
    std::cin >> num;
    std::cout << "Angka yang dimasukkan adalah: " << num << std::endl;

    std::cout << "Masukkan nilai Pi: ";
    std::cin >> pi;
    std::cout << "Nilai Pi yang dimasukkan adalah: " << std::fixed << std::setprecision(2) << pi << std::endl;

    return 0;
}
```

Gambar 2.3 penulisan file header iostream

Hasil:



Gambar 2.4 hasil penulisan file header iostream

Dalam program diatas, cin digunakan untuk membaca input dari pengguna, dan cout digunakan untuk mencetak output ke layer. Selain itu manipulator seperti std::endl dan std::setprecision untuk mengatur format output sesuai kebutuhan

Dengan menggunakan file header iostream, dapat mempermudah melakukan input/output dalam program C++ dengan menggunakan cin, dan cout, serta memanfaatkan fitur-fitur lainnya yang disediakan untuk mengatur format dan memanipulasi output.

iostream.h

iostream.h adalah header file yang digunakan dalam versi awal dari Bahasa pemrograman C+=. iostream.h digunakan untuk mengatur operasi input/output (IO) standar dalam program C++, serupa dengan iostream. Namun iostream.h sudah tidak disarankan lagi karena sudah using dan tidak termasuk dalam standar C++ modern.

Fungsi-fungsi utama yang disediakan oleh

iostream.h mirip dengan iostream, tetapi berbeda dalam penggunaannya :

1. Objek Input dan Output Standar :

- cin : Objek standar input cin digunakan untuk membaca data dari pengguna melalui keyboard
- cout : Objek standar output cout digunakan untuk mencetak data ke layer
- endl : Digunakan untuk membuat baris baru dalam output.
- flush : Digunakan untuk mengosongkan buffler output.

2. Pengunaan Namespace :

- Dalam iostream.h , tidak ada penggunaan namespace std. Oleh karena itu tidak perlu menulis lagi std:: sebelum cin dan cout.
- Namum, ini bisa menyebabkan konflik nama jika ada dua fungsi atau variable dengan nama yang sama.

3. Keterbatasan :

- iostream.h adalah bagian dari C++ yang lebih tua dan mungkin tidak didukung lagi sepenuhnya oleh kompiler yang lebih modern.
- Dalam standar C++ yang lebih baru, penggunaan iostream. direkomendasikan sebagai penggantinya. Penggunaan iostream lebih portable dan sesuai dengan standar yang lebih modern.

Perbedaan iostream.h dengan iostream :

Iostream.h dan iostream adalah dua file header

yang berfungsi serupa dalam pemrograman C++, namun iostream.h adalah bagian dari versi awal Bahasa pemrograman C++ sedangkan iostream adalah bagian dari standar yang lebih modern. Berikut adalah perbedaan kedua program tersebut :

iostream.h:

1. Fungsi:

- iostream.h digunakan untuk operasi input-output (IO) dalam bahasa pemrograman C++.
- Menyediakan objek input-output standar seperti cin, cout, cerr, dan clog.
- Objek cin digunakan untuk membaca data dari pengguna melalui keyboard.
- Objek cout digunakan untuk mencetak data ke layar.
- Objek cerr digunakan untuk mencetak pesan-pesan kesalahan di layar.
- Objek clog digunakan untuk mencetak pesan-pesan log ke layar.

2. Perbedaan dengan iostream:

- Tidak menggunakan namespace std. Oleh karena itu, tidak perlu menulis std:: sebelum cin dan cout.
- Manipulator seperti endl, setw, setprecision, dan lainnya tidak tersedia dalam iostream.h. Harus menggunakan header tambahan seperti iomanip.h untuk manipulator tertentu.
- Digunakan dalam versi awal dari bahasa

pemrograman C++ dan mungkin tidak didukung sepenuhnya oleh kompiler modern.

- Penggunaannya tidak disarankan dalam pengembangan program yang lebih modern dan portabel.

iostream:

1. Fungsi:

- iostream juga digunakan untuk operasi input-output (IO) dalam bahasa pemrograman C++.
- Menyediakan objek input-output standar seperti cin, cout, cerr, dan clog.
- Objek cin digunakan untuk membaca data dari pengguna melalui keyboard.
- Objek cout digunakan untuk mencetak data ke layar.
- Objek cerr digunakan untuk mencetak pesan-pesan kesalahan kedalam layar.
- Objek clog digunakan untuk mencetak pesan-pesan log ke layar.

2. Perbedaan dengan iostream.h:

- Menggunakan namespace std. Oleh karena itu perlu menulis std:: sebelum cin dan cout jika ingin menggunakannya.
- Manipulator seperti endl, setw, setprecision, dan lainnya tersedia dalam iostream.
- Direkomendasikan untuk penggunaan dalam

pengembangan program yang lebih modern dan portabel.

stdio.h

stdio.h adalah salah satu header file standar dalam bahasa pemrograman C yang digunakan untuk operasi input/output. Meskipun umumnya digunakan dalam bahasa C, dalam bahasa C++ kita dapat menggunakan versi modernnya yaitu cstido, yang memiliki fungsi-fungsi yang sama.

Berikut beberapa fungsi utama yang disediakan oleh stdio.h, atau cstdio dalam bahasa C++ :

1. Operasi Input-Output Standar :

- printf() : Digunakan untuk mencetak data ke layer dengan format tertentu.
- scanfI() : Digunakan untuk membaca data dari pengguna dengan format tertentu.
- getchar() : Digunakan untuk membaca satu karakter dari input standar.
- putchar() : Digunakan untuk mencetak satu karakter ke output standar.
- gets() : Digunakan untuk membaca string dari input standar.
- puts() : Digunakan untuk mencetak string ke output standar.

2. Operasi Posisi File :

- fseek() : Digunakan untuk merubah posisi pointer file.

- ftell() : Digunakan untuk memberikan posisi saat ini dari pointer file.
- rewid() : Digunakan untuk memindahkan pointer file ke awal dari file.

3. Operasi File :

- fopen() : Digunakan untuk membuka suatu file.
- fclose() : Digunakan untuk menutup suatu file.
- fprintf() : Digunakan untuk mencetak data kedalam file dengan format tertentu.
- fscanf() : Digunakan untuk membaca data dari file dengan format tertentu.

4. Operasi Karakter :

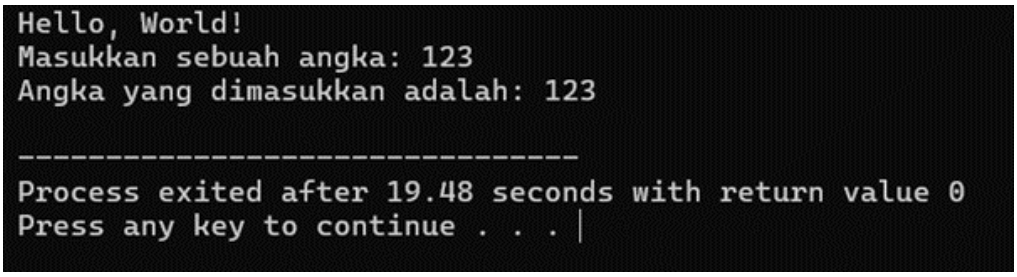
- isdigit(), isalpha(), isalnum(), isspace(), dan lain lain : Digunakan untuk menguji sifat karakter (digit, huruf, karakter alfanumerik, spasi dan sebagainya.)
- tolower(), toupper() : Digunakan untuk mengubah karakter menjadi huruf besar dan kecil.

Berikut Contoh Program Menggunakan stdio.h :

```c
#include <stdio.h>

int main() {
    printf("Hello, World!\n");

    int num;

    printf("Masukkan sebuah angka: ");
    scanf("%d", &num);

    printf("Angka yang dimasukkan adalah: %d\n", num);

    return 0;
}
```

Gambar 2.5 penulisan file header stdio.h

Hasil :



Gambar 2.6 hasil penulisan file header stdio.h

Dalam program diatas menggunakan fungsi printf() untuk mencetak teks ke layar, scanf() untuk membaca angka, dan kemudian printf() digunakan untuk mencetak angka tersebut ke layar. Perhatikan bahwa format specifier %d digunakan untuk scanf() dan printf() untuk membaca dan mencetak interger.

Perlu diingat bahwa meskipun menggunakan header file stdio.h, masih bisa menggunakan fungsi-fungsi ini dalam bahasa C++. Meskipun demikian, untuk pengembangan program C++ yang lebih modern, disarankan untuk menggunakan header file yang sesuai dengan standar C++ iostram dan cstdio untuk memanfaatkan fitur-fitur yang lebih canggih dan lebih aman.

Berikut perbedaan utama antara stdio.h dan cstdio terletak pada penggunaan namespace dan kompatbilitas dengan C++ :

1. Namespace :

- stdio.h : Tidak temasuk dalam namespace std. Oleh karena itu, semua fungsi dan tipe data dalam stdio.h tersedia secara global, dan tidak perlu menambahkan std:: sebeleum menggunakannya.

- cstdio : Termasuk dalam namespace std. Semua fungsi dan tipe dalam cstdio terletak dalam namespace std, sehingga perlu menambahkan std:: sebelum menggunakannya dalam program C++

2. Kompatibilitas C++ :

- stdio.h : Merupakan bagian dari standar C, dan digunakan dalam bahasa C untuk operasi input-output (IO). Meskipun dapat digunakan dalam C++, penggunaannya tidak disarankan dalam pengembangan program C++ yang modern.

- cstido : Merupakan versi modern dari stdio.h untuk pengembangan dalam program C++. Meskipun fungsinya serupa dengan stdio.h, penggunaan cstdio lebih disukai dalam pengembangan program C++ modern karena sesuai dengan standar C++ dan dapat menangani spesifikasi dalam bahasa C++ dengan lebih baik.

Dengan demikian stdio.h lebih umum digunakan dalam pengembangan program dalam bahasa C, sedangkan cstdio lebih disukai dalam pengembangan program C++ untuk memanfaatkan fitur-fitur modern dan memastikan kompatbilitas dengan standar C++.

conio.h

conio.h adalah bagian dari library standar yang umumnya digunakan dalam pengembangan C++ pada

platform Windows. Namun, perlu diperhatikan bahwa conio.h bukanlah bagian dari standar bahasa C atau C++, dan fungsi-fungsinya mungkin tidak tersedia di semua lingkungkan pengembangan atau system operasi.

Berikut beberapa fungsi utama yang disediakan oleh conio.h antara lain :

1. Input Tanpa Buffering :

- getch() : Digunakan untuk membaca satu karakter dari keyboard tanpa menampilkan karakter yang dibaca. Input langsung diterima dan dikembalikan ke program.
- getche() : Sama seperti getch(), tetapi karakter yang dibaca ditampilkan dilayar.

2. Manipulasi Layar :

- clrscr() : Digunakan untuk membersihkan layar konsol.
- gotoxy(x, y) : Digunakan untuk memindahkan kursor ke koordinat (x, y) di layar konsol.

3. Warna Teks :

- textcolot(color) : Digunakan untuk mengatur sebuah warna teks yang akan ditampilkan ke dalam layar.
- textbackground(color) ; Digunakan untuk mengatur sebuah warna latar belakang teks.

Contoh Program Menggunakan conio.h :

```cpp
#include <iostream>
#include <conio.h>

int main() {
    char ch;

    std::cout << "Tekan tombol apa saja untuk melanjutkan..." << std::endl;

    ch = getch();

    std::cout << "Anda menekan" << ch << std::endl;

    return 0;
}
```

Gambar 2.7 penulisan file header conio.h

Hasil :

```
Tekan tombol apa saja untuk melanjutkan...
Anda menekan

--------------------------------
Process exited after 11.03 seconds with return value 0
Press any key to continue . . .
```

Gambar 2.8 hasil penulisan file header conio.h

Dalam program diatas, menggunakan fungsi getch() dari conio.h untuk membaca satu karakter dari keyboard tanpa menunggu tombol “Enter”. Setelah tombol ditekan, program akan melanjutkan eksekusi dan mencetak karakter yang ditekan ke layar.

Catatan Penting :

- Header file conio.h tidak termasuk dalam standar C atau C++, dan fungsinya sangat bergantung pada system operasi. Oleh karena itu, penggunaannya

tidak disarankan dalam pengembangan program yang portable atau lintas platform.

- Meskipun conio.h umumnya digunakan dalam pengembangan program C++ pada platform Windows, ada alternatif yang lebih portable dan sesuai dengan standar C++, seperti menggunakan header file iostream untuk operasi input-output standar.

math.h

math.h atau cmath dalam bahasa C++ adalah bagian dari library standar yang menyediakan fungsi-fungsi matematika untuk operasi-operasi seperti perhitungan trigonometri, perpangkatan, akat kuadrat, logaritma, dan sebagainya.

Berikut beberapa fungsi utama yang disediakan oleh math.h atau cmath antara lain :

1. Fungsi Trigonometri :

- sin(x) : Untuk menghitung nilai-nilai sinus dari sudut ‘x’ (dalam radian).
- cos(x) : Untuk menghitung nilai- nilai kosinus dari sudut ‘x’ (dalam radian).
- tan(x) : Untuk menghitung nilai-nilai tangen dari sudut ‘x’ (dalam radian).
- asin(x) : Untuk menghitung nilai-nilai arcsinus dari ‘x’, mengembalikan hasil dalam radian.
- acos(x) ; Untuk menghitung nilai-nilai arccosinus dari ‘x’, mengembalikan hasil dalam radian.
- atan(x) : Untuk menghitung nilai-nilai

arctangen dari ‘x’, menghitung hasil dalam radian.

2. Fungsi Perpangkatan dan Akar :

- pow(x, y) : Untuk menghitung nilai ‘x’ dipangkatkan dengan ‘y’.
- sqrt(x) : Untuk menghitung nilai akar kuadrat dari ‘x’.
- cbrt(x) : Untuk menghitung nilai akar kubik dari ‘x’.

3. Fungsi Logaritma :

- log(x) : Untuk menghitung logaritma natural dari ‘x’.
- log10(x) : Untuk menghitung logaritma basis 10 dari ‘x’

4. Fungsi Lainnya :

- fabs(x) : Unuk mengembalikan nilai absolut dari ‘x’.
- ceil(x) ; Untuk mengembalikan nilai ‘x’ dibulatkan ke atas (ke nilai interger terdekat yang lebih besar atau sama dengan ‘x’).
- floor(x) : Untuk mengembalikan nilai ‘x’ dibulatkan ke bawah (ke nilai interger terdekat yang lebih kecil atau sama dengan ‘x’).
- round(x) : Untuk mengembalikan nilai ‘x’ dibulatkan ke nilai interger terdekat.

Contoh Program Menggunakan math.h :

```cpp
#include <iostream>
#include <math.h>

int main() {
    //Menggunakan fungsi sin() untuk menghitung sinus dari sudut 45 derajat
    double sinus_45 = sin(45 * M_PI / 180); // M_PI adalah konstanta untuk nilai Pi

    std::cout << "Sinus dari sudut 45 derajat: " << sinus_45 << std::endl;

    //Menggunakan fungsi pow() untuk menghitung 2 pangkat 3
    double pangkat = pow(2, 3);

    std::cout << "2 pangkat 3 adalah: " << pangkat << std::endl;

    //Menggunakan fungsi sqrt() untuk menghitung akar kuadrat dari 16
    double akar_kuadrat = sqrt(16);

    std::cout << "Akar kuadrat dari 16 adalah: " << akar_kuadrat << std::endl;

    //Menggunakan fungsi log() untuk menghitung logaritma natural dari 10
    double logaritma = log(10);

    std::cout << "Logaritma natural dari 10 adalah: " << logaritma << std::endl;

    return 0;
}
```

Gambar 2.9 penulisan file header math.h

Hasil :

```
Sinus dari sudut 45 derajat: 0.707107
2 pangkat 3 adalah: 8
Akar kuadrat dari 16 adalah: 4
Logaritma natural dari 10 adalah: 2.30259

--------------------------------
Process exited after 7.439 seconds with return value 0
Press any key to continue . . .
```

Gambar 2.10 hasil penulisan file header math.h

Dalam program di atas. Menggunakan beberapa fungsi dari math.h untuk melakukan berbagai operasi matematika :

- sin() : Digunakan untuk menghitung sinus dari sudut 45 derajat.

- pow() : Digunakan untuk menghitung 2 pangkat 3.
- sqrt() : Digunakan untuk menghitung akar kuadrat dari 16.
- log() : Digunakan untuk menghitung logaritma natural dari 10.

Berikut perbedaan utama antara math.h dan csmath terletak pada penggunaan namespace dan kompatibilitas dengan C++ :

1. Namespace :

- math.h : Fungsi-fungsi matematika dalam math.h tidak termasuk dalam namespace std. Oleh karena itu, dapat digunakan langsung dalam program tanpa menambahkan std:: sebelumnya.
- cmath : Fungsi-fungsi matematika dalam cmath didefinisikan dalam namespace std. Ini berarti bahwa untuk menggunakan fungsi-fungsi tersebut, perlu menambahkan std:: sebelumnya.
- Contoh penggunaan:

math.h : double result = sin(x);

cmath : double result = std::sin(x);

2. Kompatibilitas C++ :

- math.h : Merupakan bagian dari standar bahasa C, sehingga tidak sepenuhnya sesuai dengan standar bahasa C++. Meskipun masih dapat digunakan dalam program C++, penggunaannya tidak disarankan dalam pengembangan program yang lebih modern.
- cmath : Merupakan bagian dari standar C++,

sehingga lebih sesuai untuk pengembangan program dalam bahasa C++ yang modern.

3. Standar C++ :

- math.h : Merupakan bagian dari standar bahasa C.
- cmath : Merupakan bagian dari standar bahasa C++.

4. Portabilitas :

- cmath : Lebih portabel karena sesuai dengan standar C++ dan dapat digunakan secara konsisten di berbagai platform.
- math.h : Masalah portabilitas tergantung pada kompiler dan platform. Beberapa implementasi kompiler mungkin memiliki implementasi yang berbeda untuk math.h.

5. Rekomendasi Penggunaan :

- csmath : Disarankan untuk pengembangan program C++ yang modern karena sesuai dengan standar C++ dan memanfaatkan namespace std, yang memberi kemudahan pemeliharaan dan mengurangi resiko tabrakan nama.
- math.h : Mungkin masih diperlukan dalam pengembangan program dilingkungan yang memerlukan atau mendukung pengguna header file kuno dari bahasa C. Namun, untuk pengembangan program yang lebih modern, disarankan untuk menggunakan csmath untuk portabilitas yang lebih baik.

stdlib.h

stdlib.h adalah header file standar dalam bahasa pemrograman C dan C++ yang menyediakan berbagai fungsi utilitas untuk alokasi memori, pengelolaan string, pengaturan proses, dan fungsi-fungsi lainnya.

Berikut adalah beberapa fungsi yang disediakan stdilb.h dan contoh penggunaannya dalam program C++ :

1. Alokasi Memori Dinamis :

- malloc(), calloc(), realloc() : Mengalokasikan memori secara dinamis.
- free() : Membebaskan memori yang telah dialokasikan sebelumnya.

2. Konversi String ke Tipe Data Numerik:

- atoi(), atol(), atoll() : Mengonversi string menjadi bilangan bulat.
- strtof(), strtod(), strtold() : Mengonversi string menjadi bilangan pecahan (float, double, long double).
- strtol(), strtoll(), strtoul(), strtoull() : Mengonversi string menjadi bilangan bulat dengan basis tertentu.

3. Konversi Tipe Data Numerik ke String:

- itoa(), ltoa(), lltoa() : Mengonversi bilangan bulat menjadi string.
- ftoa(), dtostr(), ldtostr() : Mengonversi bilangan pecahan menjadi string.

4. Penanganan Kesalahan :

- abort() : Memaksa program untuk berhenti.
- exit() : Mengakhiri program dengan status

tertentu.

- atexit() : Mendaftarkan fungsi yang akan dipanggil saat program keluar.

5. Pengendalian Program :

- system() : Menjalankan perintah di shell.

6. Fungsi-fungsi Pembangkit Bilangan Acak :

- rand(), srand() : Menghasilkan bilangan acak.

7. Pengelolaan Lingkungan :

- getenv() : Mendapatkan nilai dari sebuah variabel lingkungan.
- putenv(), setenv(), unsetenv() : Memanipulasi variabel lingkungan.

8. Fungsi-fungsi Pemrosesan Kata :

- bsearch() : Mencari elemen dalam array terurut.
- qsort() : Mengurutkan elemen dalam array.

Contoh Program Menggunakan stdlib.h :

```cpp
#include <iostream>
#include <time.h>
#include <stdlib.h>

using namespace std;

int main() {
    int i, j;

    srand((unsigned) time(NULL));

    // mencetak 10 angka acak
    for (i = 0; i < 10; i++) {
        // mencetak angka acak
        j = rand();
        cout <<"Angka acak : " << j << endl;
    }

    return 0;
}
```

Gambar 2.11 penulisan file header stdlib.h

Hasil :

```
Angka acak : 31383
Angka acak : 21406
Angka acak : 291
Angka acak : 7291
Angka acak : 13735
Angka acak : 25524
Angka acak : 31564
Angka acak : 25597
Angka acak : 7539
Angka acak : 20646

--------------------------------
Process exited after 7.8 seconds with return value 0
Press any key to continue . . .
```

Gambar 2.12 hasil penulisan file header stdlib.h

stdlib.h biasanya digunakan dalam bahasa C++, meskipun implementasi C++ modern cenderung lebih memilih header file C++ yang setara, seperti cstdlib, cstring, cmath, dan lainnya.

Perbedaan antara stdlib.h dan cstdlib :

1. Namespace : Dalam bahasa C++, cstdlib adalah versi dari stdlib.h yang dimasukkan ke dalam namespace std. Artinya, semua fungsi dan simbol yang didefinisikan dalam cstdlib terdapat di dalam namespace std. Sebagai contoh, fungsi malloc() dalam stdlib.h akan disebut std::malloc() dalam cstdlib.

2. Kompatibilitas : cstdlib lebih sesuai digunakan dalam lingkungan pemrograman C++ modern. Header file stdlib.h lebih umum digunakan dalam kode C atau dalam lingkungan yang memerlukan kompatibilitas mundur dengan kode C lama.

3. Praktik Penggunaan : Dalam kode C++ modern, lebih disarankan untuk menggunakan cstdlib daripada stdlib.h karena lebih konsisten dengan praktik pengkodean C++ dan menggunakan namespace std.

4. Header File : cstdlib adalah bagian dari standar bahasa C++ dan didefinisikan dalam header file cstdlib, sementara stdlib.h adalah header file dari bahasa C yang masih didukung untuk kompatibilitas dengan kode C lama.

Secara fungsional, tidak ada perbedaan signifikan antara stdlib.h dan cstdlib. Keduanya menyediakan fungsi-fungsi utilitas yang sama untuk alokasi memori, konversi string, pengelolaan proses, dll. Perbedaan

utamanya adalah dalam praktik penggunaan dan namespace.

string.h

Dalam pemrograman C++, string.h adalah header file yang sering menyediakan berbagai fungsi untuk menipulasi string. Meskipun penggunaannya umum dalam bahasa C, tetapi beberapa fungsi string.h juga dapat digunakan dalam C++. Namun dalam pengembangan C++ modern, lebih umum menggunakan header file cstring atau string.

Berikut adalah beberapa fungsi yang disediakan oleh string.h beserta penjelasannya:

1. Manipulasi String:

- strcpy() : Menyalin string dari satu variabel ke variabel lainnya.
- strcat() : Menggabungkan dua string dengan menambahkan string kedua ke belakang string pertama.
- strlen() : Menghitung panjang string, yaitu jumlah karakter sebelum null terminator.

2. Pencarian dan Manipulasi Karakter:

- strchr() : Mencari kemunculan pertama dari karakter tertentu dalam string.
- strstr(): Mencari kemunculan pertama dari substring tertentu dalam string.

3. Perbandingan String:

- strcmp() : Membandingkan dua string secara leksikografis.

- strncmp() : Membandingkan dua string hanya untuk sejumlah karakter tertentu.

4. Manipulasi Memori:

- memset() : Mengatur sejumlah byte dalam memori menjadi nilai tertentu.
- memcpy() : Menyalin sejumlah byte dari satu lokasi memori ke lokasi lainnya.

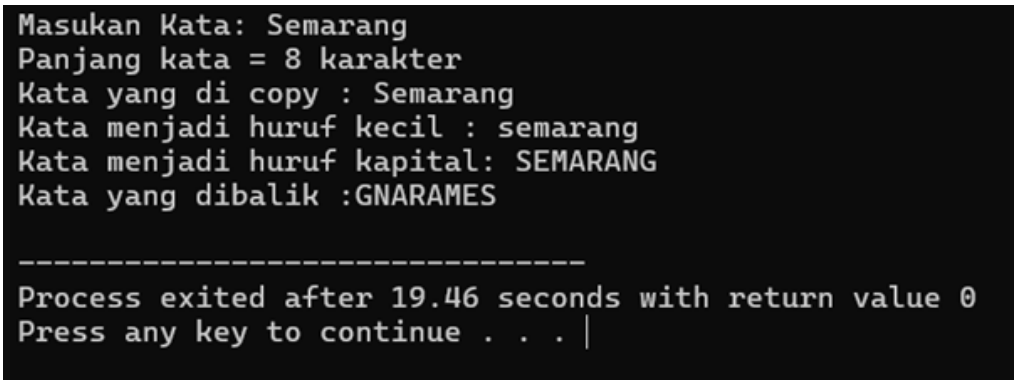
Fungsi-fungsi ini sangat berguna dalam manipulasi string dan memori dalam bahasa C++. Namun, dalam pengembangan C++ modern, lebih disarankan untuk menggunakan objek string std::string dari heade string karena lebih aman dan mudah digunakan.

Contoh program menggunakan string.h :

```
#include <iostream>
#include <conio.h>
#include <stdio.h>
#include <string.h>

using namespace std;

int main() {
    char kata[50];
    char kata2[50];

    cout << "Masukan Kata: ";
    gets(kata);

    cout << "Panjang kata = " << strlen(kata) << " karakter" << endl;

    strcpy(kata2, kata);
    strlwr(kata);

    cout << "Kata yang di copy : " << kata2 << endl;
    cout << "Kata menjadi huruf kecil : " << kata << endl;

    strupr(kata);
    cout << "Kata menjadi huruf kapital: " << kata << endl;

    strrev(kata);
    cout << "Kata yang dibalik :" << kata << endl;

    getch();
    return 0;
}
```

Gambar 2.13 penulisan file header string.h

Hasil :



Gambar 2.14 cara penulisan file header string.h

Perbedaan string.h dan cstring

1. string.h :

- string.h adalah header file dalam bahasa C yang menyediakan fungsi-fungsi untuk memanipulasi string dalam bentuk array karakter null-terminated.
- Fungsi-fungsi dalam string.h sering digunakan dalam pemrograman C untuk memanipulasi string, seperti mencari panjang string (strlen()), menyalin string (strcpy(), strncpy()), menggabungkan string (strcat()), membandingkan string (strcmp(), strncmp()), dan sebagainya.
- Fungsi-fungsi dalam string.h memanipulasi string yang diimplementasikan sebagai array karakter.

2. cstring :

- cstring adalah bagian dari C++ Standard

Library yang menyediakan kelas std::string dan fungsi-fungsi yang terkait dengan manipulasi string.

- cstring menyediakan objek string yang dikelola secara dinamis, yang memungkinkan manipulasi string dengan lebih mudah dan aman dibandingkan dengan array karakter null-terminated.

- cstring juga menyediakan fungsi-fungsi tambahan untuk manipulasi string, seperti pencarian substring, penggantian karakter, pemotongan string, dan sebagainya.

Perbedaan utama antara string.h dan cstring adalah bahwa string.h berisi fungsi-fungsi untuk manipulasi string menggunakan array karakter null-terminated, sementara cstring menyediakan kelas std::string dan fungsi-fungsi yang terkait dengan manipulasi string yang lebih aman dan mudah digunakan dalam C++. Dalam pengembangan aplikasi C++, disarankan untuk menggunakan std::string dari cstring karena lebih aman, nyaman, dan sesuai dengan paradigma pemrograman C++ modern.

iomanip.

iomanip adalah header file dalam C++ yang berisi definisi dari manipulator yang digunakan untuk memanipulasi format input/output (IO). Manipulator adalah fungsi yang digunakan dalam C+= untuk mengontrol aliran input/output. iomanip menyediakan sejumlah manipulator yang digunakan untuk mengatur format output yang dihasilkan oleh program C++.

Berikut beberapa fungsi dari iomanip :

1. Manipulator Format Bidang (std::setw(), std::setfill(), std::left, std::right, std::internal):

- std::setw(int n) : Menetapkan lebar bidang untuk data berikutnya yang akan dicetak.
- std::setfill(char c) : Mengatur karakter pengisi untuk bidang kosong dalam cetakan.
- std::left, std::right, std::internal : Mengatur rata kiri, rata kanan, atau rata tengah untuk nilai dalam bidang.

2. Manipulator Format Numerik (std::setprecision(), std::fixed, std::scientific):

- std::setprecision(int n) : Menetapkan jumlah angka desimal yang akan dicetak untuk nilai floating point.
- std::fixed : Mengatur format cetakan floating point menjadi tetap (non-scientific).
- std::scientific : Mengatur format cetakan floating point menjadi notasi ilmiah.

3. Manipulator Kontrol IO (std::endl, std::flush, std::setw):

- std::endl : Memasukkan karakter baris baru dan mengosongkan buffer.
- std::flush : Mengosongkan buffer tanpa memasukkan karakter baris baru.
- std::setw : Menetapkan lebar bidang untuk nilai yang akan dicetak.

4. Manipulator Format Numerik Lainnya (std::hex, std::oct, std::dec):

- std::hex, std::oct, std::dec : Mengatur basis numerik untuk cetakan (heksadesimal, oktal, desimal).

5. Manipulator Format Mata Uang (std::showbase, std::put_money, std::get_money):

- std::showbase : Menampilkan basis numerik (prefix) untuk nilai numerik.
- std::put_money() : Menampilkan nilai sebagai mata uang.
- std::get_money() : Membaca nilai sebagai mata uang.

6. Manipulator Lainnya (std::noshowpoint, std::noshowpos, std::noskipws, std::uppercase):

- std::noshowpoint : Menyembunyikan angka desimal yang tidak signifikan.
- std::noshowpos : Menyembunyikan tanda positif (+) untuk nilai positif.
- std::noskipws : Menonaktifkan pengabaian whitespace pada operasi input.
- std::uppercase : Mengonversi huruf ke dalam huruf kapital pada cetakan.

Manipulator dalam iomanip memungkinkan untuk mengontrol tampilan output secara lebih detail, seperti menetapkan lebar bidang, mengatur format numerik, menampilkan nilai sebagai mata uang, dan banyak lagi. Dengan menggunakan manipulator ini, dapat menghasilkan output yang sesuai dengan kebutuhan aplikasi dan lebih mudah dibaca.

Contoh program menggunakan iomanip :

```cpp
#include <iostream>
#include <iomanip>

int main() {
    // Menampilkan judul
    std::cout << std::setw(20) << std::left << "Nama" << std::setw(10)
    << "Umur" << std::setw(10) << "Nilai" << std::endl;

    // Menampilkan data
    std::cout << std::setw(20) << std::left << "Kevin"
    << std::setw(10) << 25 << std::setw(10)
    << std::setprecision(2) << std::fixed << 85.5 << std::endl;
    std::cout << std::setw(20) << std::left << "Jason Smith"
    << std::setw(10) << 30 << std::setw(10)
    << std::setprecision(2) << std::fixed << 90.75 << std::endl;

    return 0;
}
```

Gambar 2.15 penulisan file header iomanip

Hasil :

```
Nama                Umur      Nilai
Kevin               25        85.50
Jason Smith         30        90.75

--------------------------------
Process exited after 0.07767 seconds with return value 0
Press any key to continue . . .
```

Gambar 2.16 hasil penulisan file header iomanip

Program diatas menggunakan manipulator dari iomanip untuk memanipulasi format output. ini termasuk std::setw() untuk menentukan lebar bidang, std::left untuk rata kiri, std::setprecision() untuk mementukan presisi angka decimal, dan std::fixed untuk memastikan angka decimal ditampilkan dengan presisi tetap.

3
Type Data

DATA TYPE

Type data adalah klasifikasi yang digunakan untuk menunjukkan jenis nilai yang dapat disimpan dalam suatu variabel atau digunakan dalam suatu program. Ini bisa berupa tipe data dasar seperti integer(bilangan bulat), float(bilangan desimal), string(teks), boolean(nilai benar atau salah), atau tipe data yang lebih kompleks seperti array, list, dan dictionary. Setiap tipe data memiliki sifat dan operasi yang berbeda yang dapat diterapkan padanya.Tipe data pemrograman merupakan atribut yang berkaitan dengan data yang akan memberi tahu sistem komputer. Sehingga nantinya bisa menafsirkan nilai dari data tersebut. Secara khusus, tipe data adalah format penyimpanan data.Data ini bisa dalam bentuk variabel untuk tipe data tertentu.Berikut ini beberapa tipe dan penjelasannya yang sering digunakan :

No	Type	Penjelasan
1.	Integer	Sering juga ditulis dengan int, integer merupakan tipe data dalam bentuk bilangan bulat. Umumnya data ini digunakan untuk menyimpan angka tanpa pecahan. Tipe data ini tidak memiliki komponen pecahan.
2.	Float	Tipe data float digunakan untuk menyimpan bilangan pecahan. Pada umumnya, tipe data ini digunakan untuk melakukan perhitungan matematika yang melibatkan angka desimal.
3.	String	Dalam pemrograman, "string" merujuk pada serangkaian karakter yang diurutkan. Karakter dalam string dapat berupa huruf, angka, atau simbol lainnya. String biasanya digunakan

Gambar 3.1 tabel penjelasan tipe data

		untuk merepresentasikan teks dan data dalam program. Misalnya, "Hello, World!" adalah contoh string yang umum digunakan dalam pemrograman.
4.	Boolean	Tipe data ini hanya memiliki dua nilai: benar (true) atau salah (false). Tipe data boolean digunakan untuk mengekspresikan kondisi logis dalam suatu program, seperti apakah suatu pernyataan benar atau salah. Ini sering digunakan dalam struktur pengendalian seperti percabangan (if-else statements) dan perulangan (loops) untuk mengontrol alur eksekusi program.
5.	Array	Array disebut juga larik adalah struktur data yang menyimpan sekumpulan/sederetan variabel yang bertipe data sama. Setiap variabel dapat diakses dengan menggunakan suatu indeks sebagai identitas/alamat. Array adalah struktur data yang statis artinya: – Jumlah variabel harus diketahui sebelum dijalankan .
6.	List	"list" adalah struktur data yang dapat menyimpan kumpulan elemen dalam urutan tertentu. Setiap elemen dalam list memiliki posisi yang disebut indeks, yang dimulai dari 0. List sangat berguna untuk menyimpan dan mengelola data dalam program, seperti daftar nama, nilai-nilai yang dikumpulkan dari input pengguna, atau hasil dari operasi pengolahan

Gambar 3.2 tabel penjelasan tipe data

		data.
7.	Dictionary	Dictionary adalah struktur data yang menyimpan pasangan kunci-nilai. Setiap elemen dalam dictionary terdiri dari dua bagian: kunci yang bersifat unik dan nilai yang terkait dengan kunci tersebut. Kunci digunakan untuk mengakses nilai yang sesuai dalam dictionary. Dictionary sering digunakan untuk merepresentasikan data yang terstruktur, seperti kamus, daftar kontak, atau konfigurasi aplikasi.

Gambar 3.3 tabel penjelasan tipe data

Type Data Dalam Bahasa C

Type	Length	Range
Unsigned char	8 bits	0 to 255
Char	8 bits	-128 to 127
Short int	16 bits	-32,768 to 32,767
Unsigned int	32 bits	0 to 4,294,967,295
Int	32 bits	-2,147,483,648 to 2,147,483,648
Unsigned long	32 bits	0 to 4,294,967,295
Enum	16 bits	-2,147,483,648 to 2,147,483,648
Long	32 bits	-2,147,483,648 to 2,147,483,648

Type	Lenght	Range
Float	32 bits	3.4 x 10-38 to 3.4 x 10+38
Double	64 bits	1.7 x 10-308 to 1.7 x 10+308
Long double	80 bits	3.4 x 10-4932 to 3.4 x 10+4932

Gambar 3.4 ukuran panjang tipe data

Near(pointer)	32 bits	Not applicable
Far(pointer)	32 bits	Not applicable

Gambar 3.5 ukuran panjang tipe data

Type data dalam pemrograman juga mempunyai beberapa fungsi,yang pertama ada Validasi Data Tipe yaitu data memastikan bahwa nilai yang dimasukkan sesuai dengan format yang diharapkan. Misalnya, jika variabel bertipe integer, maka hanya nilai numerik bulat yang dapat dimasukkan.Kedua, Optimasi Memori yaitu tipe data membantu dalam alokasi memori yang efisien untuk menyimpan nilai. Tipe data yang tepat dapat menghemat ruang memori dan meningkatkan kinerja program.Ketiga, Operasi yang Dapat Dilakukan yaitu tipe data membatasi jenis operasi yang dapat dilakukan pada nilai. Misalnya, operasi matematika hanya dapat dilakukan pada nilai numerik, sedangkan operasi string hanya dapat dilakukan pada teks.

Keempat, Keamanan yaitu dengan menggunakan tipe data yang tepat, programmer dapat memastikan bahwa operasi yang dilakukan pada nilai sesuai dengan kebutuhan aplikasi, sehingga mengurangi risiko terjadinya kesalahan atau kerentanan keamanan.Kelima, Ketelitian dan Konsistensi yaitu penggunaan tipe data yang tepat membantu menjaga ketelitian dan konsistensi dalam pengolahan data, sehingga meminimalkan kemungkinan terjadinya kesalahan logika atau interpretasi yang salah.Keenam, Pemahaman Kode yaitu dengan melihat type data suatu variabel, programmer dapat dengan cepat

memahami jenis nilai yang diolah dan operasi yang mungkin dilakukan pada variabel tersebut, memudahkan dalam pemeliharaan dan pengembangan kode.

Dengan demikian, fungsi tipe data sangat penting dalam pengembangan perangkat lunak untuk memastikan keandalan, keamanan, dan kinerja aplikasi. Berikut adalah contoh pemrograman dari beberapa type data yang sudah di bahas sebelumnya :

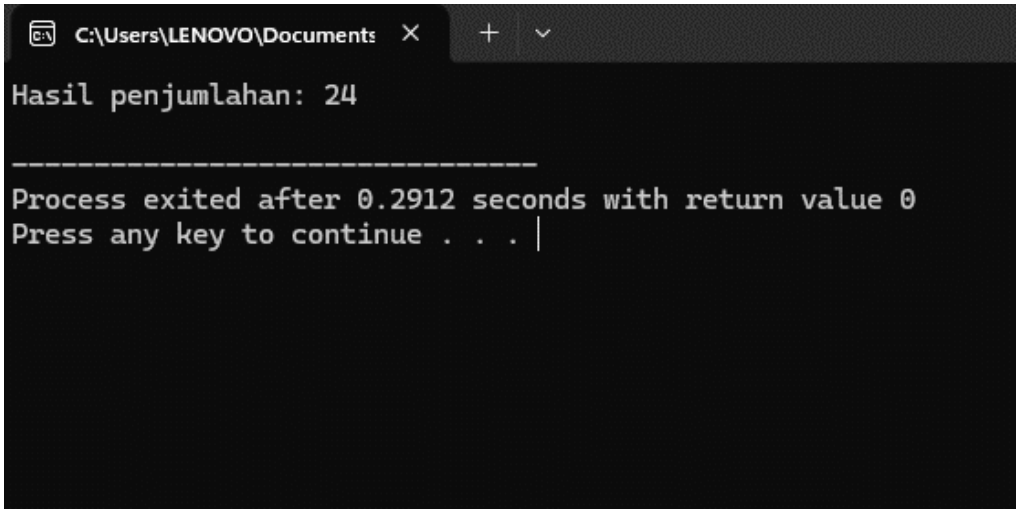
- CONTOH DARI PEMROGRAMAN INTEGER

contohinteger.cpp

```cpp
#include <iostream>

int main() {
    // Mendeklarasikan variabel integer
    int angka1 = 8;
    int angka2 = 16;
    int hasil;

    // Melakukan operasi penjumlahan
    hasil = angka1 + angka2;

    // Menampilkan hasil penjumlahan
    std::cout << "Hasil penjumlahan: " << hasil << std::endl;

    return 0;
}

```

Gambar 3.6 penulisan tipe data integer

Hasil dari contoh pemrograman diatas:



Gambar 3.7 hasil penulisan tipe data integer

Dalam contoh di atas, ini mendeklarasikan dua variabel bertype integer, angka1 dan angka2, dengan nilai masing-masing 8 dan 16. Kemudian, tambahkan kedua variabel tersebut dan menyimpan hasilnya ke dalam variabel hasil. Akhirnya, kita mencetak hasil penjumlahan menggunakan std::cout.

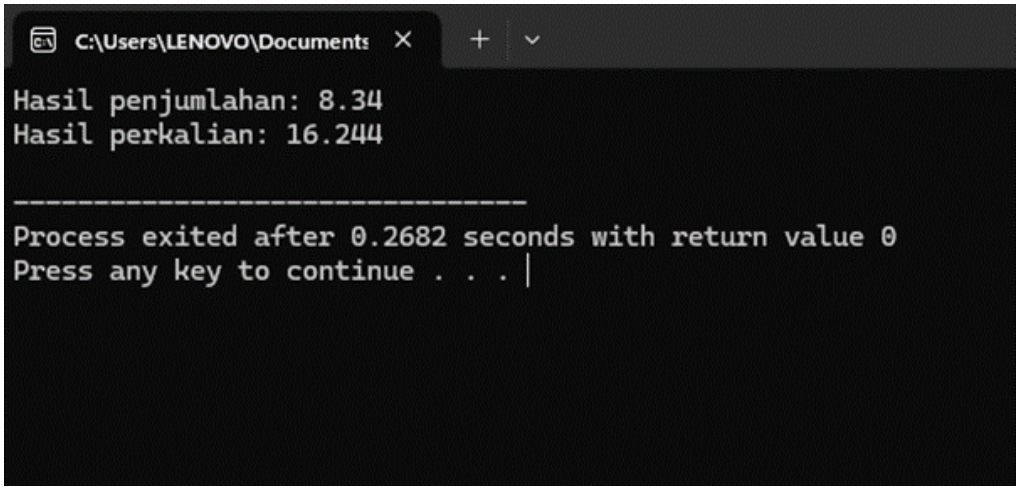
- CONTOH DARI PEMROGRAMAN FLOAT

contohinteger.cpp contohfloat.cpp

```cpp
#include <iostream>
using namespace std;

int main() {
    // Mendeklarasikan variabel bertipe float
    float nilaiFloat1 = 5.24;
    float nilaiFloat2 = 3.10;

    // Melakukan operasi aritmatika dengan tipe data float
    float hasilPenjumlahan = nilaiFloat1 + nilaiFloat2;
    float hasilPerkalian = nilaiFloat1 * nilaiFloat2;

    // Menampilkan hasil operasi
    cout << "Hasil penjumlahan: " << hasilPenjumlahan << endl;
    cout << "Hasil perkalian: " << hasilPerkalian << endl;

    return 0;
}
```

Gambar 3.8 penulisan tipe data float

Hasil dari contoh pemrograman diatas :



Gambar 3.9 hasil penulisan tipe data float

Dalam contoh tersebut, ini mendeklarasikan dua variabel dengan type data float (nilaiFloat1 dan nilaiFloat2), melakukan operasi penjumlahan dan perkalian menggunakan variabel-variabel tersebut, dan menampilkan hasilnya.

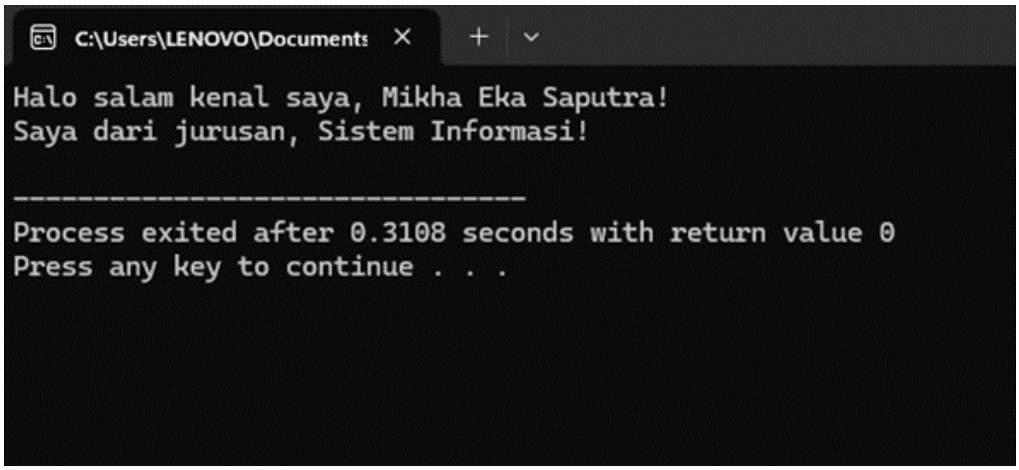
• CONTOH DARI PEMROGRAMAN STRING

contohinteger.cpp | contohfloat.cpp | function_string.cpp | contohstring.cpp

```cpp
#include <iostream>
#include <string>

int main() {
    std::string nama = "Mikha Eka Saputra";
    std::cout << "Halo salam kenal saya, " << nama << "!" << std::endl;
    std::string jurusan = "Sistem Informasi";
    std::cout << "Saya dari jurusan, " << jurusan << "!" << std::endl;
    return 0;
}
```

Gambar 3.10 penulisan tipe data string

Hasil dari contoh pemrograman diatas :



Gambar 3.11 hasil penulisan tipe data string

Dalam contoh tersebut, ini mendeklarasikan dua variabel dengan type data float (nilaiFloat1 Dalam contoh ini, kita menggunakan kelas string dari pustaka standar C++ (#include <string>) untuk mendeklarasikan

variabel nama dan jurusan sebagai string dan menginisialisasinya dengan nilai “Mikha Eka Sapautra” dan”Sistem Informasi”. Kemudian, kita mencetak pesan “Halo salam kenal saya,” dan “Saya dari jurusan” diikuti dengan nilai dari variabel nama,jurusan, dan diakhiri dengan tanda seru menggunakan std::cout.

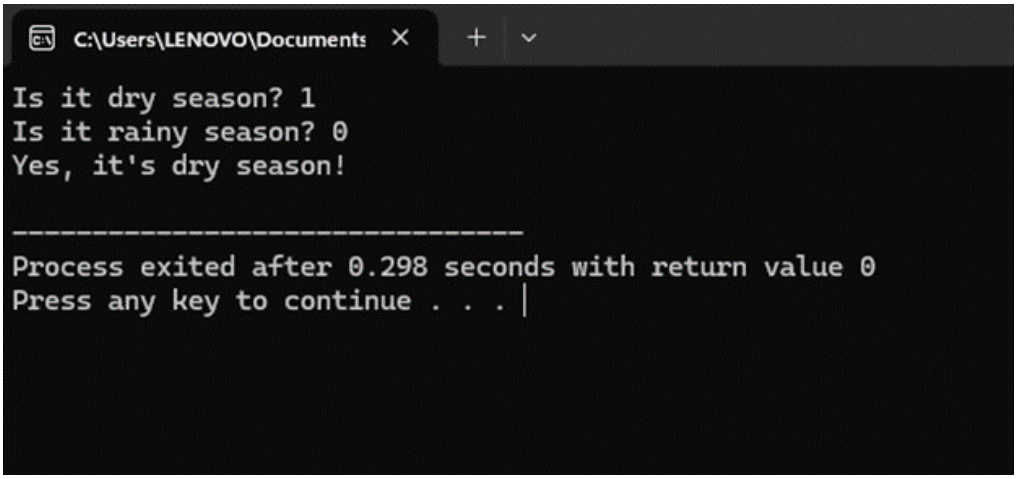
- CONTOH DARI PEMROGRAMAN BOOLEAN

contohboolean.cpp

```cpp
#include <iostream>
using namespace std;

int main() {
    bool isDry = true;
    bool isRainy = false;

    cout << "Is it dry season? " << isDry << endl;
    cout << "Is it rainy season? " << isRainy << endl;

    if (isDry) {
        cout << "Yes, it's dry season!" << endl;
    } else {
        cout << "No, it's not dry season." << endl;
    }

    return 0;
}
```

Gambar 3.12 penulisan tipe data boolean

Hasil dari contoh pemrograman diatas :



Gambar 3.13 hasil penulisan tipe data boolean

Dalam contoh diatas,ini mendeklarasikan dua variabel boolean isDry dan isRainy.Kemudian mencetak nilai variabel variabel tersebut,kemudian yang terakhir membuat percabangan dengan menggunakan if else untuk menentukan apakah musim kemarau atau musim hujan.

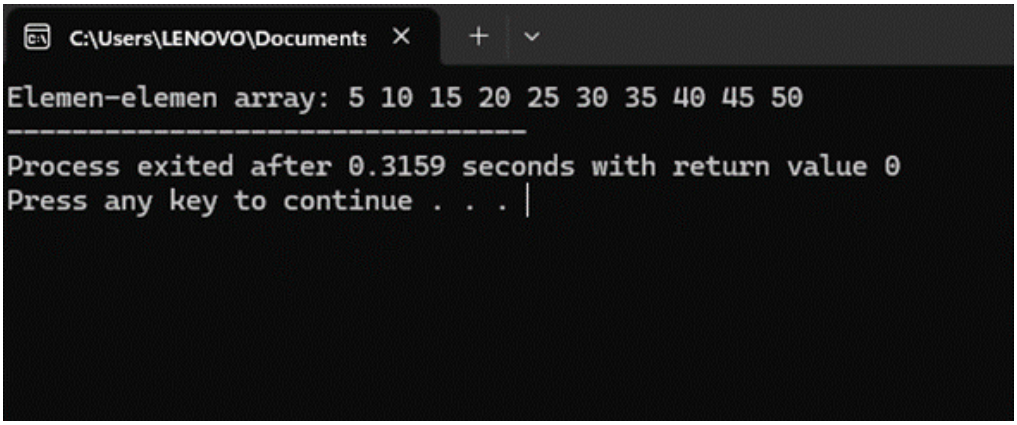
- CONTOH DARI PEMROGRAMAN ARRAY

contohboolean.cpp ContohArray.cpp

```cpp
#include <iostream>

int main()
{
    // Mendeklarasikan array dengan 5 elemen bertipe integer
    int angka[10];

    // Menginisialisasi elemen-elemen array
    angka[0] = 5;
    angka[1] = 10;
    angka[2] = 15;
    angka[3] = 20;
    angka[4] = 25;
    angka[5] = 30;
    angka[6] = 35;
    angka[7] = 40;
    angka[8] = 45;
    angka[9] = 50;
    angka[10] = 55;

    // Mengakses dan mencetak nilai elemen-elemen array
    std::cout << "Elemen-elemen array: ";
    for(int i = 0; i < 10; ++i) {
        std::cout << angka[i] << " ";
    }

    return 0;
}
```

Gambar 3.14 penulisan tipe data array

Hasil dari contoh pemrograman diatas :



Gambar 3.15 hasil penulisan tipe data array

Output dari pemrograman ini yaitu mencetak :

Elemen-elemen array: 5 10 15 20 25 30 35 40 45 50

Dalam contoh diatas,ini mendklarasikan array angka dengan 10 elemen bertipe integer.Kemudian menginisialisasi masing-masing elemen array dengan nilai tertentu.Selanjutnya menggunakan loop for untuk mengakses dan mencetak nilai dari semua elemen array.

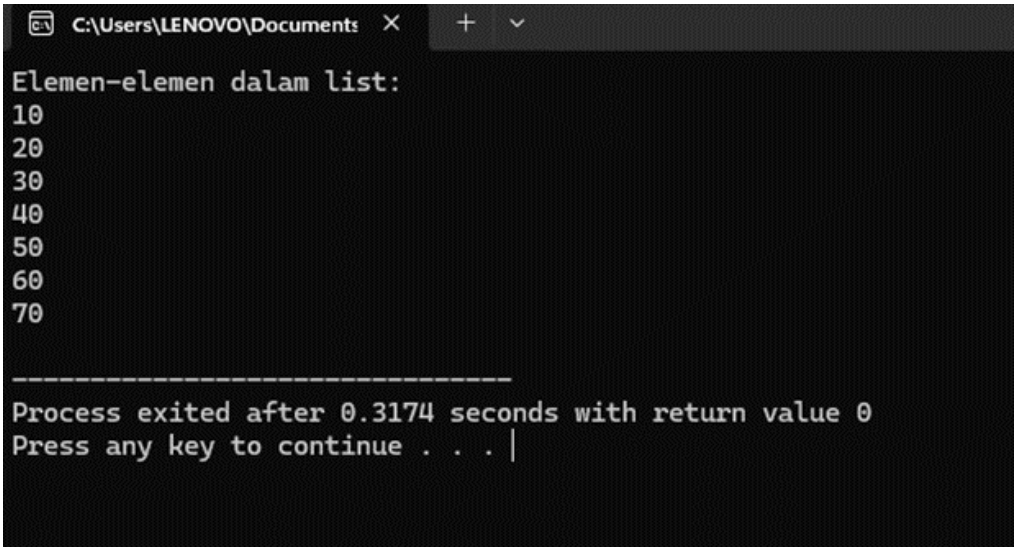
- CONTOH DARI PEMROGRAMAN LIST

contohboolean.cpp | ContohArray.cpp | ContohList.cpp

```cpp
#include <iostream>
#include <vector>

using namespace std;

int main() {
    // Mendeklarasikan variabel list
    vector<int> numbers;

    // Menambahkan elemen ke dalam list
    numbers.push_back(10);
    numbers.push_back(20);
    numbers.push_back(30);
    numbers.push_back(40);
    numbers.push_back(50);
    numbers.push_back(60);
    numbers.push_back(70);

    // Mengakses dan menampilkan elemen dari list
    cout << "Elemen-elemen dalam list:" << endl;
    for (int i = 0; i < numbers.size(); ++i) {
        cout << numbers[i] << endl;
    }

    return 0;
}
```

Gambar 3.16 penulisan tipe data list

Hasil dari contoh pemrograman diatas :



Gambar 3.17 hasil penulisan tipe data list

Dalam contoh ini, kita menggunakan vector dari STL C++ untuk mendefinisikan list. Kemudian,di tambahkan beberapa elemen ke dalam list menggunakan push_back() dan menampilkan elemen-elemennya dengan melakukan perulangan melalui list tersebut.

- CONTOH DARI PEMROGRAMAN DICTIONARY

contohboolean.cpp | ContohArray.cpp | ContohList.cpp | Contohdictionary.cpp | c++0x_warning.h

```cpp
#include <iostream>
#include <map>
#include <string>

using namespace std;

int main() {
    // Membuat dictionary dengan key bertipe string dan value bertipe int
    map<string, int> data;

    // Menambahkan elemen ke dalam dictionary
    data["anggur"] = 10;
    data["mangga"] = 20;
    data["kelapa"] = 15;

    // Mengakses nilai dengan menggunakan key
    cout << "Jumlah anggur: " << data["anggur"] << endl;
    cout << "Jumlah mangga: " << data["mangga"] << endl;
    cout << "Jumlah kelapa: " << data["kelapa"] << endl;

    // Mengubah nilai
    data["anggur"] = 5;
    cout << "Setelah diubah, jumlah anggur: " << data["anggur"] << endl;

    // Menghapus elemen dari dictionary
    data.erase("mangga");

    // Menampilkan semua elemen dalam dictionary
    cout << "Elemen dalam dictionary setelah menghapus mangga:" << endl;
    // Mengubah nilai
    data["anggur"] = 5;
    cout << "anggur:" << data["anggur"] << endl;
    // Mengubah nilai
    data["kelapa"] = 15;
    cout << "kelapa:" << data["kelapa"] << endl;

return 0;
}
```

Gambar 3.18 penulisan tipe data dictionary

Hasil dari contoh pemrograman diatas :

```
C:\Users\LENOVO\Documents
Jumlah anggur: 10
Jumlah mangga: 20
Jumlah kelapa: 15
Setelah diubah, jumlah anggur: 5
Elemen dalam dictionary setelah menghapus mangga:
anggur:5
kelapa:15

--------------------------------
Process exited after 0.3003 seconds with return value 0
Press any key to continue . . .
```

Gambar 3.19 hasil penulisan tipe data dictionary

Dalam contoh di atas, map<string, int> data adalah sebuah dictionary yang menggunakan string sebagai kunci (key) dan integer sebagai nilai (value). Kita dapat menambahkan, mengakses, mengubah, dan menghapus elemen-elemen dalam dictionary tersebut menggunakan berbagai operasi yang disediakan oleh kelas map.

Ada beberapa materi yang akan dibahas yaitu,tentang kode penentu format,increment dan decrement,operator aritmatika dan hierarki operator aritmatika,relational operations dan logical operators.

1. KODE PENENTU FORMAT

Kode penentu format dalam C++ yaitu istilah yang mengacu pada mekanisme yang digunakan untuk mengontrol tampilan output dari data, seperti angka, teks, dan variabel lainnya. Kode ini biasanya digunakan bersama dengan fungsi output.Cara memasukkan/menampilkan hasil Output tersebut, kita harus menggunakan sebuah kode format didalam fungsi printf,seperti:

- %c : Membaca sebuah karakter
- %s : Membaca sebuah string
- %i, %d : Membaca sebuah bilangan bulat (integer)
- %f, %e : Membaca sebuah bilangan pecahan (real)

- %o : membaca sebuah bilangan octal
- %x : Membaca sebuah bilangan heksadesimal
- %u : Membaca sebuah bilangan tak bertanda
- %lf : Membaca sebuah bilangan pecahan berganda(double)

2. INCREMENT DAN DECREMENT

Increment adalah suatu penambahan nilai yang terjadi pada sebuah variabel. Operator yang digunakan untuk melakukan increment adalah operator ++.Ada dua jenis increment di dalam C++ yaitu:

- Pre-increment

Pre-increment adalah operasi penambahan satu pada nilai variabel sebelum nilai variabel tersebut digunakan dalam ekspresi. Misalnya, jika memiliki variabel x dan kita menggunakan pre-increment, operasinya adalah ++x, yang artinya nilai x akan ditambah satu sebelum nilai x tersebut digunakan dalam ekspresi.

- Post-increment

Post-increment adalah menaikkan nilai yang terdapat pada sebuah variabel setelah nilai dari variabel tersebut diproses di dalam program. Pada post-increment operator ++ ditulis setelah variabel atau nilai yang akan dinaikkan.

Decrement merupakan kebalikan dari proses

increment, yaitu menurunkan (mengurangi) nilai dari suatu variabel. Operator yang digunakan untuk melakukan decrement adalah operator —.Ada dua jenis decrement dalam C++ yaitu:

- Pre-decrement

Pre-decrement adalah operasi di mana nilai variabel dikurangi satu sebelum nilai tersebut digunakan dalam ekspresi lain. Ini berarti nilai variabel dikurangi sebelum evaluasi ekspresi yang mengandung variabel tersebut.

- Post-decrement

Post-decrement adalah operasi di mana nilai variabel dikurangi satu setelah nilai tersebut digunakan dalam ekspresi lain.Nilai variabel tetap digunakan dalam ekspresi, kemudian dikurangi satu setelah ekspresi dievaluasi.

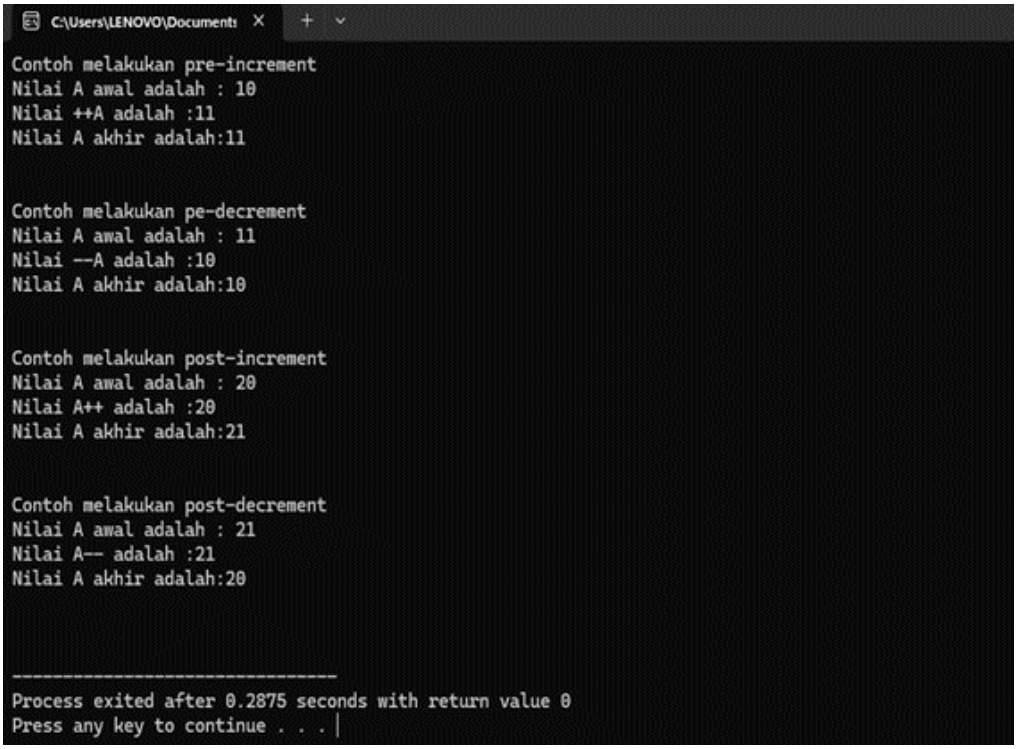
Contoh pemrograman dari increment dan decrement:

IncrementdanDecrement.cpp

```cpp
#include <iostream>
using namespace std;

int main()
{
    int A; //mendeklarasikan variabel
    A = 10;
    cout<<"Contoh melakukan pre-increment \n";
    cout<<"Nilai A awal adalah : "<<A<<endl;
    cout<<"Nilai ++A adalah :"<<++A<<endl;
    cout<<"Nilai A akhir adalah:"<<A<<endl;
    cout<<'\n'<<endl;

    cout<<"Contoh melakukan pe-decrement \n";
    cout<<"Nilai A awal adalah : "<<A<<endl;
    cout<<"Nilai --A adalah :"<<--A<<endl;
    cout<<"Nilai A akhir adalah:"<<A<<endl;
    cout<<'\n'<<endl;

    //mengubah nilai A menjadi 20
    A = 20;
    cout<<"Contoh melakukan post-increment \n";
    cout<<"Nilai A awal adalah : "<<A<<endl;
    cout<<"Nilai A++ adalah :"<<A++<<endl;
    cout<<"Nilai A akhir adalah:"<<A<<endl;
    cout<<'\n'<<endl;

    cout<<"Contoh melakukan post-decrement \n";
    cout<<"Nilai A awal adalah : "<<A<<endl;
    cout<<"Nilai A-- adalah :"<<A--<<endl;
    cout<<"Nilai A akhir adalah:"<<A<<endl;
    cout<<'\n'<<endl;


}
```

Gambar 3.20 penulisan increment dan decrement

Hasil dari contoh pemrograman diatas :



Gambar 3.21 hasil penulisan increment dan decrement

3. OPERATOR ARITMATIKA DAN HIERARKI OPERATOR ARITMATIKA

Operator aritmatika dalah operator yang digunakan untuk melakukan operasi-operasi aritmatika seperti perkalian, pembagian, penjumlahan, dan pengurangan.Ada beberapa jenis operator aritmatika yang bisa kita gunakan,yaitu meliputi:

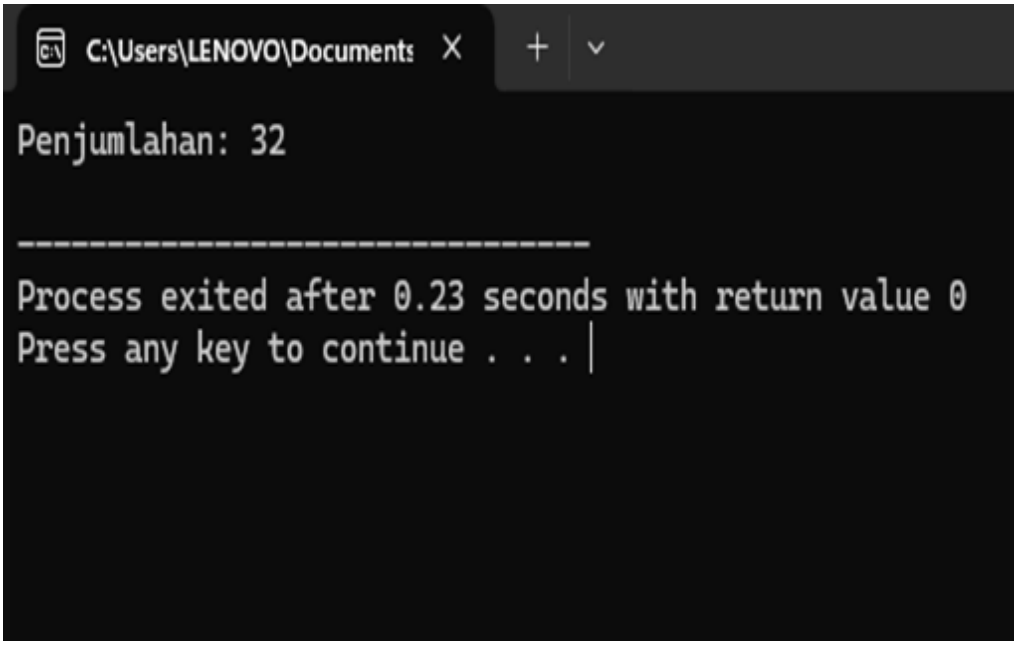
1. Penjumlahan(+)

Digunakan untuk menambahkan dua nilai.

Contoh pemrograman operator aritmatika penjumlahan:

```
#include<iostream>
using namespace std;

int main(){
	int a = 20;
	int b = 12;
	int hasil;

	// penjumlahan
	hasil = a + b;
	cout << “Penjumlahan: ” << hasil << endl;

	return 0;
}
```

Hasil dari pemrograman diatas:



Gambar 3.22 hasil penggunaan aritmatika penjumlahan

2.Pengurangan(-)

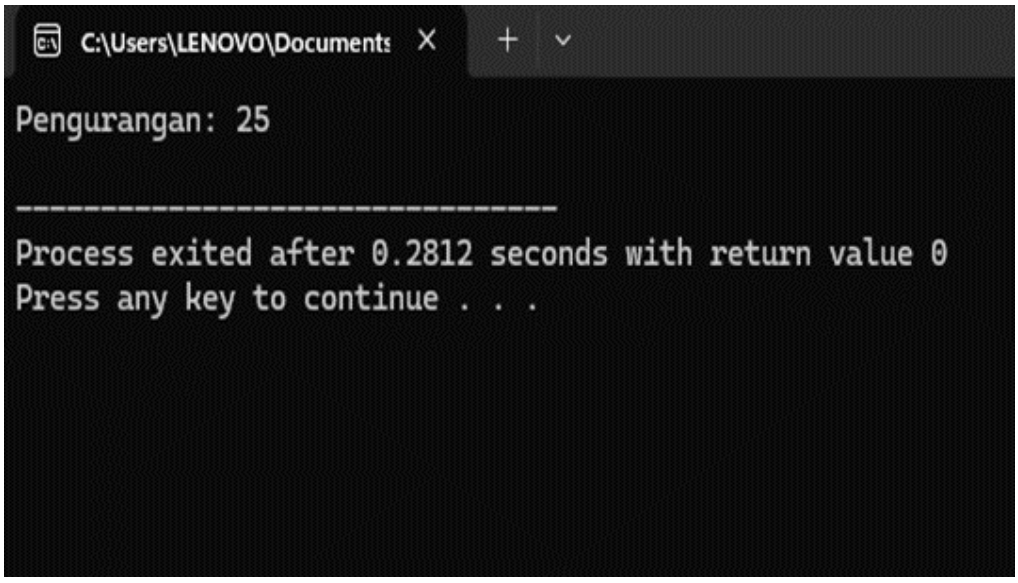
Digunakan untuk mengurangkan satu nilai dengan nilai lain.

Contoh pemrograman operator aritmatika pengurangan:

```
#include<iostream>
using namespace std;

int main(){
    int a = 50;
    int b = 25;
```

```
    int hasil;

    // Pengurangan
    hasil = a - b;
    cout << “Pengurangan: ” << hasil << endl;

    return 0;
}
```

Hasil dari pemrograman diatas:



Gambar 3.23 hasil penggunaan aritmatika pengurangan

3.Perkalian(*)

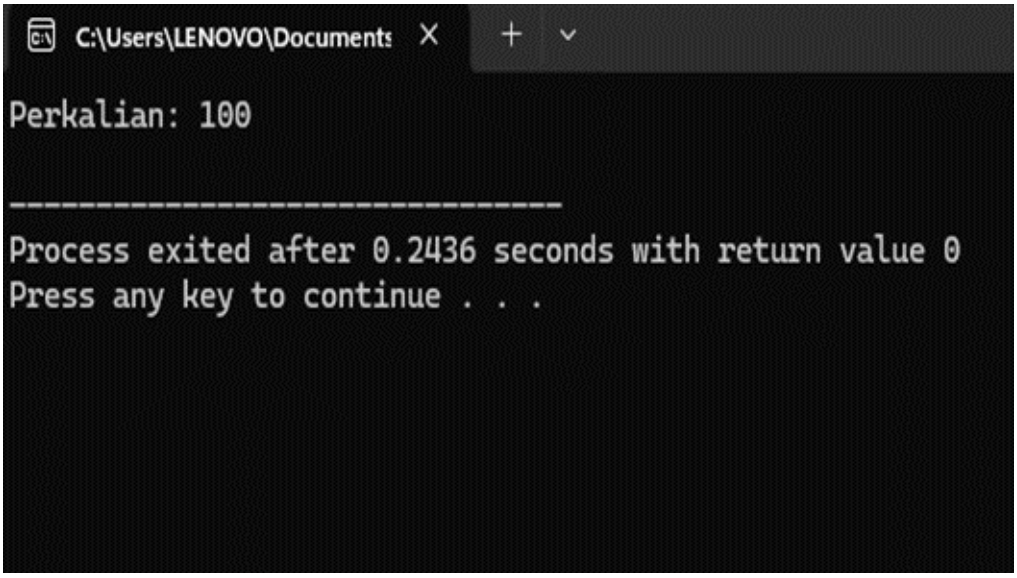
Digunakan untuk mengalikan dua nilai.

Contoh pemrograman operator aritmatika

perkalian:

```
#include<iostream>
using namespace std;

int main(){
    int a = 25;
    int b = 4;
    int hasil;

    // Perkalian
    hasil = a * b;
    cout << “Perkalian: ” << hasil << endl;

    return 0;
}
```

Hasil dari pemrograman diatas:



Gambar 3.24 hasil penggunaan aritmatika perkalian

4.Pembagian(/)

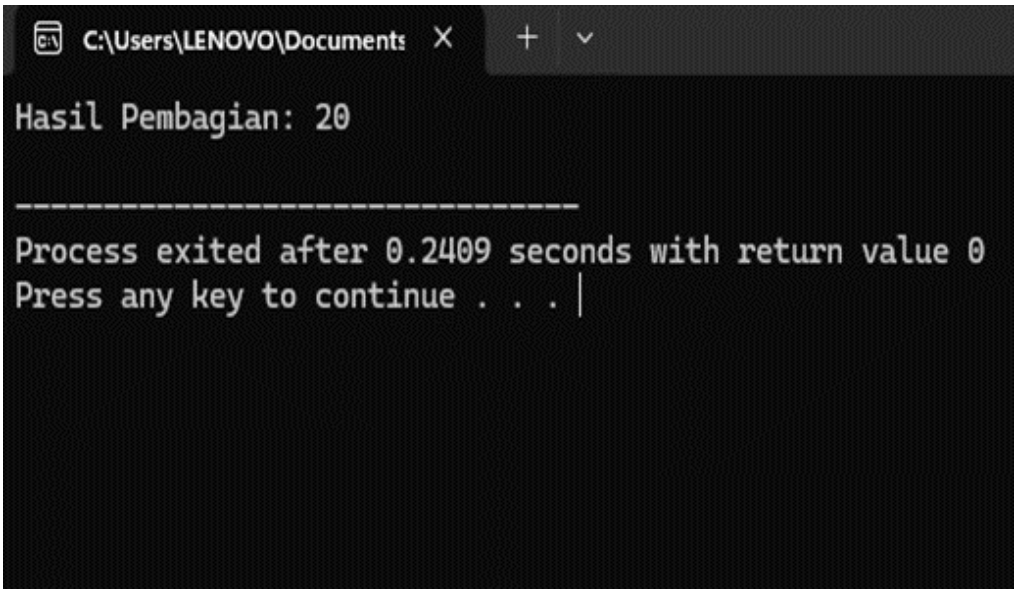
Digunakan untuk membagi nilai satu dengan nilai nilai yang lain.

Contoh pemrograman operator aritmatika pembagian:

```
#include<iostream>
using namespace std;

int main(){
    int a = 80;
    int b = 4;
    int hasil;
```

```
        // Pembagian
        hasil = a / b;
            cout << “Hasil Pembagian: ” << hasil << endl;

        return 0;
    }
```

Hasil dari pemrograman diatas:



Gambar 3.25 hasil penggunaan aritmatika pembagian

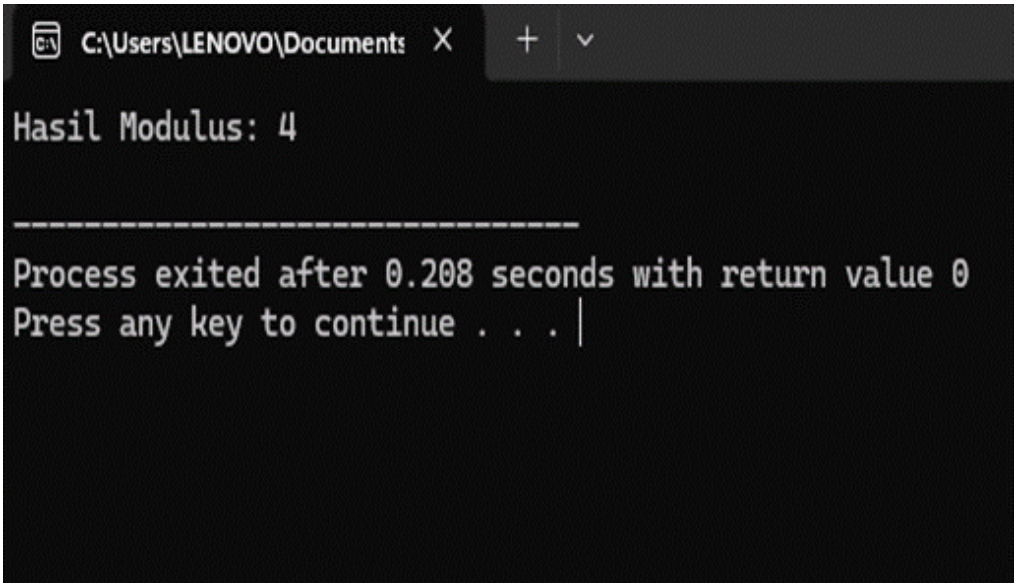
5.Modulus(%)

Digunakan untuk mendapatkan sisa dari pembagian dua nilai.

Contoh pemrograman operator aritmatika modulus:

```
#include<iostream>
using namespace std;

int main(){
    int a = 19;
    int b = 5;
    int hasil;

    // Modulus
    hasil = a % b;
    cout << “Hasil Modulus: ” << hasil << endl;

    return 0;
}
```

Hasil dari pemrograman diatas:



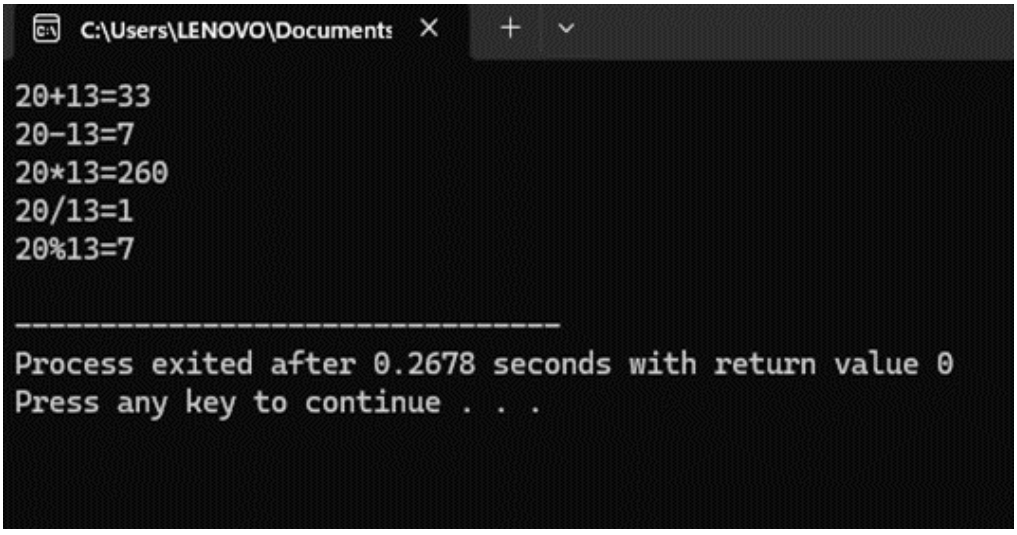
Gambar 3.26 hasil penggunaan aritmatika modulus

Contoh lain dari pemrograman operator aritmatika penjumlahan, pengurangan, perkalian, pembagian, dan modulus:

```
#include<iostream>
using namespace std;

int main()
{
    int A = 20, B = 13;
    int jumlah, kurang, kali, bagi, modulus;
    jumlah = A + B;
    kurang = A - B;
    kali = A * B;
```

```
    bagi = A / B;
    modulus = A % B;

    cout<<A<<"+"<<B<<"="<<jumlah<<endl;
    cout<<A<<"-"<<B<<"="<<kurang<<endl;
    cout<<A<<"*"<<B<<"="<<kali<<endl;
    cout<<A<<"/"<<B<<"="<<bagi<<endl;
    cout<<A<<"%"<<B<<"="<<modulus<<endl;

    return 0;
}
```

Hasil dari pemrograman operator aritmatika diatas:



Gambar 3.27 hasil penggunaan aritmatika

- HIERARKI OPERATOR ARITMATIKA

Di dalam suatu ekspresi aritmatika selalu menjupai beberapa operator aritmatika yang berbeda dan dapat digunakan secara bersamaaan.Urutan operator aritmatika yaitu sebagai berikut:

- Operator : * atau /

Keterangan : tingkatan operator sama,penggunaanya tergantung letak,yang di depan didahulukan.

- Operator : %

Keterangan : sisa pembagian

- Operator : + atau –

Keterangan : tingkatan operator sama,penggunaanya tergantung letak,yang didepan didahulukan.

Jika ada beberapa operator dalam ekspresi,operator yang memiliki hierarki lebih tinggi akan dieksekusi terlebih dahulu.Jika terdapat beberapa operator dengan hierarki yang sama,eksekusi dilakukan dari arah kiri ke kanan.

Contoh penggunaan:

A = 5 + 4 * 2 / 8

Langkah perhitunganya:

1. Yang dihitung adalah 4 * 2 (=8)

2.Yang dihitung adalah 8 / 8 (=1)

3.Yang dihitung adalah 5 + 1 = 6

- Operator Penugasan Gabungan

Operator To	Contoh	Setara
+ =	X + = 5;	X = X + 5;
-=	X - = 5;	X = X – 5;
* =	X * = 5;	X = X * 5;
/=	X / = 5;	X = X / 5;
%=	X % = 5;	X = X % 5;

Gambar 3.28 tabel hirearki operator aritmatika

4.RELATIONAL OPERATIONS DAN LOGICAL OPERATORS

Operator relasional biner menentukan hubungan berikut:

- Kurang dari (<)
- Lebih besar dari (>)
- Tidak sama dengan (!=)
- Kurang dari atau sama dengan (<=)

- Lebih besar dari atau sama dengan (>=)

Operator relational memiliki asokiativitas kiri-ke-kanan.Kedua operan operator relational harus dari jenis aritmatika atau pointer.Mereka menghasilkan jenis bool. Nilai yang dikembalikan yaitu false (0) jika hubungan dalam ekspresi salah, jika tidak, nilai yang dikembalikan yaitu true (1).

Operator	Meaning	Sample Expression	Evaluates to
= =	Equal	5 = = 5 5 = = 8	True False
! =	Not equal to	5 ! = 8 5 ! = 5	True False
>	Greater than	8 > 5 5 > 8	True False
<	Less than	5 < 8 8 < 5	True False

Operator	Meaning	Sample Expression	Evaluates to
= =	Greater than or equal	8 > = 5 5 > = 8	True False
< =	Less than or equal to	5 < = 8 8 < = 5	True False

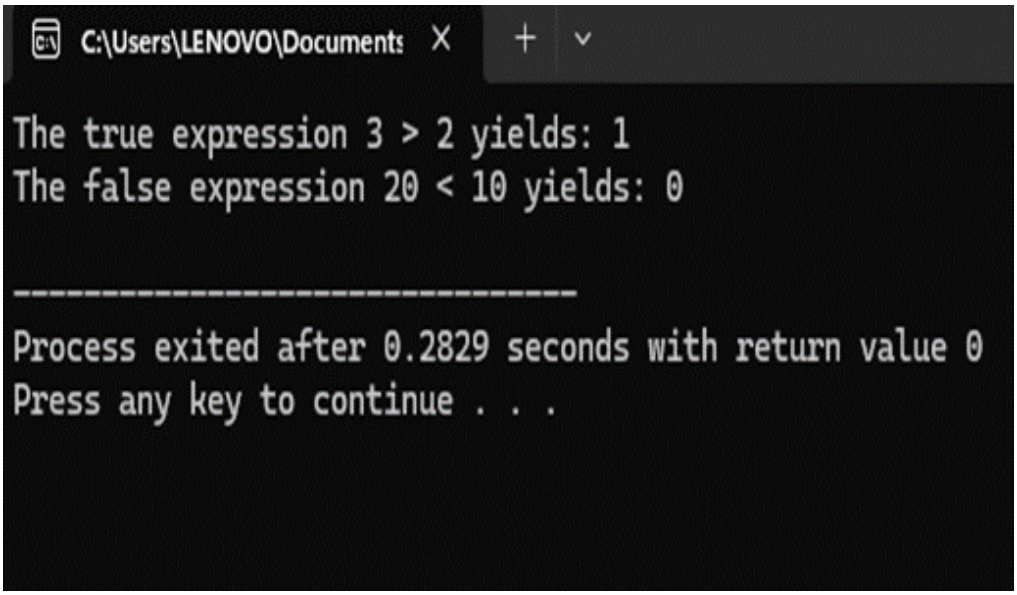
Gambar 3.29 tabel operator relasional biner

Contoh pemrograman operator relational:

```
// expre_Relational_Operators.cpp
// compile with: /EHsc
#include <iostream>
```

```
using namespace std;

int main() {
  cout  << “The true expression 3 > 2 yields: “
       << (3 > 2) << endl
       << “The false expression 20 < 10 yields: “
       << (20 < 10) << endl;
}
```

Hasil dari contoh pemrograman diatas:



Gambar 3.30 hasil program operator relational

- LOGICAL OPERATORS

Sama seperti pada matematika dan statistika, Operator Logika juga dibutuhkan dalam bahasa pemrograman. Operator logika atau Penghubung

logis adalah konstanta yang digunakan untuk menghubungkan atau menambahkan dua atau lebih rumus atau variabel. Mereka digunakan untuk mengetahui Logika antara nilai atau variabel.

Meskipun Operator Relasional digunakan untuk menguji apakah suatu kondisi tertentu benar atau salah, mereka hanya menguji satu kondisi pada satu waktu. Ada saatnya perlu menguji lebih dari satu kondisi dalam satu waktu. Situasi ini ditangani oleh Operator Logis. Dalam ilmu komputer, Operator Logis memberi kita fungsionalitas untuk menguji beberapa kondisi sekaligus.

Penggunaan – Dalam bahasa pemrograman, Operator Logis terutama digunakan dalam pernyataan kondisional dan loop untuk mengevaluasi kondisi. Operator logika digunakan untuk memeriksa apakah suatu ekspresi benar atau salah.

Setelah menggunakan Operator Logis pada variabel, hasil yang dikembalikan yaitu nilai boolean yaitu nilai Benar atau Salah. Secara default, nilai operan diubah menjadi tipe Boolean lalu dibandingkan dan dihitung.

Operator	Description	Example
&&	AND operator	(A && B) is false
||	OR operator	(A || B) is true
!	NOT operator	(A && B) is true

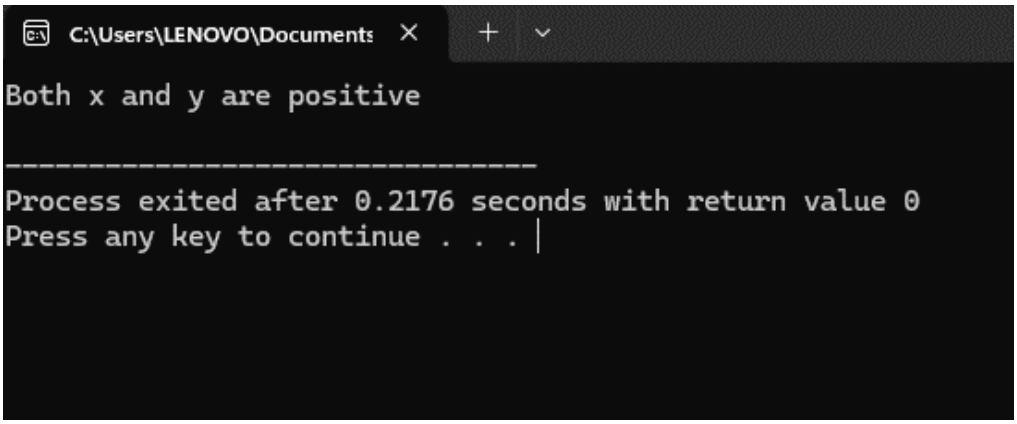
Gambar 3.31 tabel logical operator

Contoh pemrograman diatas :

1. AND (&&): menggunakan operator && untuk mengecek dua kondisi yang harus benar.

```
#include <iostream>
  using namespace std;


  int main() {
     int x = 10;
     int y = 20;
     if (x > 0 && y > 0) {
        cout << “Both x and y are positive” << endl;
     }
     return 0;
  }
```

Hasil pemrograman diatas:

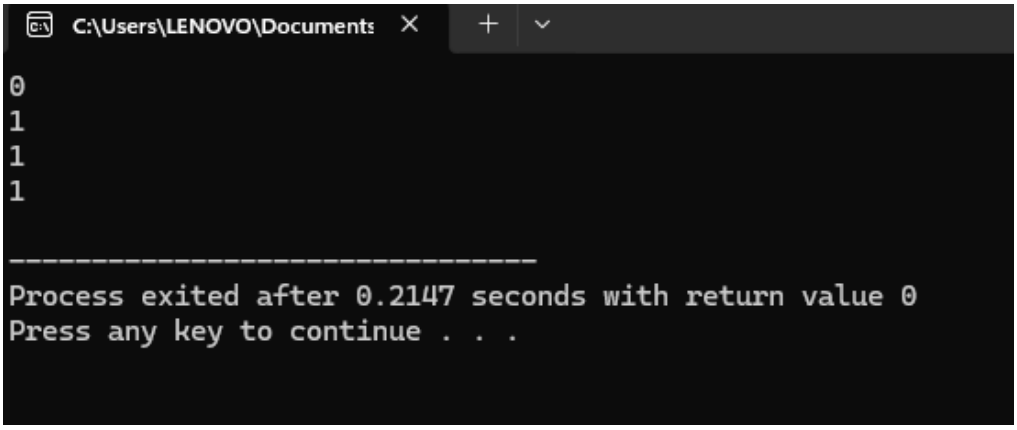


Gambar 3.32 hasil program logical operator AND

2.OR (||): Menggunakan operator || untuk mengecek salah satu dari dua kondisi yang benar.

```
#include <iostream>
using namespace std;

int main() {
   int a = 5;
   int b = 9;
   // false && false = false
   cout << ((a == 0) || (a > b)) << endl;
   // false && true = true
   cout << ((a == 0) || (a < b)) << endl;

   // true && false = true
   cout << ((a == 5) || (a > b)) << endl;

   // true && true = true
   cout << ((a == 5) || (a < b)) << endl;

   return 0;
}
```

Hasil pemrograman diatas:

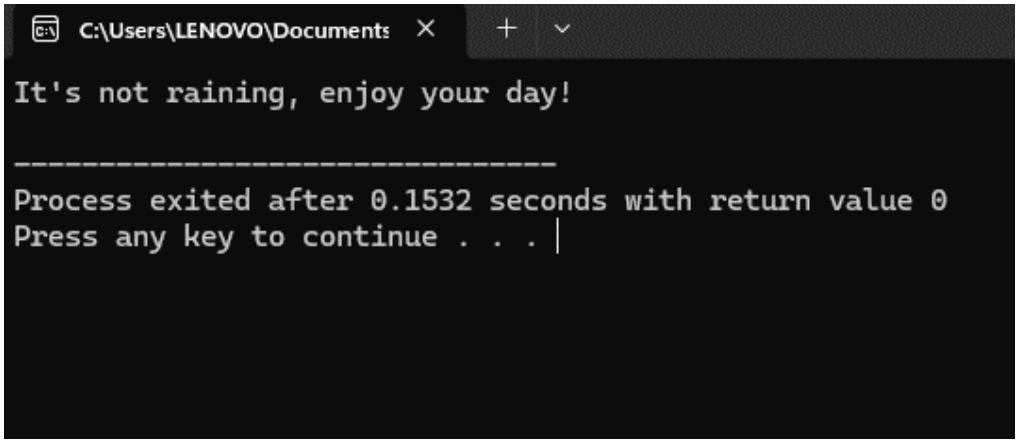


Gambar 3.33 hasil program logical operator OR

3.NOT (!): Menggunakan operator ! untuk membalik nilai kebenaran suatu kondisi.

```
#include <iostream>
  using namespace std;

  int main() {
     bool isRainy = false;
     if (!isRainy) {
        cout << “It’s not raining, enjoy your day!” << endl;
     }
     return 0;
  }
```

Hasil dari pemrograman diatas:



Gambar 3.34 hasil program logical operator NOT

4
Dynamic Response

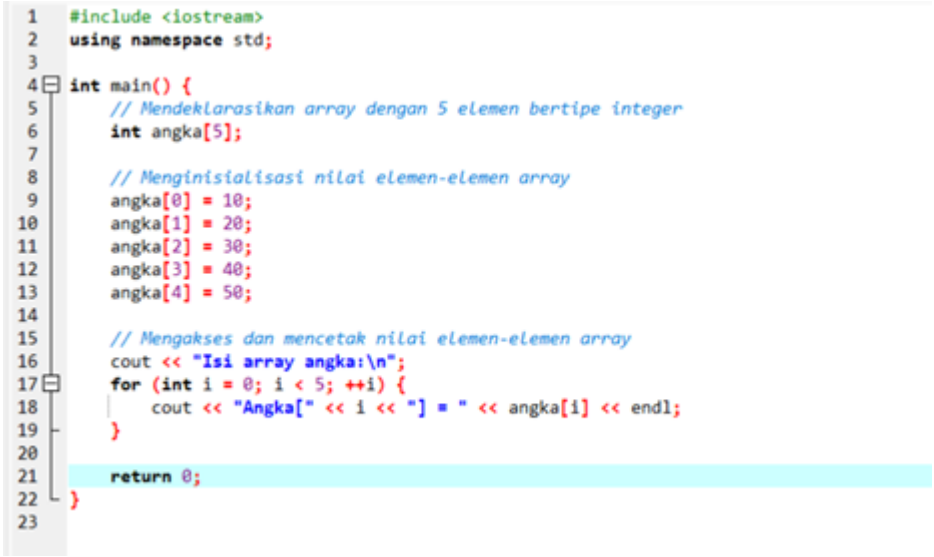
DYNAMIC RESPONSES

Dalam chapter ini akan dibahas berbagai bentuk dynamic responses hingga decision making. Di chapter empat ini akan dibahas dua topik utama, yakni Array dan Decision Making. Pada dasarnya kedua hal ini saling berhubungan sehingga program yang dibuat semakin lebih efektif serta efisien. Berikut akan dijelaskan bagaimana array dan decision making digunakan dalam bahasa pemrograman C++.

Array

Ketika Anda membuat file lalu menyimpannya, secara otomatis perangkat Anda akan menampilkan file terbaru yang telah Anda simpan. Sebagai contoh, Alfon dan teman-temannya sedang mengetik tugas proposal ATGW di Google Docs. Alfon telah mengetik sebanyak lima halaman. Akan tetapi salah satu temannya merevisi serta menyimpan hasil pekerjaan Alfon. Dengan demikian perangkat Alfon beserta teman-temannya secara otomatis akan menampilkan Dokumen terbaru hasil revisi yang telah disimpan oleh temannya tersebut. Analogi ini memberikan

gambaran bagaimana perangkat Anda bekerja. Perangkat Anda akan menyimpan file perubahan yang baru yang disimpan. Array merupakan sekumpulan data yang digunakan untuk menyimpan nilai dalam satu variabel.



```cpp
#include <iostream>
using namespace std;

int main() {
    // Mendeklarasikan array dengan 5 elemen bertipe integer
    int angka[5];

    // Menginisialisasi nilai elemen-elemen array
    angka[0] = 10;
    angka[1] = 20;
    angka[2] = 30;
    angka[3] = 40;
    angka[4] = 50;

    // Mengakses dan mencetak nilai elemen-elemen array
    cout << "Isi array angka:\n";
    for (int i = 0; i < 5; ++i) {
        cout << "Angka[" << i << "] = " << angka[i] << endl;
    }

    return 0;
}
```

Gambar 4.1 cara penulisan array

Dalam bahasa pemrograman C++ penggunaan array dapat dibagi menjadi tiga, antara lain pengolahan data, algoritma dan program yang efisien. Pengolahan data dalam array digunakan untuk menyimpan serta memanipulasi data yang telah diperoleh. Data-data tersebut dapat berupa matriks, statistik, ataupun daftar. Selain itu array memiliki penggunaan sebagai algoritma. Array dapat menjadi basis dalam algoritma pencarian dan pengurutan. Array juga dapat meningkatkan efisiensi program. Anda dapat menggunakan array untuk meningkatkan efisiensi program dalam melakukan operasi tertentu.

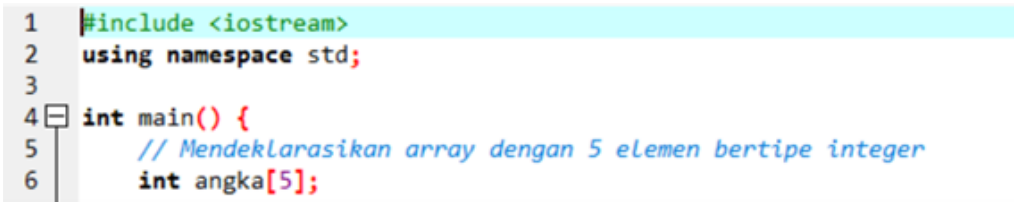
Pembuat program harus memahami kapan Array ini digunakan. Kekuatan logika pembuat program sungguh diuji. Pengulangan dan latihan terus menerus menjadi kunci agar logika berpikir pembuat program semakin tajam. Harapannya, beberapa langkah panduan pembuatan array di bawah ini menjadi pegangan bagi pembuat program agar program yang dihasilkan terus berkembang.

Langkah-langkah pembuatan Array

Pembuatan array dalam pemrograman C++ melibatkan beberapa langkah dasar yang telah Anda pelajari di chapter-chapter sebelumnya. Chapter-chapter sebelumnya menjadi dasar bagi Anda untuk memahami chapter keempat ini. Panduan yang akan diberikan merupakan panduan umum yang dibuat sesederhana mungkin sehingga Anda dapat memahaminya dengan baik. Terdapat lima langkah utama dalam pembuatan array, antara lain ialah deklarasi array, inisialisasi array, akses elemen array, penggunaan ukuran array, dan Looping dalam array.

1. Deklarasi array

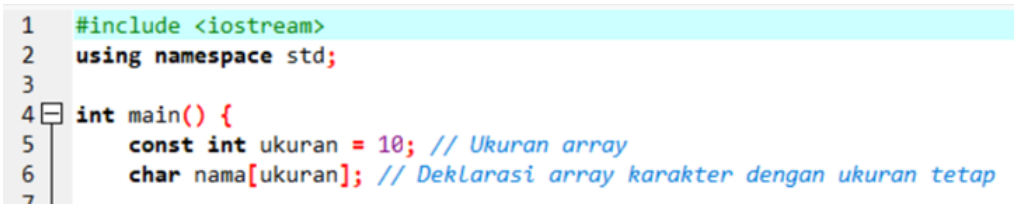
Langkah pertama yang Anda harus buat adalah mendeklarasikan array. Anda perlu mengidentifikasi serta menentukan tipe data serta jumlah elemen-elemen yang akan dimasukkan. Anda dapat memasukkan tipe data integer (‘int’), karakter (‘char’), boolean (‘bool’), string (‘string’), dan lain sebagainya.



```
#include <iostream>
using namespace std;

int main() {
    // Mendeklarasikan array dengan 5 elemen bertipe integer
    int angka[5];
```

Gambar 4.2 penulisan deklarasi array

Gambar di atas merupakan salah satu bagian deklarasi dari array. Tipe data yang digunakan dalam deklarasi di atas adalah integer (‘int’). Tipe data ini digunakan untuk menyimpan nilai bilangan bulat. int angka berarti variabel yang digunakan oleh pembuat program adalah angka dan [5] berarti jumlah elemen.



```
#include <iostream>
using namespace std;

int main() {
    const int ukuran = 10; // Ukuran array
    char nama[ukuran]; // Deklarasi array karakter dengan ukuran tetap
```

Gambar 4.3 penulisan deklarasi array

Gambar di atas merupakan deklarasi array dari tipe data karakter. Tipe data yang digunakan dalam deklarasi di atas adalah (‘char’). Tipe data ini digunakan untuk menyimpan karakter dan dengan jumlah karakter 10. Variabel yang digunakan adalah ukuran.

2. Inisialisasi *array*

```
8       // Menginisialisasi nilai elemen-elemen array
9       angka[0] = 10;
10      angka[1] = 20;
11      angka[2] = 30;
12      angka[3] = 40;
13      angka[4] = 50;
```

Gambar 4.4 penulisan inisialisasi array

Insialisasi array adalah proses memberikan nilai awal kepada setiap elemen dalam array. Gambar di atas merupakan salah satu contoh insialisasi nilai elemen-elemen array. Dari inisialisasi nilai elemen-elemen array di atas, kita semua dapat mengetahui bahwa jumlah elemen yang dibuat adalah lima. Jumlah elemen yang diberikan harus sama dengan ukuran array yang dideklarasikan. Jika jumlah nilai kurang dari ukuran array, elemen yang tersisa akan secara otomatis terisi dengan nilai default sesuai dengan tipe data yang dideklarasikan.

3. Akses elemen array

Akses elemen array mengacu pada cara mendapatkan nilai dari elemen-elemen individu dalam array. Dalam C++, kita menggunakan indeks array untuk merujuk ke elemen tertentu dalam array tersebut. Indeks dimulai dari 0 untuk elemen pertama dan berlanjut hingga (ukuran array - 1) untuk elemen terakhir.

Berikut adalah cara mengakses elemen array dalam C++:

- Menggunakan Indeks: Indeks array digunakan

dengan menempatkannya di dalam tanda kurung siku [] setelah nama array. Misalnya, nama_array[indeks].

- Menggunakan Indeks untuk Membaca Nilai: Untuk membaca nilai dari elemen array, gunakan indeks array yang sesuai. Misalnya, int nilai = nama_array[indeks]; akan menyimpan nilai elemen ke-indeks dari nama_array ke dalam variabel nilai.

- Menggunakan Indeks untuk Menetapkan Nilai: Untuk menetapkan nilai ke elemen array, gunakan indeks array yang sesuai di sebelah kiri dari operator penugasan (=). Misalnya, nama_array[indeks] = nilai; akan menetapkan nilai ke elemen ke-indeks dari nama_array.

- Penggunaan Loop: Loop sering digunakan untuk mengakses dan memanipulasi elemen-elemen array secara berurutan atau berdasarkan kondisi tertentu.

Contoh penggunaan akses elemen array dalam C++:

```cpp
#include <iostream>

using namespace std;

int main() {
    int angka[5] = {10, 20, 30, 40, 50};

    // Membaca nilai dari elemen array dan mencetaknya
    cout << "Elemen-elemen array angka:\n";
    for (int i = 0; i < 5; ++i) {
        cout << "Angka[" << i << "] = " << angka[i] << endl;
    }

    // Menetapkan nilai baru ke elemen array dan mencetaknya
    angka[2] = 35;
    cout << "\nSetelah menetapkan nilai baru untuk Angka[2]:\n";
    for (int i = 0; i < 5; ++i) {
        cout << "Angka[" << i << "] = " << angka[i] << endl;
    }

    return 0;
}
```

Gambar 4.5 penulisan akses elemen array

4. Penggunaan ukuran array

Ukuran array adalah jumlah elemen yang ditampung oleh array tersebut. Ukuran array dibuat oleh pembuat program pada saat deklarasi. Ukuran array disesuaikan dengan jumlah elemen yang akan disimpan.

5. Looping dalam array

Looping dalam array digunakan untuk mengakses setiap elemen array secara berurutan.

```cpp
    // Mengakses dan mencetak nilai elemen-elemen array
    cout << "Isi array angka:\n";
    for (int i = 0; i < 5; ++i) {
        cout << "Angka[" << i << "] = " << angka[i] << endl;
    }
```

Gambar 4.6 penulisan looping dalam array

Gambar di atas merupakan contoh dari loop for yang digunakan untuk looping setiap elemen dalam array. Contoh di atas menggunakan variabel penghitung, yakni ‘i’. Anda bisa menggunakan cara looping yang lainnya sesuai dengan kebutuhan Anda.

ARRAY DUA DIMENSI

Sebelumnya telah dibahas pengertian, konsep dan penerapan sederhana Array. Array yang digunakan dalam dalam sub bab sebelumnya merupakan array satu dimensi. Di sub bab yang selanjutnya ini akan dibahas array dua dimensi meliputi pengertian, konsep penerapan serta kelebihan serta kekurangannya.

Pengertian dan Konsep dasar Array Dua Dimensi

Array dua dimensi merupakan perkembangan dari array satu dimensi. Dalam array dua dimensi, setiap elemen memiliki dua indeks. Indeks yang pertama untuk baris, dan yang satunya untuk kolom. Hal ini membantu serta memungkinkan Anda untuk menampilkan data secara terstruktur seperti tabel, kolom maupun bentuk matriks. Ketika Anda memperoleh data yang cukup beragam, array dua dimensi ini sangat cocok untuk Anda gunakan agar pengolahan serta tampilan data dapat dibuat secara jelas dan terstruktur. Dengan demikian Anda perlu memiliki lebih dari satu data yang berikutnya akan diolah. Array dua dimensi ini juga bisa terus dikembangkan menjadi array tiga dimensi dan lain sebagainya.

Array dua dimensi mengorganisir data menjadi berbagai baris dan berbagai kolom sehingga memungkinkan merepresentasikan indeks lebih efisien. Selain itu array dua dimensi pula memungkinkan data lebih terbentuk secara terstruktur. Ada beberapa konsep dalam array dua dimensi ini, antara lain struktur matriks, akses elemen, dan inisialisasi. Array dua dimensi juga tidak terlepas dengan operasi matematika di dalamnya.

Berikut merupakan beberapa konsep dasar array dua dimensi:

1. Struktur matriks

Array dua dimensi dapat Anda bayangkan sebagai array di dalam array. Artinya, array yang digunakan bersifat multidimensi. Terdapat dua indeks untuk baris serta kolom. Sebagai contoh, jika Anda memiliki array ‘A’ dengan ‘m’ sebagai baris dan ‘n’ sebagai kolom, Anda dapat membuatnya dengan matriks ‘m x n.’

2. Akses elemen

Untuk mengakses elemen dalam array dua dimensi, Anda membutuhkan dua indeks, yakni indeks untuk baris dan kolom.

3. Inisialisasi

Seperti yang sudah dibahas dalam sub bab sebelumnya, inisialisasi array adaah proses memberikan nilai awal kepada setiap elemen dalam array. Penting untuk diingat, bahwa jumlah elemen yang diberikan harus sama dengan ukuran array yang dideklarasikan. Sebagai contoh matriks 3 x 3

A = {

[1, 2, 3]

[4, 5 ,6]

[7, 8, 9]

}

Di sini, Anda memiliki matriks 3x3 dengan nilai-nilai yang telah ditetapkan.

Penggunaan Array Dua Dimensi dalam Pemrograman

Setelah memahami pengertian dan konsep dasar array dua dimensi, kini Anda perlu mengaplikasikannya secara langsung. Seperti yang kita semua ketahui bahwa array dua dimensi dalam pemrograman C++ digunakan untuk menyimpan data dalam bentuk tabel dua dimensi. Contoh dari bentuk-bentuk tabel dua dimensi yakni, matriks, tabel berbagai data, dan lain sebagainya.

Ada beberapa poin penting dalam penggunaan array dua dimensi, yakin pendeklarasian, inisialisasi, akses dan fungsi. Dalam penggunaanya, Anda perlu mendeklarasikan Array Dua Dimensi kedalam program C++ yang dibuat. Selanjutnya Anda perlu inisialisasi awal pada saat pendeklarasian. Langkah berikutnya ialah akses manipulasi data dan penggunaan array dua dimensi di dalam fungsi. Berikut merupakan beberapa penggunaan array dua dimensi dalam pemrograman:

1. Mendeklarasikan Array Dua Dimensi

Seperti yang sudah dipelajari di sub bab sebelumnya bahwa langkah pertama dalam penerapan array adalah dengan pendeklarasian. Penting untuk Anda ingat langkah-langkah deklarasi array dua dimensi, antara lain: Menetukan Tipe data, Menetukan Dimensi Array, dan inisialisasi. Berikut langkah teknis pendeklarasian array dua dimensi:

a. Menentukan Tipe Data

Langkah pertama adalah menentukan tipe data yang ingin disimpan dalam array. Hal ini dapat dilakukan dengan menuliskan tipe data sebelum nama array. Contoh

```
1  int array2D[baris][kolom]; // array dua dimensi bertipe data integer
2  double nilai[5][3]; // array dua dimensi bertipe data double
```

Gambar 4.7 penulisan array dua dimensi

Kode di atas merupakan contoh penentuan tipe data. Di baris pertama diketahui memiliki tipe data integer sehingga Anda dapat menggunakan ‘int’. Sedangkan di baris kedua diketahui memiliki tipe data double sehingga Anda dapat menggunakan ‘double’ dalam aplikasi Dev C++.

b. Menentukan Dimensi Array

Langkah selanjutnya ialah menentukan jumlah baris serta kolom. Penulisan jumlah baris dan kolom dapat

dituliskan dengan tanda kurung siku ‘[]’ Jumlah yang dituliskan terlebih dahulu adalah jumlah baris lalu diikut dengan jumlah kolom. Penulisan keduanya dipisahkan oleh koma. Contoh:

Gambar 4.8

Dari contoh di atas ini kita semua mengetahui jumlah baris dan kolomnya. Di baris pertama diketahui memiliki tiga jumlah baris dan empat jumlah kolom. Sedangkan di baris kedua diketahui memiliki dua jumlah baris dan sepuluh jumlah kolom. Mengapa demikian? Sebab angka 3 dan 2 yang dituliskan di awal menunjukkan jumlah baris serta angka 2 dan 10 yang menunjukkan kolom ditulis di belakangnya.

c. Deklarasi lengkap

Deklarasi lengkap merupakan penggabungan langkah satu dan langkah kedua. Dengan demikian Anda mendapatkan deklarasi array dua dimensi yang lengkap.

2. Inisialisasi Awal Array Dua Dimensi

Inisialisasi awal array dua dimensi dalam C++ berarti memberikan nilai awal untuk elemen array saat deklarasi.

Inisialisasi Daftar Nilai . Dalam Inisialisasi ini Anda dapat menggunakan cara ini dengan menuliskan daftar nilai awal di dalam kurung siku ‘[]’ setelah deklarasi. Seperti materi sub bab selanjutnya, jumlah nilai harus sama dengan jumlah elemen dalam array.

Contohnya:

```
1   int data[3][4]; // array 2 dimensi dengan 3 baris dan 4 kolom
2   char huruf[2][10]; // array 2 dimensi dengan 2 baris dan 10 kolom
```

Gambar 4.8 penulisan array dua dimensi

int NilaiTugas[5][3] = { {70, 75, 80}, {60, 90, 68}, {95, 100, 92}, {82, 79, 89}, {55, 67, 71} };

Pada contoh inisialisasi di atas, dapat kita simpulkan bahwa jumlah baris 5 dan jumlah kolomnya 3. Contoh di atas juga menunjukkan jumlah nilai dengan jumlah elemen array telah sesuai.

Sebagai catatan bahwa inisialisasi awal array hanyalah opsional. Anda dapat menggunakan inisialisasi setelah deklarasi tipe data. Penting diingat pula bahwa jumlah nilai dalam inisialisasi harus sama dengan jumlah elemen dalam array.

3. Akses dan Manipulasi Data Array Dua Dimensi

Setelah mendeklarasikan dan menginisalisasi array dua dimensi, langkah selanjutnya, Anda dapat mengakses dan menginisialisasi array dua dimensi. Berikutnya Anda pula dapat memanipulasi data di dalamnya. Berikut merupakan akses elemen array dan manipulasi data dalam array.

a. Akses Elemen Array

Penomoran indeks dalam array dimulai dari angka 0 (nol). Anda dapat mengases elemen tertentu dalam array dua dimensi. Anda perlu menggunakan dua indeks yang dipisahkan dengan koma. Contoh:

```
6   int nilaiUjian[5][3];
7
8   // Mengakses elemen baris ke-2, kolom ke-1
9   int nilai = nilaiUjian[2][1];
10
11  // Mengubah nilai elemen baris ke-4, kolom ke-2
12  nilaiUjian[4][1] = 88;
```

Gambar 4.9 penulisan akses elemen array dua dimensi

b. Manipulasi Elemen Array

Manipulasi data elemen array dua dimensi dalam C++ mengacu pada tindakan mengubah, mengakses, atau mengolah data yang disimpan dalam array dua dimensi. Array dua dimensi adalah struktur data yang berguna untuk menyimpan data yang tersusun dalam baris dan kolom, laksana tabel atau matriks. Berikut adalah beberapa contoh manipulasi data elemen array dua dimensi:

1. Mengubah Nilai Elemen:

Mengubah Nilai Tunggal: Anda dapat mengubah nilai elemen tunggal dalam array dua dimensi dengan menggunakan dua indeks dan operator penugasan (=).

int nilaiUjian[5][3];

// Mengubah nilai elemen baris ke-2, kolom ke-1

menjadi 95

nilaiUjian[2][1] = 95;

2. Mengubah Nilai Berkelompok: Anda dapat mengubah nilai sekelompok elemen dalam array dua dimensi menggunakan perulangan.

// Mengubah nilai semua elemen pada kolom ke-2 menjadi 100

```
for (int i = 0; i < 5; i++) {
  nilaiUjian[i][1] = 100;
}
```

3. Mencari Nilai tertentu:

Anda dapat mencari nilai tertentu dalam array dua dimensi menggunakan perulangan dan pernyataan kondisional.

// Mencari nilai 90 dalam array dan menampilkan posisinya

```
bool ditemukan = false;
for (int i = 0; i < 5; i++) {
  for (int j = 0; j < 3; j++) {
    if (nilaiUjian[i][j] == 90) {
      cout << “Nilai 90 ditemukan di baris ” << i + 1 << “, kolom ” << j + 1 << endl;
```

```
      ditemukan = true;
      break;
    }
  }
  if (ditemukan) {
    break;
  }
}
```

4. Melakukan Perhitungan Matematika:

Anda dapat melakukan berbagai operasi matematika pada elemen array dua dimensi, seperti penjumlahan, pengurangan, perkalian, dan pembagian.

```
// Menghitung total nilai pada baris ke-3
int total = 0;
for (int j = 0; j < 3; j++) {
  total += nilaiUjian[2][j];
}
cout << “Total nilai pada baris ke-3: ” << total << endl;
```

5. Mengurutkan Elemen:

Anda dapat mengurutkan elemen array dua dimensi menggunakan algoritma pengurutan, seperti bubble sort atau selection sort

```
// Mengurutkan elemen pada kolom ke-1 (secara ascending)
for (int i = 0; i < 4; i++) {
  for (int j = 0; j < 5 - i - 1; j++) {
    if (nilaiUjian[j][0] > nilaiUjian[j + 1][0]) {
      swap(nilaiUjian[j][0], nilaiUjian[j + 1][0]);
    }
  }
}
```

6. Mencari Nilai Maksimum dan Minimum:

Anda dapat mencari nilai maksimum dan minimum dalam array dua dimensi menggunakan perulangan dan variabel pembanding

```
// Mencari nilai maksimum dan minimum dalam array
int nilaiMax = nilaiUjian[0][0];
int nilaiMin = nilaiUjian[0][0];

for (int i = 0; i < 5; i++) {
  for (int j = 0; j < 3; j++) {
    nilaiMax = max(nilaiMax, nilaiUjian[i][j]);
    nilaiMin = min(nilaiMin, nilaiUjian[i]
```

4. Fungsi Array Dua Dimensi dalam Pemrograman C++

Array dua dimensi dalam pemrograman C++ adalah struktur data yang sangat berguna untuk menyimpan data dalam bentuk tabel dengan baris dan kolom. Fungsi array dua dimensi ini sangat penting dalam pemrograman karena memungkinkan penggunaan matriks untuk merepresentasikan data dalam format yang lebih terstruktur. Berikut merupakan beberapa fungsinya:

1. Representasi Data Tabel

Array dua dimensi sangat berguna untuk merepresentasikan data dalam bentuk tabel. Misalnya, matriks dua dimensi dapat digunakan untuk menyimpan data penilaian siswa, data matriks, data piksel dalam gambar, dan banyak lagi.

Contoh:

```
6   int grades[3][4];  // Matriks 3x4 untuk menyimpan nilai siswa
```

Gambar 4.10 penulisan representasi data tabel

2. Pengolahan Matriks

Fungsi array dua dimensi sangat berguna untuk pengolahan matriks matematika, seperti penjumlahan, pengurangan, perkalian, dan operasi matriks lainnya.

Contoh:

```
// Penjumlahan dua matriks
int matrix1[2][2] = {{1, 2}, {3, 4}};
int matrix2[2][2] = {{5, 6}, {7, 8}};
int result[2][2];

for(int i = 0; i < 2; ++i) {
    for(int j = 0; j < 2; ++j) {
        result[i][j] = matrix1[i][j] + matrix2[i][j];
    }
}
```

Gambar 4.11 penulisan pengolahan matriks

3. Manipulasi Gambar dan Grafika

Dalam pemrograman grafika, array dua dimensi sering digunakan untuk merepresentasikan gambar atau citra. Setiap elemen array dapat merepresentasikan nilai piksel dari gambar.

Contoh:

```
int image[WIDTH][HEIGHT]; // Array untuk menyimpan citra dengan lebar dan tinggi tertentu

// Manipulasi piksel pada citra
image[x][y] = 255;  // Mengatur piksel pada koordinat (x, y) menjadi putih (nilai maksimum)
```

Gambar 4.12 penulisan manupulasi gambar dan grafika

4. Penanganan Peta atau Grid

Ketika Anda bekerja dengan peta atau grid dalam permainan atau aplikasi yang memerlukan pemetaan ruang, array dua dimensi dapat digunakan untuk merepresentasikan peta atau grid.

Contoh:

```
6   char map[10][10]; // Array 2D untuk mewakili peta 10x10
7
8   // Inisialisasi peta
9   for(int i = 0; i < 10; ++i) {
10      for(int j = 0; j < 10; ++j) {
11          map[i][j] = '.';
12      }
13  }
14
15  // Memasukkan karakter 'X' sebagai karakter pemain di koordinat (5, 5)
16  map[5][5] = 'X';
```

Gambar 4.13 penulisan penanganan peta atau grid

5. Pemrosesan Data Tabular

Saat bekerja dengan data tabular, seperti data dari spreadsheet, array dua dimensi memungkinkan penyimpanan dan pengolahan data yang terstruktur.

Contoh:

```
6   int data[ROWS][COLS]; // Matriks untuk menyimpan data tabular
7
8   // Mengisi data
9   for(int i = 0; i < ROWS; ++i) {
10      for(int j = 0; j < COLS; ++j) {
11          data[i][j] = getSomeData();
12      }
13  }
```

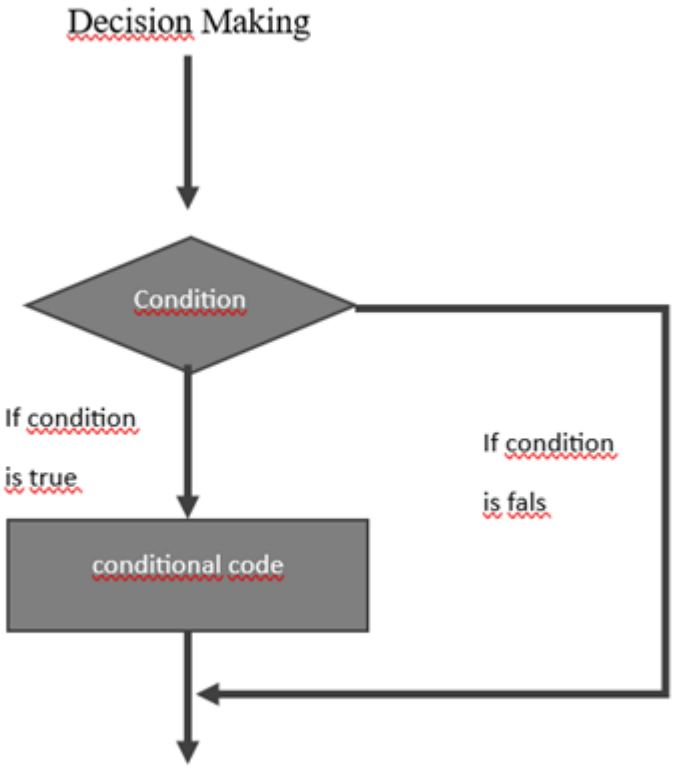
Gambar 4.14 penulisan pemrosesan data tabular

Dalam semua contoh di atas, array dua dimensi digunakan untuk menyimpan dan memanipulasi data dalam bentuk yang lebih terstruktur dan terorganisir. Hal ini membuatnya menjadi alat yang sangat berguna

dalam banyak aplikasi pemrograman yang melibatkan data dalam bentuk matriks atau tabel.

DECISION MAKING

Decision Making merupakan salah satu teknik di mana program mengambil keputusan dari kondisi-kondisi tertentu. Decision making merupakan suatu keadaan dalam dua nilai, yakni true (ya) atau false (tidak). Tujuan dari decision making ialah melakukan komparasi antara dua hal atau lebih sehingga selanjutnya bisa dilakukan eksekusi. Seorang pembuat program perlu memasukkan beberapa nilai ke dalam programnya agar decision making dapat berjalan. Dengan beberapa struktur fungsinya, nilai-nilai tersebut akan dibandingkan dan dieksekusi sesuai dengan ketentuan yang pembuat program buat. Komparasi yang dibuat tersebut akan menghasilkan keputusan benar atau salah, True or False.

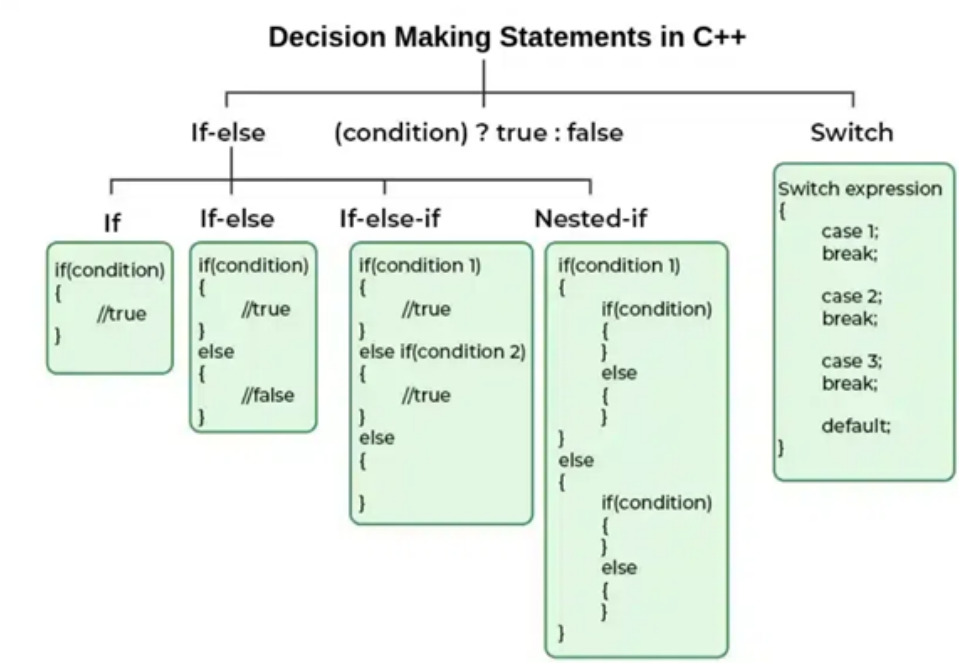


Gambar 4.15 gambar logika decision making

Gambar 4.15 menunjukkan flowchart dari decision making sederhana. Flowchart menggambarkan ketika suatu program diberikan beberapa kondisi atau nilai dengan conditional code makan program tersebut akan langsung mengeksekusi ya atau tidak. Eksekusi ya dan tidak dalam program dipengaruhi oleh conditional code yang ditentukan oleh pembuat program.

```
If (condition) {
                    //conditional code
} else{
                    //other code
}
```

Gambar 4.16 konsep penulisan decision making



Gambar 4.17 konsep penulisan decision making

Gambar 4.17 dapat menjadi panduan Anda dalam menentukan conditional statement. Conditional Statement tersebut perlu Anda sesuaikan dengan program yang ingin Anda buat. Conditional Statement merupakan salah satu struktur kontrol fundamental dalam bahasa pemrograman C++. Struktur ini memungkinkan programmer untuk membuat keputusan berdasarkan kondisi tertentu, sehingga program apat menjalankan kode yang berbeda berdasarkan hasil

evaluasi kondisi tersebut. Conditional statement sangat penting dalam pemrograman karena memungkinkan program untuk lebih fleksibel dan responsif terhadap berbagai situasi. Berikut merupakan beberapa jenis conditional statement, antara lain:

1. If

If adalah salah satu bentuk conditional statement. If merupakan conditional statement yang paling mendasar. Conditional statement ini menjalankan blok kode tertentu hanya jika suatu kondisi tertentu terpentuhi. Dalam kata lain Anda sebagai pembuat program dapat memerintahkan program Anda untuk segera mengeksekusi dan mengevaluasi satu kondisi jika kondisi tersebut benar.

if.cpp

```cpp
#include <iostream>

using namespace std;

int main() {
    int x = 10;

    if (x > 5) {
        cout << "Nilai x adalah " << x << ", dan itu lebih besar dari 5." << endl;
    }
    if (x <= 5) {
        cout << "Nilai x adalah " << x << ", dan itu tidak lebih besar dari 5." << endl;
    }

    return 0;
}
```

Gambar 4.18 penulisan if

Gambar di atas merupakan contoh sederhana dari penggunaan conditional statement ‘if’. Setiap statement ‘if’ ditulis terlebih dahulu sehingga program akan secara otomatis mengecek. Jika kondisi terpenuhi, kode yang ada di dalamnya akan langsung dieksekusi. Kode di atas menggunakan deklarasi variabel ‘x’ yang diisi dengan

nilai 10. Deklarasi variabel ‘x’ dengan nilai 10 ditulis setelah titik masuk setiap program C++, yakni main ditulis. Pada baris ini, sebuah variabel ‘x’ dideklarasikan sebagai integer. Selanjutnya, penulisan conditional statement ‘if’. Statement ini akan membuat program dengan sendirinya mengecek apakah nilai ‘x’ lebih besar dari 5. Hal ini ditunjukkan dengan nilai yang ada di dalam kurung ‘(x > 5). Apabila kondisi tersebut benar (nilai ‘x’ leibh besar dari 5), maka kode di dalam kurung kurawal ‘{}’ akan segera dieksekusi sesuai dengan program yang ingin Anda jalankan. Di dalam hal ini, program akan mencetak output “Nilai x adlaah [nilai x], dan itu lebih besar dari 5.” ke dalam layar menggunakan ‘cout’. Perlu diingat dari chapter-chapter sebelumnya bahwa fungsi-fungsi seperti cout, endl, cin akan tereksekusi apabila Anda menyertakan file header ‘iostream.’ Poin selanjutnya dari contoh program di atas ialah pernyataan ‘if’ yang kedua. Kondisi yang akan dievaluasi di sini adalah apakah nilan ‘x’ kurang dari atau sama dengan 5. Evaluasi ini ditunjukkan dengan (x <= 5). Namun, karena nilai ‘x’ adalah 10 (lebih besar dari 5), maka blok kode ini tidak akan tereksekusi. Program Anda dengan sendiri akan mengecek, mengevaluasi, dan mengeksekusi nilai yang sesuai dengan ketentuan dalam conditional statement.

if.cpp

```cpp
#include <iostream>

using namespace std;

int main() {
    double temperature;

    // Meminta pengguna untuk memasukkan suhu dalam Celsius
    cout << "Masukkan suhu dalam Celsius: ";
    cin >> temperature;

    // Menampilkan pesan sesuai dengan suhu yang dimasukkan
    if (temperature > 25) {
        cout << "Saat ini cuaca sedang panas." << endl;
    }
    if (temperature >= 10 && temperature <= 25) {
        cout << "Saat ini cuaca sedang nyaman." << endl;
    }
    if (temperature < 10) {
        cout << "Saat ini cuaca sedang dingin." << endl;
    }

    return 0;
}
```

Gambar 4.19 penulisan lebih dari if

Gambar 4.19 merupakan contoh berikutnya dari conditional statement ‘if’. Di dalam program ini terdapat ‘cout’ dan’cin’ di mana pengguna dapat memasukkan angka untuk suhu. Berikut merupakan penjelasan lengkapnya:

a. double

Dalam program ini, dalam fungsi ‘main()’, dideklarasikan variabel ‘temperature’ sebagai double. Variabel ini akan digunakan untuk menyimpan suhu yang dimasukkan oleh pengguna.

b. cout dan cin

Seperti yang telah Anda pelajari sebelumnya, cout

digunakan untuk mencetak tulisan sehingga muncul ke dalam program ketika dijalankan. Sedangkan cin digunakan agar pengguna bisa memasukkan data ke dalam program. Data yang dimasukkan oleh pengguna tersebut akan disimpan di double.

c. Pernyataan ‘if (temperature > 25)

Pernyataan ini digunakan untuk mengevaluasi data yang dimasukkan pengguna. Pembuat program menetapkan bahwa suhu di atas 25 derajat adalah situasi cuaca yang sedang panas. Sehingga ketika temperature yang dimasukkan pengguna lebih dari 25 program akan langsung mengeksekusi dan menampilkan bahwa cuaca sedang panas dengan fungsi ‘cout << “Saat ini cuaca sedang panas.” << endl;

d. Pernyataan ‘if (temperature >= 10 && temperature <=25)

Pernyataan ini akan menyeleksi dan mengevaluasi apakah suhu yang dimasukkan oleh pengguna berada di antara 10 dan 25 derajat celcius. Jika temperatur yang dimasukkan pengguna berada dalam 10 – 25 derajat maka, program akan secara langsung mengeksekusi dan menampilkan bahwa “Saat ini cuaca sedang nyaman.”

e. Pernyataan ‘if (temperatur < 10)

Sama seperti pernyataan if sebelumnya, di dalam pernyataan ini, apabila temperatur yang dimasukkan pengguna kurang dari 10 makan program akan secara otomatis tereksekusi dan mencetak pesan “Saat ini cuaca sedang dingin.” ke layar menggunakan ‘cout’.

Anda dapat segera mencobanya secara langsung, dan buatlah perubahan-perubahan pada angka yang dipilih

dari contoh di atas. Anda pula dapat mengembangkan program Anda sendiri dari contoh program di atas.

2. If-Else

Pernyataan if-else adalah salah satu bentuk conditional statement paling dasar dalam C++. Ini memungkinkan program untuk memilih di antara dua blok kode yang akan dieksekusi berdasarkan kondisi yang dievaluasi. Struktur dasar dari if-else statement adalah sebagai berikut:

```
if (kondisi) {
   // blok kode yang akan dieksekusi jika kondisi bernilai true
} else {
   // blok kode yang akan dieksekusi jika kondisi bernilai false
}
```

Contoh penggunaan If-else

if.cpp

```cpp
#include <iostream>

using namespace std;

int main() {
    int num = 10;

    // Memeriksa apakah num lebih besar dari 5
    if (num > 5) {
        cout << "Num lebih besar dari 5" << endl;
    } else {
        cout << "Num tidak lebih besar dari 5" << endl;
    }

    return 0;
}
```

Gambar 4. 20 penulisan if else

Gambar 4. merupakan contoh sederhana dari if-else statement. Berikut merupakan penjelasan lengkapnya:

a. Deklarasi variabel ‘num’ sebagai integer dan menginisialisasinya dengan nilai 10.

b. If (num > 5) {...}else{...}

Blok kode di atas merupakan struktur kontrol percabangan. Jika kondisi di dalam tanda kurung () terpenuhi (nilai num lebih besar dari 5), maka blok kode di dalam kurung kurawal {} setelah if akan dieksekusi. Jika tidak, blok kode di dalam kurung kurawal {} setelah else akan dieksekusi. Blok kode akan dieksekusi menggunakan cout.

if.cpp

```cpp
#include <iostream>

using namespace std;

int main() {
    int angka;

    // Meminta pengguna memasukkan sebuah angka
    cout << "Masukkan sebuah angka: ";
    cin >> angka;

    // Memeriksa apakah angka positif, negatif, atau nol
    if (angka > 0) {
        cout << "Angka yang dimasukkan adalah positif." << endl;
    } else if (angka < 0) {
        cout << "Angka yang dimasukkan adalah negatif." << endl;
    } else {
        cout << "Angka yang dimasukkan adalah nol." << endl;
    }

    return 0;
}
```

Gambar 4.21 penulisan if else if

Gambar 4. merupakan contoh berikutnya dari if-else statement. Dalam program ini, pengguna dapat menginput data. Hal ini ditunjukkan dengan fungsi cout. Berikut merupakan penjelasan singkatnya:

a. ‘int angka’

Pembuat program mendeklarasikan variabel ‘angka’ sebagai integer, yang berikutnya akan digunakan untuk menyimpan angka yang dimasukkan oleh pengguna. Penting untun diingat, bahwa integer variabel yang dideklarasikan ini harus secara konsisten digunakan dengan penulisan yang benar.

b. Cout dan Cin

Di program ini, pengguna dapat memasukkan sebuah angka ke layar. Angka yang dimasukkan nantinya menjadi data dan disimpan ke dalam ‘int angka’. Setelah pengguna memasukkan angka, program Anda akan menampilkan angka yang telah diproses melalui ‘cin’.

c. If-else

If-else digunakan untuk memeriksa dan mengevaluasi nilai dari variabel ‘angka’. Terdapat tiga poin conditional statements, antara lain:

a. Jika nilai ‘angka’ lebih besar dari 0, program mencetak “Angka yang dimasukkan adalah positif.”

b. Jika nilai ‘angka’ kurang dari 0, program mencetak “Angka yang dimasukkan adalah negatif.”

c. Jika nilai ‘angka’ sama dengan 0, program mencetak “Angka yang dimasukkan adalah nol.”

If-else sangat berguna ketika kita hanya memiliki dua kemungkinan hasil yang ingin kita evaluasi. Namun, dalam beberapa kasus, kita mungkin memiliki lebih dari dua kemungkinan hasil. Dalam hal ini, kita dapat menggunakan if-else-if.

3. If-else-if

Conditional statement berikutnya ialah If-else if. If-else-if statement merupakan perkembangan dari if-else statement. Dalam penggunaanya, dibutuhkan lebih dari dua hasil. If-else-if statement memunginkan program untuk memilih di antara beberapa blok kode untuk dieksekusi berdasarkan beberapa kondisi yang berbeda. Struktur dasar dari if-else-if statement adalah sebagai berikut:

```
if (kondisi1) {
    // blok kode yang akan dieksekusi jika kondisi1 bernilai true
```

} else if (kondisi2) {

// blok kode yang akan dieksekusi jika kondisi1 false dan kondisi2 true

} else {

// blok kode yang akan dieksekusi jika semua kondisi bernilai false

}

Contoh progmram C++ If-else-if conditional statement:

```
5  int main() {
6    int nilaiUjian;
7
8    cout << "Masukkan nilai ujian Anda (0-100): ";
9    cin >> nilaiUjian;
10
11   if (nilaiUjian >= 90) {
12     cout << "Selamat! Anda mendapatkan nilai A" << endl;
13   } else if (nilaiUjian >= 80) {
14     cout << "Bagus! Anda mendapatkan nilai B" << endl;
15   } else if (nilaiUjian >= 70) {
16     cout << "Cukup! Anda mendapatkan nilai C" << endl;
17   } else if (nilaiUjian >= 60) {
18     cout << "Lulus! Anda mendapatkan nilai D" << endl;
19   } else {
20     cout << "Maaf, Anda tidak lulus. Nilai Anda E" << endl;
21   }
22
23   return 0;
24 }
```

Gambar 4.22 penulisan if else if

Gambar di atas merupakan contoh dari program yang menggunakan conditional statemen if-else-if. Program ini meminta pengguna untuk memasukkan nilai ujian

dari 0-100. Selanjutnya, program akan dengan sendirinya mengecek, mengevaluasi serta mengeksekusi program berdasarkan kondisi-kondisi yang telah dibuat oleh pembuat program. Pernyataan if-else-if digunakan untuk menentukan kelulusan dan kategori penilaian, antara lain:

a. Jika nilaiUjian >= 90, maka program akan mencetak output “Selamat! Anda mendapatkan nilai A”. Hal ini ditunjukkan dengan conditional statement if {...}

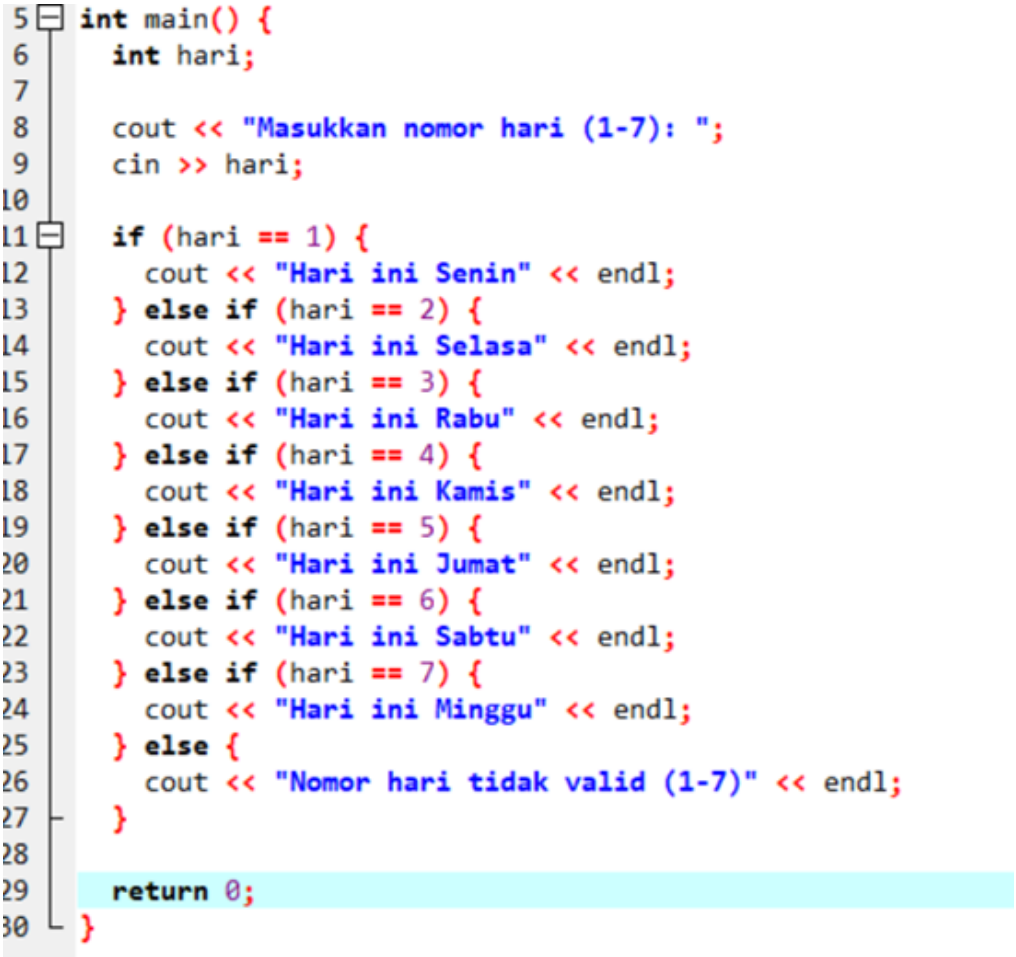
b. Jika nilaiUjian >= 80, maka program akan mencetak output “Bagus! Anda mendapatkan nilai B”. Hal ini ditunjukkan dengan conditional statement else if {...}

c. Jika nilaiUjian >=70, maka program akan mencetak output “Cukup! Anda mendapatkan nilai C”. Hal ini ditunjukkan dengan conditional statement else if {...}

d. Jika nilaiUjian >=60, maka program akan mencetak output “Lulus! Anda mendapatkan nilai D”. Hal ini ditunjukkan dengan conditional statemen else if{...}

e. Jika nilaiUjian di bawah angka-angka di atas, maka program akan mencetak output “Maaf, Anda tidak lulus. Nilai Anda E”. Hal ini ditunjukkan dengan conditonal statement else{....}

Program ini memiliki lima kondisi sehingga pembuat program menggunakan conditional statement if-else-if dan satu kondisi else untuk mengeksekusi nilai ujian di bawah 60. Anda dapat mencoba program ini secara langsung, dan melakukan input nilai ke dalam program.



```
int main() {
  int hari;

  cout << "Masukkan nomor hari (1-7): ";
  cin >> hari;

  if (hari == 1) {
    cout << "Hari ini Senin" << endl;
  } else if (hari == 2) {
    cout << "Hari ini Selasa" << endl;
  } else if (hari == 3) {
    cout << "Hari ini Rabu" << endl;
  } else if (hari == 4) {
    cout << "Hari ini Kamis" << endl;
  } else if (hari == 5) {
    cout << "Hari ini Jumat" << endl;
  } else if (hari == 6) {
    cout << "Hari ini Sabtu" << endl;
  } else if (hari == 7) {
    cout << "Hari ini Minggu" << endl;
  } else {
    cout << "Nomor hari tidak valid (1-7)" << endl;
  }

  return 0;
}
```

Gambar 4.23 penulisan kondisi if else if

Gambar di atas merupakan contoh berikutnya dari conditional statement if-else-if. Program ini meminta pengguna untuk memasukkan nomor hari dimulai dari 1-7. Berikut unsur-unsur yang terdapat dalam program di atas.

a. Program ini meminta pengguna untuk memasukkan nomor hari dari 1-7.

b. Statement if-else-if digunakan untuk mengeksekusi dan mencetak nama hari.

c. Jika hari yang diinput oleh pengguna adalah 1, maka

output yang akan dicetak “Hari ini Senin”.

d. Jika hari yang diinput oleh pengguna adalah 2, maka output yang akan dicetak “Hari ini Selasa”.

e. Jika hari yang diinput oleh pengguna adalah 3, maka output yang akan dicetak “Hari ini Rabu”.

f. Jika hari yang diinput oleh pengguna adalah 4, maka output yang akan dicetak “Hari ini Kamis”.

g. Jika hari yang diinput oleh pengguna adalah 5, maka output yang akan dicetak “Hari ini Jumat”.

h. Jika hari yang diinput oleh pengguna adalah 6, maka output yang akan dicetak “Hari ini Sabtu”.

i. Jika hari yang diinput oleh pengguna adalah 7, maka output yang akan dicetak “Hari ini Minggu”.

j. Apablia pengguna memasukkan angka selain 1-7, maka output yang akan dicetak adalah “Nomor hari tidak valid (1-7)”.

4. Nested If

Nested if adalah salah satu conditional statemen dalam bahasa pemrograman C++ yang memungkinkan Anda membuat keputusan bersyarat dan bertingkat. Nested if atau percabangan bersarang dapat memeriksa beberapa kondisi secara berurutan, dan mengeksekusi bagian kode yang berbeda tergantung pada hasil dari setiap pemeriksaan. Hal ini memungkinkan Anda untuk menulis program yang lebih fleksibel dan menangani berbagai skenario logis dengan elegan. Berikut merupakan struktur umum nested if:

```
if (kondisi_luar) {
  if (kondisi_dalam1) {
     // Pernyataan yang dieksekusi jika kondisi luar dan dalam1 benar
  } else if (kondisi_dalam2) {
    // Pernyataan yang dieksekusi jika kondisi luar benar dan kondisi_dalam1 salah, tapi kondisi_dalam2 benar
  } else {
    // Pernyataan yang dieksekusi jika kondisi luar benar dan kedua kondisi_dalam salah
  }
} else {
  // Pernyataan yang dieksekusi jika kondisi luar salah
}
```

Struktur di atas merupakan struktur umum dari nested if dalam bahasa pemrograman C++. Dalam statement pertama dituliskan if(kondisi_luar) yang berarti kondisi awal yang dievaluasi terlebih dahulu. Blok kode tersebut jika bernilai true, maka blok kode di dalam if ini akan dieksekusi. Selanjutnya adalah if (kondisi_dalam) dan else if (kondisi_dalam2). Kedua statement ini ditepatkan di dalam blok if yang pertama sehingga terciptalah percabangan (nested if) yang memeriksa kondisi tambahan. Kondisi_dalam1 dan kondisi_dalam2 adalah kondisi tambahan yang dievaluasi di dalam blok if luar. Dalam if (kondisi_dalam) jika kondisi_dalam1 bernilai true, maka blok kode di dalam if (kondisi_dalam)

pertama akan dieksekusi. Selanjutnya blok else if memungkinkan Anda untuk memeriksa kondisi tambahan lain jika kondisi_dalam1 salah. Blok else yang berada di dalam if(kondisi_dalam) jika semua statement if dan else if dalam percabanga dalam salah, maka blok else dalam akan dieksekusi. Unsur terakhir adalah blok else luar. Jika kondisi_luar bernilai false, maka blok kode di dalam else luar akan dieksekusi. Dari struktur dasar yang telah dijelaskan di atas, berikut adalah contoh penggunaan:

```
int main() {
  double nilai_rata2, kehadiran;

  cout << "Masukkan nilai rata-rata: ";
  cin >> nilai_rata2;

  cout << "Masukkan persentase kehadiran (dalam angka desimal 0.0 - 1.0): ";
  cin >> kehadiran;

  if (nilai_rata2 >= 80) {
    if (kehadiran >= 0.8) {
      cout << "Selamat! Anda lulus dengan nilai baik dan kehadiran tinggi." << endl;
    } else {
      cout << "Anda lulus, namun perlu meningkatkan kehadiran." << endl;
    }
  } else if (nilai_rata2 >= 70) {
    if (kehadiran >= 0.9) {
      cout << "Anda lulus dengan nilai cukup dan kehadiran baik." << endl;
    } else {
      cout << "Anda lulus dengan nilai cukup, namun perlu meningkatkan kehadiran dan belajar lebih giat." << endl;
    }
  } else {
    cout << "Anda tidak lulus. Tetap semangat dan belajar lebih giat untuk semester berikutnya." << endl;
  }

  return 0;
}
```

Gambar 4.24 penulisan nested if

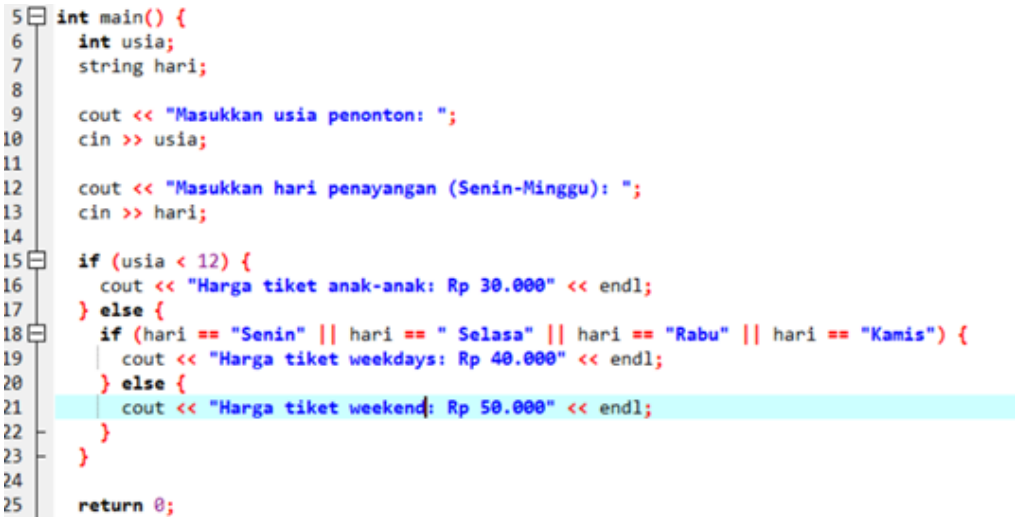
Contoh program pada Gambar 4. , pengguna diminta memasukkan nilai rata-rata dan persentase kehadiran dalam angka desimal 0.0-1.0. Program di atas merupakan salah satu contoh nested if, karena terdapat percabangan if (if di dalam if). Blok if yang utama adalah if (nilai_rata2 >= 80). Blok if ini mengevaluasi apakah nilai_rata2 lebih besar dari atau sama dengan 80. Jika termasuk dari kondisi yang dimaksud maka program

akan langsung mengeksekusi ke input selanjutnya, yakni ke blok if dalam untuk memeriksa nilai kehadiran. Namun jika nilai_rata2 yang diinput kurang dari 80, program melompat langsung ke blok else if. Selanjutnya, apabila jumlah nilai rata-rata lebih dari atau sama dengan 80, maka program berlanjut ke blok if dalam kehadiran >= 0.8. Blok ini mengevaluasi apakah kehadiran lebih besar atau sama dengan 0.8. Jika kehadiran >= 0.8, program menampilkan pesan ucapan selamat karena siswa lulus dengan nilai baik dan kehadiran tinggi.

Unsur selanjutnya dari contoh di atas ialah blok else if (Nilai rata-rata >=70. Blok ini akan dieksekusi jika nilai_rata2 tidak mencapai 80, namun masih lebih besar dari atau sama dengan 70. Blok ini juga memeriksa nilai kehadiran untuk menentuan pesan yang sesuai. Blok if Dalam (kehadiran >= 0.9) serupa dengan blok if dalam di atas. Jika kehadrian >=0.9, program menampilkan pesan bahwa siswa lulus dengan nilai cukup dan kehadiran baik.

Unsur nested if yang terakhir adlah blok else (nilai_rata2 < 70). Blok ini dieksekusi jika nilai_rata2 kurang dari 70 dan berarti siswa tidak lulus. Program menampilkan pesan yang mendorong siswa untuk tetap semangat dan belajar lebih giat untuk semester berikutnya, dengan:

cout << “Anda tidak lulus. Tetap semangat dan belajar lebih giat untuk semester berikutnya.” << endl;



```
int main() {
  int usia;
  string hari;

  cout << "Masukkan usia penonton: ";
  cin >> usia;

  cout << "Masukkan hari penayangan (Senin-Minggu): ";
  cin >> hari;

  if (usia < 12) {
    cout << "Harga tiket anak-anak: Rp 30.000" << endl;
  } else {
    if (hari == "Senin" || hari == " Selasa" || hari == "Rabu" || hari == "Kamis") {
      cout << "Harga tiket weekdays: Rp 40.000" << endl;
    } else {
      cout << "Harga tiket weekend: Rp 50.000" << endl;
    }
  }

  return 0;
```

Gambar 4.25 penulisan nested if

Gambar 4. merupakan contoh kedua dari nested if yang digunakan untuk membuat keputusan berdasarkan beberapa kondisi yang saling bergantung. Terdapat dua percabangan yakni, blok if luar dan blok if dalam. Berikut penjelasannya:

a. Blok if luar:

• Memeriksa apakah usia penonton kurang dari 12 tahun.

o Jika usia < 12 :

 Mencetak harga tiket anak-anak (Rp 30.000).

o Jika usia >= 12:

 Program beralih ke blok if dalam untuk menentukan harga berdasarkan hari penayangan.

b. Blok if dalam:

• Memeriksa apakah hari penayangan termasuka hari kerja .

- Jika hari yang diinput termasuk ke dalam hari kerja, maka akan mencetak harga tiket weekdays (Rp 40.000).
- Jika hari yang diinput bukan hari kerja, maka akan mencetak harga weekend (Rp 50.000).

Dengan demikian, struktur nested if pada contoh ini memungkinkan program untuk menentukan harga tiket secara lebih rinci. Blok if luar menentukan harga tiket secara umum berdasarkan usia, sementara blok if dalam selanjutnya menentukan harga berdasarkan hari penayangan. Dengan struktur ini, harga tiket yang tepat dapat dihitung untuk setiap kombinasi usia dan hari.

5. Switch

Switch merupakan bagian terakhir dari decision making C++. Struktur ini memungkinkan Anda mengeksekusi kode yang berbeda berdasarkan nilai yang Anda berikan. Switch biasanya digunakan untuk menangani situasi yang memerulukan tindakan berbeda dari beberapa kondisi. Berikut struktur umum switch:

```
switch (nilai_ekspresi) {
  case nilai1:
    // Blok kode untuk nilai1
    break;
  case nilai2:
    // Blok kode untuk nilai2
    break;
  case nilai3:
    // Blok kode untuk nilai3
```

```
    break;
  // ... lebih banyak kasus
  default:
    // Blok kode default (opsional)
}
```

Ada tiga elemen berbeda yang digunakan dalam struktur switch, yakni:

a. nilai_ekspresi

nilai_ekspresi adalah ekspresi yang nilainya akan digunakan untuk menentukan kasus mana yang akan dieksekusi.

b. case

case merupakan awal dari kasus tertentu.

c. Default

Default merupakan blok kode opsional yang akan dieksekusi jika tidak ada nilai kasus yang cocok dengan nilai_ekspresi.

Berikut adalah contoh penggunaan switch

```
int main() {
  int pilihan;

  cout << "Menu Pilihan:" << endl;
  cout << "1. Tambah" << endl;
  cout << "2. Kurang" << endl;
  cout << "3. Kalikan" << endl;
  cout << "4. Bagi" << endl;
  cout << "Masukkan pilihan Anda: ";
  cin >> pilihan;

  switch (pilihan) {
    case 1:
      int a, b;
      cout << "Masukkan dua angka untuk ditambahkan: ";
      cin >> a >> b;
      cout << a << " + " << b << " = " << a + b << endl;
      break;
    case 2:
      int c, d;
      cout << "Masukkan dua angka untuk dikurangi: ";
      cin >> c >> d;
      cout << c << " - " << d << " = " << c - d << endl;
      break;
    case 3:
      int e, f;
      cout << "Masukkan dua angka untuk dikalikan: ";
      cin >> e >> f:
```

Gambar 4.26 penulisan switch case

```
      cout << e << " * " << f << " = " << e * f << endl;
      break;
    case 4:
      int g, h;
      cout << "Masukkan dua angka untuk dibagi: ";
      cin >> g >> h;
      cout << g << " / " << h << " = " << g / h << endl;
      break;
    default:
      cout << "Pilihan tidak valid." << endl;
  }

  return 0;
}
```

Gambar 4.27 penulisan switch case

Gambar 4.26 dan 4.27 merupakan contoh program

switch. Program ini menampilkan menu pilihan kepada pengguna (1-4). Pengguna perlu memasukkan input antara 1-4. Pernyataan switch dalam program ini mengevaluasi pilihan: tambah, kurang, kali, bagi. Jika nilai cocok dengan salah satu nilai case, blok kode untuk kasus tersebut dieksekusi. Pernyataan break di setiap kasus memastikan bahwa eksekusi program berhenti setelah blok kode kasus dijalankan. Jika nilai pilihan tidak cocok dengan nilai kasus manapun, blok kode default akan dieksekusi.

Switch memiliki beberapa manfaat, antara lain:

a. Kejelasan: mempermudah logika pemrograman

b. Efisiensi: dengan menggunakan switch, pembuat program dapat mengurangi penggunaan pernyataan if-else bertingkat yang berlebih.

Secara keseluruhan, memahami dan menggunakan pernyataan switch sebagai decision making dalam bahasa pemrograman C++ dapat membantu Anda menulis kode secara efektif dan efisien. Selain itu penggunaan decision making yang tepat akan membuat program yang Anda ciptakan menjadi lebih terstruktur, efisien, dan mudah dipahami di berbagai kondisi. Untuk semakin memperjelas kode yang Anda buat, Anda dapat menambahkan komentar pada kode kasus tertentu.

5
Looping

LOOPING C++

Looping adalah konsep dasar dalam bahasa pemrograman C++. Ini memungkinkan kita untuk mengeksekusi serangkaian pernyataan berulang kali, berdasarkan kondisi yang ditentukan. Looping memungkinkan pengulangan tugas- tugas yang sama tanpa perlu menulis pernyataan yang sama berulang kali.

Dalam bahasa C++ tersedia suatu fasilitas yang digunakan untuk melakukan proses yang berulang-ulang sebanyak keinginan kita. Misalnya saja, bila kita ingin menginput dan mencetak bilangan dari 1 sampai 100 bahkan 1000, tentunya kita akan merasa kesulitan. Namun dengan struktur perulangan proses, kita tidak perlu menuliskan perintah sampai 100 atau 1000 kali, cukup dengan beberapa perintah saja, maka dari itu Looping datang dengan suatu manfaat, yaitu :

1. Automatisasi

Looping memungkinkan otomatisasi proses pengulangan kode tanpa harus menulis kode yang sama berulang kali.

2. Peningkatan Efisiensi

Dengan looping, kode program menjadi lebih efisien

karena instruksi yang sama dapat dieksekusi berulang kali tanpa penulisan ulang.

3. Penanganan Data

Looping memungkinkan penanganan data yang besar atau kompleks dengan cara yang terstruktur dan efisien.

Struktur perulangan dalam bahasa C mempunyai bentuk yang bermacam macam. Sebuah/kelompok instruksi diulang untuk jumlah pengulangan tertentu. Baik yang terdifinisikan sebelumnya ataupun tidak. Struktur pengulangan terdiri atas dua bagian :

1. Kondisi pengulangan yaitu ekspresi boolean yang harus dipenuhi untuk melaksanakan pengulangan;

2. Isi atau badan pengulangan yaitu satu atau lebih pernyataan (aksi) yang akan diulan

Dengan pemahaman yang mendalam tentang konsep looping dan penggunaan yang tepat, seorang pengembang dapat menulis kode program yang lebih efisien, terstruktur, dan mudah dipelihara. Praktek dan eksperimen yang terus-menerus akan membantu meningkatkan pemahaman dan keterampilan dalam menggunakan looping dalam pemrograman.

Dalam C++, ada beberapa jenis looping yang umum digunakan, seperti for, while, dan do-while. Mari kita eksplorasi lebih lanjut tentang konsep dan penggunaan looping dalam bahasa pemrograman C++. Mari kita

membahas jenis - jenis Looping yang ada didalam bahasa C++.

1. FUNGSI LOOP FOR

For adalah salah satu jenis loop yang umum digunakan dalam pemrograman untuk mengulang kode program sejumlah kali dengan menggunakan variabel yang diinisialisasi, kondisi loop, dan perubahan variabel iterasi. Dalam konteks bahasa pemrograman C++, Loop For memiliki struktur yang terdiri dari tiga bagian:

a) Inisialisasi Variabel Iterasi

b) Kondisi Loop

c) Perubahan Variabel Iterasi

Pada awal loop, variabel iterasi diinisialisasi dengan nilai awal. Selanjutnya, kondisi loop dievaluasi, dan jika kondisi tersebut terpenuhi, kode program di dalam loop akan dieksekusi. Setelah setiap iterasi, variabel iterasi akan diubah sesuai dengan perubahan yang telah ditentukan. Loop For akan terus berjalan hingga kondisi loop tidak terpenuhi.

Dengan menggunakan Loop For, pengguna dapat mengulang sebuah statement atau sekelompok statement sebanyak nilai yang telah ditentukan sebelumnya. Hal ini memungkinkan operasi untuk terus dilakukan sampai jumlah pengulangan telah mencapai batas yang telah ditetapkan di awal. Loop For sangat berguna ketika kita ingin melakukan perhitungan atau

operasi yang memerlukan pengulangan sejumlah tertentu.

Dalam implementasinya, Loop For dapat digunakan untuk berbagai keperluan, seperti mengakses elemen dalam sebuah array, melakukan iterasi pada data, atau menjalankan operasi yang memerlukan pengulangan sejumlah tertentu. Dengan struktur yang jelas dan terorganisir, Loop For memungkinkan pengembang untuk mengontrol proses pengulangan dengan baik dan efisien.

Struktur perulangan for biasa digunakan untuk mengulang suatu proses yang telah diketahui jumlah perulangannya. Dari segi penulisannya, struktur perulangan for tampaknya lebih efisien karena susunannya lebih simpel dan sederhana.

Pernyataan for digunakan untuk melakukan looping. Pada umumnya looping yang dilakukan oleh for telah diketahui batas awal, syarat looping dan perubahannya. Selama kondisi terpenuhi, maka pernyataan akan terus dieksekusi. Bentuk umum perulangan for adalah sebagai berikut :

```
For ( inisialisasiNilai; SyaratPerulangan;
PerubahanNilai )
{
        Statement yang diulang;
```

Gambar 5.1 konsep penulisan for

a) Ungkapan1 merupakan statement awal (inisialisasi)

b) Ungkapan2 merupakan kondisi/syarat perulangan dilakukan

c) Ungkapan3 merupakan statement control untuk perulangan

Statement merupakan pernyataan/perintah yang dijalankan jika syarat terpenuhi.

```
for(a=1;a<=5;a++) {
cout<<"Hello World! \n"
}
```

Gambar 5.2 penulisan for

Perhatikan perintah operator --, operator – digunakan untuk decrement

Gambar 5.3 penggunaan decremen pada for

Perintah diatas mernampilak kalimat “Hello World!” sebanyak 5 baris

Tanda “a=1” adalah nilai awal variabel a. Tanda “a<=5” syarat pengulangan. Tanda “a++” kondisi pengulangan.

Gambar 5.4 penulisan for dengan increment

Perintah diatas menampilkan abjad Z – A

```
for(huruf="Z";huruf>="A";huruf--)
{
Cout<<Abjad "<<huruf<<"\n";
}
```

Gambar 5.5 penulisan for menampilkan abjad

```
#include <iostream.h>
#include <conio.h>

int main(){
    int bil, n;
    cout << "Masukkan n = ";
    cin >> n;
    for (bil = 0; bil < n; bil++)
    {
        if (bil % 2 == 0) cout << bil << " ";
    }
}
```

Gambar 5.6 penulisan for mencetak bilangan genap

Di atas ini adalah program untuk mencetak bilangan genap yang kurang dari n (n diperoleh dari input).

```
#include <iostream>

int main() {
    for (int i = 0; i < 5; i++) {
        std::cout << "Hello, World!" << std
            ::endl;
    }
    return 0;
}
```

Gambar 5.7 penulisan for dengan std::cout

Keterangan:

a) for adalah keyword untuk menginisialisasi loop.

b) (int i = 0; i < 5; i++) adalah kondisi loop yang

terdiri dari tiga bagian:

1. int i = 0 adalah inisialisasi variabel i dengan nilai 0.

2. i < 5 adalah kondisi loop yang akan diulang jika i kurang dari 5.

3. i++ adalah perintah untuk meningkatkan nilai i setiap loop.

c) std::cout << “Hello, World!” << std::endl; adalah kode program yang akan diulang beberapa kali.

Dengan pemahaman yang baik tentang loop for, kita dapat membuat kode yang lebih terstruktur, mudah dimengerti dan efisien. Ini adalah alat yang sangat berguna dalam pemrograman C++, terutama dalam skenario di mana jumlah iterasi yang tetap diperlukan dan struktur yang terdefinisi dengan jelas sangat dibutuhkan.

2. FUNGSI LOOP WHILE

Loop While adalah jenis loop yang digunakan untuk mengulang kode program beberapa kali dengan menggunakan kondisi yang diuji setiap loop. Loop while terdiri dari tiga bagian utama, yaitu :

a) Deklarasi Loop

Deklarasi Loop adalah bagian pertama dari Loop While. Di sini, kita mendeklarasikan variabel yang akan

digunakan sebagai counter untuk menghitung jumlah loop yang akan dijalankan.

b) Kondisi Loop

Kondisi Loop adalah bagian kedua dari Loop While. Di sini, kita mendefinisikan kondisi yang akan diuji setiap loop. Kondisi Loop dapat berupa perbandingan, operasi boolean, atau lain-lain. Kondisi Loop akan diuji setiap loop, dan jika kondisi tersebut tidak terpenuhi, loop akan berhenti.

c) Perubahan Variabel

Perubahan Variabel adalah bagian terakhir dari Loop While. Di sini, kita mendefinisikan cara mengubah nilai variabel counter setiap loop. Perubahan Variabel dapat berupa inkrementasi, dekrementasi, atau lain- lain.

Dengan menggunakan Loop While, kita dapat mengulang kode program beberapa kali dengan menggunakan kondisi yang diuji setiap loop. Loop While sangat berguna ketika kita ingin mengulang kode program beberapa kali dengan menggunakan kondisi yang diuji setiap loop. Didalamnya pun terdapat beberapa runtutan dalam menulis program While, sebagai berikut :

a) Inisialisasi Variabel

Sebelum loop dimulai, kita seringkali menginisialisasi sebuah variabel yang akan digunakan sebagai counter atau penanda kondisi.

b) Evaluasi Kondisi

Setelah inisialisasi variabel, kondisi loop dievaluasi. Jika kondisi tersebut bernilai true, maka blok kode di dalam loop akan dieksekusi. Jika kondisi tersebut bernilai false, loop akan diakhiri.

c) Eksekusi Blok Kode

Jika kondisi loop terpenuhi (true), maka blok kode di dalam loop akan dieksekusi. Dalam contoh di atas, kita mencetak nilai i dan kemudian meningkatkan nilainya.

d) Peningkatan Variabel

Setelah eksekusi blok kode di dalam loop, seringkali kita memperbarui nilai variabel yang digunakan dalam kondisi loop. Ini penting untuk mencegah terjadinya loop tak terbatas atau infinite loop.

e) Keluar dari Loop

Setelah eksekusi blok kode, kontrol kembali ke awal loop untuk mengevaluasi kondisi. Proses ini berlanjut hingga kondisi loop tidak lagi terpenuhi, dan eksekusi keluar dari loop.

Perulangan WHILE banyak digunakan pada program

yang terstruktur. Perulangan ini banyak digunakan bila jumlah perulangannya belum diketahui. Proses perulangan akan terus berlanjut selama kondisinya bernilai benar (≠0) dan akan berhenti bila kondisinya bernilai salah (=0).

```
while (syarat pengulangan)
{
        statement yang dijalankan;
        statement control:
}
```

Gambar 5.8 konsep penulisan while

```
#include <iostream>
using namespace std;

int main() {
    int i = 1;
    while (i <= 5) {
        cout << i << " ";
        i++;
    }
    return 0;
}
```

Gambar 5.9 contoh penulisan while

Dalam contoh di atas, kita mulai dengan menginisialisasi variabel i dengan nilai 1. Kemudian, kita memulai loop while dengan kondisi i <= 5. Setiap kali loop dieksekusi, kita mencetak nilai i dan kemudian

meningkatkan nilainya dengan i++. Proses ini terus berlanjut hingga kondisi i <= 5 tidak lagi terpenuhi.

Kedua Perintah Program ini adalah identik

```
#include <iostream>

for (a = 1; a <= 5; a++)
{
cout << "Hello world \n";
}
```

Gambar 5.10 penulisan for

```
a = 1;
while (a <= 5)
{
    cout << "Hello world \n";
    a++;
}
```

Gambar 5.11 penulisan while

Jika Anda menggunakan WHILE, pastikan bahwa suatu saat bagian kondisi sampai bernilai FALSE. Apabila tidak, proses perulangan akan terus berjalan selamanya.

Gambar 5.12 peringatan penggunaan while

Contoh program di bawah ini digunakan untuk menjumlahkan sejumlah data angka. Angka yang akan dijumlahkan diinputkan satu-persatu. Proses pemasukan data angka akan berhenti ketika dimasukkan angka –1. Setelah itu tampil hasil penjumlahannya.

```
#include <iostream.h>
#include <conio.h>
void main()
{
    int data, jumlah,cacah;
    jumlah = 0;
    data = 0;
    cacah = 0;
    while (data != -1)
    {
        cout << "Masukkan data angka : ";
        cin >> data; jumlah += data; cacah++;
    }
    cout << "Jumlah data adalah : " << jumlah" << endl;
```

Gambar 5.13 penulisan input output di dalam looping while

Kelebihan dan Kelemahan dalam menggunakan Fungsi While dalam bahasa pemograman C++. Kelebihan Loop While sebagai berikut :

1. Fleksibel untuk jumlah iterasi yang tidak diketahui

Salah satu kelebihan utama dari loop while adalah fleksibilitasnya dalam menangani kasus di mana jumlah iterasi tidak diketahui sebelumnya. Loop ini akan terus berjalan selama kondisi yang ditentukan masih terpenuhi, tanpa memerlukan pengetahuan sebelumnya tentang jumlah iterasi yang diperlukan.

2. Sederhana dan mudah dipahami

Loop while memiliki struktur yang sederhana, hanya terdiri dari kondisi yang dievaluasi sebelum setiap iterasi. Hal ini membuatnya mudah dipahami oleh pemula dan meminimalkan kesalahan dalam penulisan kode.

3. Memungkinkan iterasi berdasarkan perubahan variabel

Loop while sangat berguna ketika kita ingin melakukan iterasi berdasarkan perubahan variabel tertentu. Misalnya, kita dapat menggunakan loop while untuk membaca data dari input pengguna hingga kondisi tertentu terpenuhi.

4. Mencegah potensi infinite loop

Karena kondisi dievaluasi sebelum setiap iterasi, loop while dapat membantu mencegah terjadinya infinite loop jika kondisi tidak pernah terpenuhi. Hal ini dapat mengurangi risiko kegagalan atau crash dalam program.

Kelemahan Loop While

1. Rentan terhadap infinite loop

Salah satu kelemahan utama dari loop while adalah potensi untuk terjebak dalam infinite loop jika kondisi yang ditentukan tidak pernah bernilai false. Ini bisa terjadi jika pengaturan kondisi loop tidak benar atau tidak berubah selama iterasi.

2. perlu perhatian ekstra untuk peningkatan variabel

Karena loop while tidak memiliki bagian khusus untuk peningkatan atau dekremen variabel, pemrogram harus memastikan bahwa variabel yang digunakan dalam kondisi loop diperbarui dengan benar di dalam blok kode. Jika tidak, ini juga bisa menyebabkan infinite loop.

3. kurang efisien dalam kasus yang jumlah iterasinya diketahui Dalam kasus di mana jumlah iterasi sudah diketahui

sebelumnya, loop while mungkin kurang efisien dibandingkan dengan loop for. Loop for memungkinkan untuk menentukan inisialisasi, kondisi, dan peningkatan dalam satu baris, sementara loop while memerlukan pendeklarasian dan pembaruan variabel secara terpisah.

Meskipun memiliki beberapa kelemahan, loop while

tetap menjadi salah satu struktur pengulangan yang paling fleksibel dan umum digunakan dalam pemrograman C++. Dengan penggunaan yang tepat dan pemahaman yang baik tentang cara kerjanya, loop while dapat menjadi alat yang kuat dalam mengatasi berbagai masalah pemrograman.

Loop while adalah alat yang sangat berguna dalam pemrograman C++, memungkinkan kita untuk menjalankan blok kode secara berulang selama kondisi tertentu terpenuhi. Dengan pemahaman yang baik tentang cara kerjanya, kita dapat membuat program yang lebih fleksibel, efisien, dan andal. Namun, perlu diingat untuk selalu memperhatikan kondisi loop agar tidak jatuh ke dalam infinite loop yang tidak diinginkan.

3. FUNGSI LOOP DO –WHILE

Loop Do-While adalah jenis loop yang digunakan untuk mengulang kode program beberapa kali dengan menggunakan kondisi yang diuji setelah loop.

Pernyataan Do-While adalah salah satu jenis loop dalam pemrograman yang memungkinkan eksekusi pernyataan atau blok pernyataan hingga ekspresi yang ditentukan menjadi salah (false). Do-While memiliki struktur yang memungkinkan eksekusi setidaknya satu kali sebelum mengecek kondisi.

Perintah Do - While hampir sama dengan While sebelumnya

```
do
{
        Blok Pernyataan;
}
while(kondisi);
```

Gambar 5.14 konsep penulisan do while

Perbedaan dengan WHILE sebelumnya yaitu bahwa pada DO WHILE statement perulangannya dilakukan terlebih dahulu baru kemudian di cek kondisinya.

Sedangkan WHILE kondisi dicek dulu baru kemudia statement perulangannya dijalankan. Akibat dari hal ini adalah dalam DO WHILE minimal terdapat 1xnperulangan. Sedangkan WHILE dimungkinkan perulangan tidak pernah terjadi yaitu ketika kondisinya langsung bernilai FALSE.

```
a = 1;
do
{
cout << “Hello world \n”;
a++;
}
while(a==0);
```

Gambar 5.15 penulisan do while pada C++

Perintah di atas akan muncul satu buah Hello World. Bandingkan dengan yang berikut ini:

```
a = 1;
while(a==0)
{
cout << “Hello world \n”;
a++;
}
```

Gambar 5.16 penulisan while pada C++

Perintah di atas sama sekali tidak menampilkan Hello

World, karena kondisinya langsung FALSE.

Struktur Pernyataan Do-While

1. Pernyataan Do-While memiliki struktur umum sebagai berikut :

```
do {
        // pernyataan atau blok pernyataan
} while (ekspresi);
```

Gambar 5.17 konsep penulisan do while

2. Ekspresi dalam while akan dievaluasi setelah isi perulangan dijalankan. Oleh karena itu, tubuh perulangan selalu dieksekusi setidaknya sekali sebelum pengecekan kondisi.

Cara Kerja Pernyataan Do-While

1. Isi pernyataan dijalankan terlebih dahulu sebelum pengecekan kondisi.

2. Setelah isi perulangan dijalankan, ekspresi dievaluasi.

3. Jika ekspresi salah (false), pernyataan Do-While berakhir dan kontrol diteruskan ke pernyataan berikutnya dalam program.

4. Jika ekspresi benar (true), proses diulang, dimulai dari langkah 1

Kelebihan Loop Do-While

1. Eksekusi Setidaknya Satu Kali:

Loop do-while menjamin bahwa blok kode akan dieksekusi setidaknya satu kali, bahkan jika kondisi awalnya tidak terpenuhi. Ini bermanfaat dalam situasi di mana kita ingin menjalankan blok kode sekali dan kemudian mengevaluasi kondisi.

2. Fleksibilitas dalam Kasus dengan Jumlah Iterasi yang Tidak Diketahui:

Do-while sangat fleksibel dalam menangani kasus di mana jumlah iterasi tidak diketahui sebelumnya. Ini memungkinkan kita untuk menjalankan blok kode berulang kali hingga kondisi tertentu terpenuhi

Kelemahan Loop Do-While:

1. Potensi Infinite Loop:

Seperti loop while, do-while juga rentan terhadap infinite loop jika kondisi yang ditentukan tidak pernah bernilai false. Kondisi harus dikelola dengan hati-hati untuk menghindari situasi ini.

Dalam kasus di mana jumlah iterasi sudah diketahui

sebelumnya, do-while mungkin kurang efisien dibandingkan dengan loop for atau bahkan loop while. Ini karena do-while akan menjalankan blok kode sekali tanpa memeriksa kondisi pada awalnya.

Loop do-while adalah alat yang berguna dalam pemrograman C++, terutama dalam kasus di mana kita ingin menjamin bahwa blok kode dieksekusi setidaknya satu kali. Dengan pemahaman yang baik tentang cara kerjanya, do-while dapat digunakan secara efektif dalam berbagai situasi pemrograman. Namun, perlu diingat untuk memperhatikan kondisi agar tidak terjebak dalam infinite loop yang tidak diinginkan.

4. FUNGSI LOOP NESTED

Pengulangan Nested, juga dikenal sebagai Loop Nested, adalah konsep di mana satu loop ditempatkan di dalam loop lain. Hal ini memungkinkan untuk pengulangan yang lebih kompleks dan terstruktur. Pengulangan Nested sering digunakan untuk mengakses elemen dalam array multidimensi atau untuk mengatasi situasi yang memerlukan pengulangan dalam pengulangan. Hal ini berguna ketika kita perlu mengulangi serangkaian instruksi untuk setiap iterasi dari loop luar.

Kelebihan Loop Nested

1. Kemampuan untuk Memproses Data Berstruktur

Loop nested sangat berguna ketika kita perlu mengakses dan memproses data yang memiliki struktur bersarang, seperti array multi-dimensi atau matriks.

2. Fleksibilitas dalam Pencarian Pola atau Struktur Data

Dengan menggunakan loop nested, kita dapat dengan mudah mencari pola atau struktur tertentu dalam data yang kompleks.

3. Menghasilkan Output Berstruktur

Loop nested memungkinkan kita untuk menghasilkan output yang memiliki struktur yang terorganisir, seperti tabel atau pola yang kompleks.

Kelemahan Loop Nested

1. Kemungkinan Kinerja yang Buruk

Jika terlalu banyak loop bersarang atau jumlah iterasi yang besar, ini dapat mempengaruhi kinerja program dan menyebabkan waktu eksekusi yang lebih lama.

2. Kesulitan dalam Membaca dan Memahami Kode

Loop nested dapat membuat kode menjadi lebih sulit dibaca dan dipahami, terutama jika ada banyak tingkat loop bersarang.

3. Kesulitan dalam Debugging

Debugging kode dengan loop nested dapat menjadi lebih sulit karena kita perlu melacak bagaimana setiap tingkat loop mempengaruhi hasil akhir.

```cpp
#include <iostream>
using namespace std;

int main() {
    int rows = 5;
    for (int i = 1; i <= rows; ++i) {
        for (int j = 1; j <= i; ++j) {
            cout << "* ";
        }
        cout << endl;
    }
    return 0;
}
```

Gambar 5.18 penulisan looping nested

Gambar 5.19 hasil penulisan looping nested

Dalam contoh ini, loop dalam (for (int j = 1; j <= i; ++j)) ditempatkan di dalam loop luar (for (int i = 1; i <= rows; ++i)). Loop dalam akan diulang untuk setiap iterasi loop luar, sehingga mencetak jumlah bintang yang sesuai dengan nomor baris.

Dengan demikian, pengulangan Nested adalah salah satu konsep yang sangat berguna dalam pemrograman, terutama ketika kita perlu mengakses elemen dalam array multidimensi atau mengatasi situasi yang memerlukan pengulangan dalam pengulangan.

5. FUNGSI FOREACH LOOP

Foreach loop adalah jenis loop yang digunakan dalam pemrograman untuk mengulangi setiap elemen dalam suatu kumpulan data, seperti array, tanpa perlu khawatir tentang indeks atau kondisi. Loop ini berguna untuk mengakses dan memanipulasi elemen-elemen dalam sebuah kumpulan data dengan lebih mudah dan efisien.

Karakteristik Foreach Loop:

1. Iterasi Elemen

Foreach loop akan mengambil setiap elemen dalam kumpulan data dan menjalankan blok kode untuk setiap elemen tersebut.

2. Sederhana dan Mudah Dipahami

Loop ini lebih mudah digunakan untuk mengakses dan memanipulasi elemen dalam sebuah kumpulan data daripada loop lainnya.

3. Tidak Memerlukan Indeks

Anda tidak perlu khawatir tentang indeks atau kondisi saat menggunakan foreach loop.

Kelebihan Foreach Loop

1. Kode yang Lebih Ringkas dan Mudah Dipahami

Foreach loop membuat kode menjadi lebih ringkas

dan mudah dipahami karena tidak perlu mengurus indeks atau iterator secara eksplicit.

2. Pencegahan Kesalahan yang Umum

Karena foreach loop mengelola indeks secara internal, ini mengurangi risiko kesalahan dalam penanganan indeks, seperti penggunaan indeks yang salah atau melampaui batas.

3. Cocok untuk Struktur Data yang Bersarang

Foreach loop dapat digunakan untuk mengiterasi melalui struktur data yang bersarang dengan cara yang intuitif dan mudah dipahami.

Kelemahan Foreach Loop:

1. Keterbatasan dalam Manipulasi Indeks

Karena foreach loop tidak memberikan akses langsung ke indeks, kita mungkin menghadapi keterbatasan dalam melakukan manipulasi indeks atau mengakses elemen berdasarkan indeks tertentu.

2. Keterbatasan dalam Beberapa Bahasa Pemrograman

Tidak semua bahasa pemrograman mendukung foreach loop, atau mungkin memiliki implementasi yang berbeda. Oleh karena itu, fitur ini mungkin tidak tersedia dalam semua lingkungan pengembangan.

```cpp
#include <iostream>
using namespace std;

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    for (int x : arr) {
        cout << x << " ";
    }
    return 0;
}
```

```
/tmp/4u1j6W7xiQ.o
1 2 3 4 5

=== Code Execution Successful ===
```

Gambar 5.20 penulisan foreach looping

Foreach loop adalah alat yang sangat berguna dalam pemrograman untuk melakukan iterasi melalui elemen-elemen dalam sebuah struktur data dengan cara yang mudah dipahami dan ringkas. Meskipun memiliki beberapa keterbatasan, penggunaannya sangat disarankan ketika kita ingin melakukan operasi pada setiap elemen dalam kumpulan data tanpa melibatkan logika pengelolaan indeks atau iterator.

6. FUNGSI INFINITE LOOP

Infinite Loop adalah jenis loop yang tidak memiliki kondisi penghentian yang jelas, sehingga loop tersebut akan terus berjalan tanpa henti. Dalam pemrograman, infinite loop dapat terjadi ketika kondisi penghentian loop tidak tepat atau tidak didefinisikan dengan jelas.

```cpp
while (true) {
    // kode yang akan dijalankan secara
        berulang
}
```

Gambar 5.21 penulisan infinite looping

Dalam contoh di atas, loop while tidak memiliki kondisi penghentian yang jelas, sehingga loop tersebut akan terus berjalan tanpa henti.

Dampak Infinite Loop

1. Menghambat Sistem

Infinite loop dapat menghambat sistem karena memakan sumber daya CPU dan memori secara berlebihan.

2. Mengganggu Proses

Infinite loop dapat mengganggu proses lain yang sedang berjalan, sehingga dapat menyebabkan sistem menjadi tidak stabil.

3. Mengakibatkan Error

Infinite loop dapat mengakibatkan error karena memungkinkan sistem menjadi tidak stabil dan mengganggu proses lain.

Cara Menghindari Infinite Loop

1. Menggunakan Kondisi Penghentian Jelas

Pastikan bahwa kondisi penghentian loop jelas dan tepat.

2. Menggunakan Loop yang Tepat

Pilih loop yang tepat untuk tujuan yang diinginkan.

3. Menguji Kode

Jika Anda tidak yakin tentang kode Anda,uji kode Anda untuk memastikan bahwa tidak ada infinite loop.

Dengan demikian, infinite loop adalah hal yang perlu dihindari dalam pemrograman karena dapat menghambat sistem, mengganggu proses, dan mengakibatkan error. Oleh karena itu, pastikan bahwa Anda menggunakan loop yang tepat dan menguji kode Anda untuk memastikan bahwa tidak ada infinite loop.

7. FUNGSI CONDITIONAL LOOP

Conditional loop adalah struktur pengulangan dalam pemrograman yang terus menjalankan serangkaian instruksi sampai kondisi tertentu terpenuhi. Loop ini bergantung pada kondisi yang ditentukan untuk menentukan apakah blok kode dalam loop akan terus dieksekusi atau tidak. Conditional loop sangat berguna untuk mengulangi instruksi berdasarkan kondisi tertentu, seperti validasi input pengguna atau pengulangan hingga kondisi tertentu tercapai.

Karakteristik Conditional Loop:

1. Pengulangan Bedasarkan Kondisi

Conditional loop akan terus berjalan selama kondisi yang ditentukan masih benar.

2. Berhenti saat Kondisi Tidak Terpenuhi

Loop akan berhenti saat kondisi yang ditentukan tidak lagi benar.

3. Validasi Input

Conditional loop sering digunakan untuk validasi input pengguna atau pengulangan berdasarkan kondisi tertentu.

Metode Pembuatan Conditional Loop:

1. Pre-test Loop

Loop akan mengevaluasi kondisi sebelum menjalankan blok kode di dalamnya.

2. Post-test Loop

Loop akan menjalankan blok kode di dalamnya setidaknya sekali sebelum mengevaluasi kondisi untuk keluar dari loop.

```
int number = 0;
while (number < 10) {
    std::cout << "Enter a number: "
    std::cin >> number;
}
```

Gambar 5.22 penulisan conditional looping

Dalam contoh di atas, while loop digunakan sebagai conditional loop untuk meminta pengguna memasukkan

angka hingga angka yang dimasukkan lebih besar atau sama dengan 10.

Keuntungan Penggunaan Conditional Loop:

1. Validasi Input

Conditional loop berguna untuk validasi input pengguna.

2. Pengulangan Berdasarkan Kondisi

Loop akan berjalan berdasarkan kondisi yang ditentukan.

3. Fleksibilitas

Conditional loop memungkinkan pengulangan berdasarkan kondisi yang spesifik.

Dengan pemahaman yang baik tentang conditional loop, Anda dapat menggunakan loop ini untuk melakukan pengulangan berdasarkan kondisi tertentu dengan efisien dan efektif. Loop ini sangat berguna dalam situasi di mana Anda perlu melakukan pengulangan berdasarkan kondisi yang spesifik dan berhenti saat kondisi tersebut tidak lagi terpenuhi.

8. FUNGSI LOOP WITH BREAK

Loop with Break adalah konsep dasar dalam pemrograman yang memungkinkan pengulangan suatu set instruksi berulang kali hingga kondisi tertentu terpenuhi. Loop ini sangat berguna untuk

mengotomatisasi proses pengulangan yang berulang-ulang, seperti pengulangan hingga kondisi tertentu tercapai atau pengulangan berdasarkan kondisi tertentu.

Fungsi Loop with Break

1. Mengotomatisasi Proses Pengulangan

Loop with Break memungkinkan pengulangan suatu set instruksi berulang kali hingga kondisi tertentu terpenuhi.

2. Mengurangi Kode yang Sama Berulang

Dengan menggunakan loop, Anda tidak perlu menulis kode yang sama berulang kali.

3. Menghemat Waktu

Loop with Break memungkinkan Anda untuk menghemat waktu dengan mengotomatisasi proses pengulangan.

Cara Menggunakan Loop with Break

1. Pilih Jenis Loop yang Sesuai

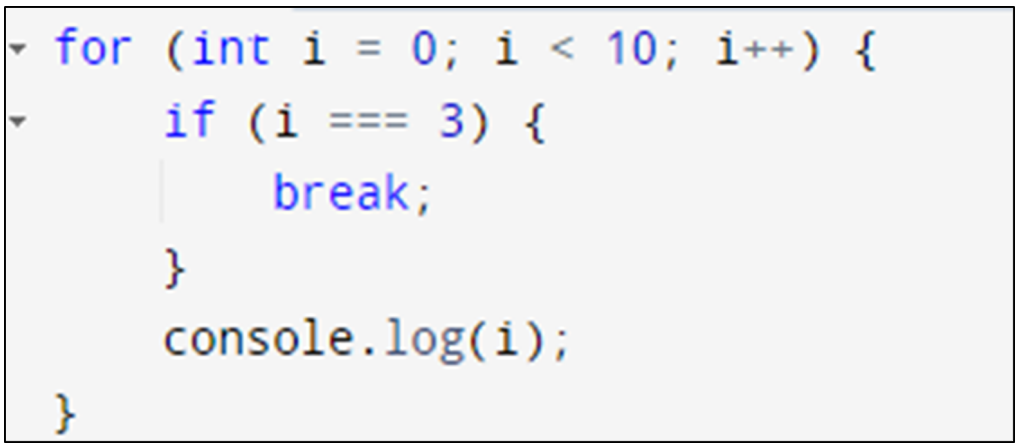
Pilih jenis loop yang sesuai dengan kebutuhan Anda.

2. Menggunakan Break Statemen

Menggunakan break statement untuk menghentikan loop ketika kondisi tertentu terpenuhi.

3. Menggunakan Continue Statement

Menggunakan continue statement untuk melompati satu iterasi dalam loop.



Gambar 5.23 penulisan continue statement

Dalam contoh di atas, loop for digunakan untuk mengulangi instruksi dari 0 sampai 9. Namun, ketika i mencapai 3, loop akan berhenti dengan menggunakan break statement.

Keuntungan Penggunaan Loop with Break

1. Mengotomatisasi Proses Pengulangan

Loop with Break memungkinkan pengulangan suatu set instruksi berulang kali hingga kondisi tertentu terpenuhi.

2. Mengurangi Kode yang Sama Berulang

Dengan menggunakan loop, Anda tidak perlu menulis kode yang sama berulang kali.

3. Menghemat Waktu

Loop with Break memungkinkan Anda untuk

menghemat waktu dengan mengotomatisasi proses pengulangan.

Dengan demikian, Loop with Break adalah konsep dasar dalam pemrograman yang memungkinkan pengulangan suatu set instruksi berulang kali hingga kondisi tertentu terpenuhi. Loop ini sangat berguna untuk mengotomatisasi proses pengulangan yang berulang-ulang, seperti pengulangan hingga kondisi tertentu tercapai atau pengulangan berdasarkan kondisi tertentu.

9. FUNGSI LOOP WITH CONTINUE

Loop with Continue adalah konsep dalam pemrograman yang memungkinkan pengulangan suatu set instruksi berulang kali hingga kondisi tertentu terpenuhi. Namun, ketika kondisi tertentu terpenuhi, loop tidak akan berhenti, melainkan akan melompati satu iterasi dan melanjutkan ke iterasi berikutnya. Dengan menggunakan continue statement, Anda dapat mengatur loop untuk melewati satu iterasi tertentu dan melanjutkan ke iterasi berikutnya tanpa mempengaruhi jalannya loop secara keseluruhan.

Fungsi Loop with Continue

1. Mengotomatisasi Proses Pengulangan

Loop with Continue memungkinkan pengulangan

suatu set instruksi berulang kali hingga kondisi tertentu terpenuhi.

2. Mengurangi Kode yang Sama Berulang

Dengan menggunakan loop, Anda tidak perlu menulis kode yang sama berulang kali.

3. Menghemat Waktu

Loop with Continue memungkinkan Anda untuk menghemat waktu dengan mengotomatisasi proses pengulangan.

Cara Menggunakan Loop with Continue

1. Pilih Jenis Loop yang Sesuai

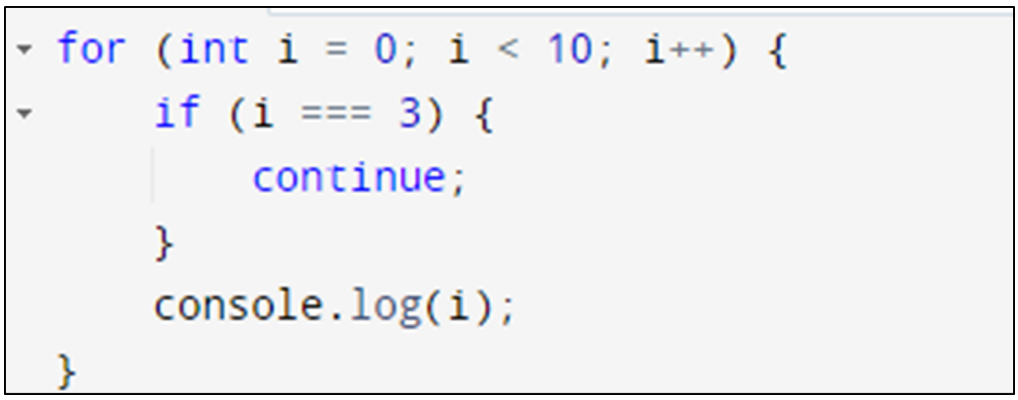
Pilih jenis loop yang sesuai dengan kebutuhan Anda.

2. Menggunakan Continue Statement

Menggunakan continue statement untuk melompati satu iterasi dalam loop.

3. Menggunakan Break Statement

Menggunakan break statement untuk menghentikan loop ketika kondisi tertentu terpenuhi.



Gambar 5.24 penulisan break statement

Dalam contoh di atas, loop for digunakan untuk mengulangi instruksi dari 0 sampai 9. Namun, ketika i mencapai 3, loop akan melompati iterasi tersebut dan melanjutkan ke iterasi berikutnya dengan menggunakan continue statement.

Keuntungan Penggunaan Loop with Continue

1. Mengotomatisasi Proses Pengulangan

Loop with Continue memungkinkan pengulangan suatu set instruksi berulang kali hingga kondisi tertentu terpenuhi.

2. Mengurangi Kode yang Sama Berulang

Dengan menggunakan loop, Anda tidak perlu menulis kode yang sama berulang kali.

3. Menghemat Waktu

Loop with Continue memungkinkan Anda untuk

menghemat waktu dengan mengotomatisasi proses pengulangan.

Dengan demikian, Loop with Continue adalah konsep dalam pemrograman yang memungkinkan pengulangan suatu set instruksi berulang kali hingga kondisi tertentu terpenuhi, namun dengan kemampuan untuk

melewati satu iterasi tertentu dan melanjutkan ke iterasi berikutnya tanpa mempengaruhi jalannya loop secara keseluruhan.

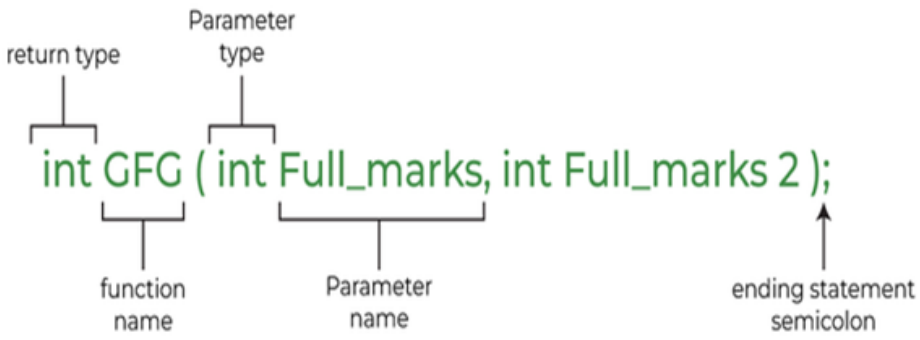
Dalam kesimpulan, penggunaan looping adalah teknik yang penting dan sering digunakan dalam pemrograman untuk menjalankan serangkaian instruksi secara berulang berdasarkan kondisi atau jumlah iterasi tertentu. Dengan demikian, looping adalah konsep dasar dalam pemrograman yang sangat berguna untuk mempermudah pengerjaan program dan untuk mempersingkat instruksi program. Namun, perlu diingat bahwa looping juga dapat menyebabkan kesalahan jika tidak digunakan dengan benar. Oleh karena itu, penting untuk memahami konsep looping dan bagaimana mengimplementasikannya dengan benar.

6
Function

FUNCTION

1.0 FUNCTION DALAM C++

Fungsi adalah sekumpulan pernyataan yang mengambil masukan, melakukan perhitungan tertentu, dan menghasilkan keluaran. Idenya adalah untuk menggabungkan beberapa tugas yang umum atau berulang kali dilakukan untuk membuat suatu fungsi sehingga alih-alih menulis kode yang sama berulang kali untuk masukan yang berbeda, kita dapat memanggil fungsi ini. Secara sederhana, suatu fungsi merupakan suatu blok kode yang dapat berjalan ketika dipanggil.

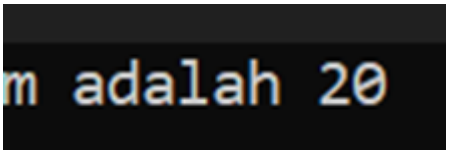


Gambar 6.1 konsep function dalam pemrograman

Contoh Program :

```
1   #include <iostream>
2   using namespace std;
3
4   int max(int x,int y)
5   {
6       if(x>y)
7       return x;
8       else
9       return y;
10  }
11
12  int main()
13    {
14      int a =10, b=20;
15      int m= max(a,b);
16      cout<<"m adalah "<<m;
17      return 0;
18    }
```

Gambar 6.2 contoh penulisan function



Gambar 6.3 hasil penulisan function

1.1 MENGAPA KITA PERLU MEMBUTUHKAN FUNGSI?

- Fungsi membantu kami mengurangi redundansi kode . Jika fungsionalitas dijalankan di beberapa tempat dalam perangkat lunak, maka daripada menulis kode yang sama, berulang kali, kita membuat fungsi dan memanggilnya di mana saja. Ini juga membantu dalam pemeliharaan karena kami harus melakukan perubahan hanya di satu tempat jika kami melakukan perubahan pada fungsionalitas di masa mendatang.

- Fungsi membuat kode menjadi modular . Bayangkan sebuah file besar yang memiliki banyak baris kode. Membaca dan menggunakan kode menjadi sangat mudah jika kode tersebut dibagi menjadi beberapa fungsi.

- Fungsi menyediakan abstraksi . Misalnya, kita dapat menggunakan fungsi perpustakaan tanpa mengkhawatirkan pekerjaan internalnya.

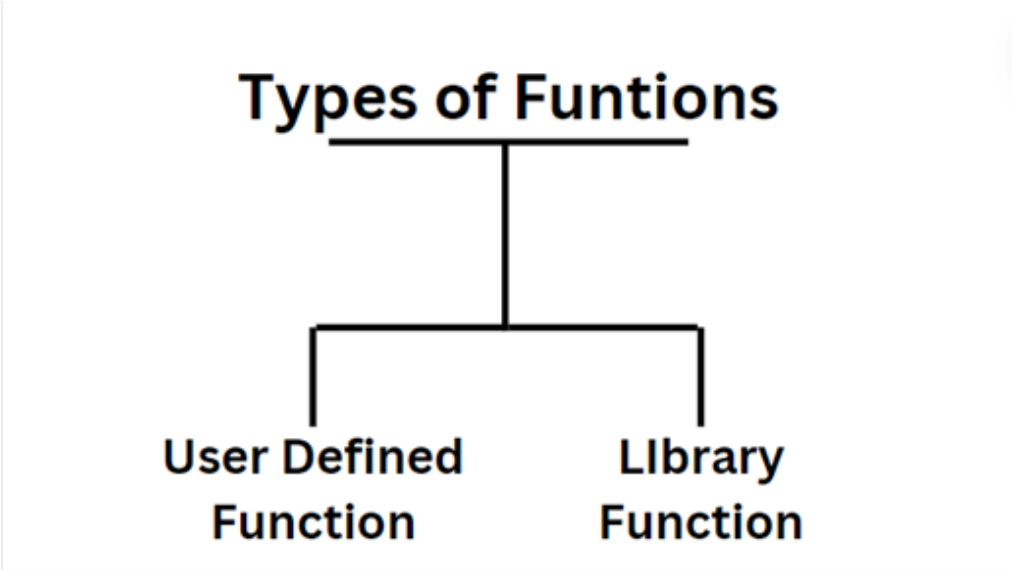
1.2 DEKLARASI FUNGSI

Deklarasi fungsi memberi tahu kompiler tentang jumlah parameter, tipe data parameter, dan tipe fungsi yang dikembalikan. Penulisan nama parameter dalam deklarasi fungsi bersifat opsional tetapi perlu untuk memasukkannya ke dalam definisi. Di bawah ini adalah contoh deklarasi fungsi. (nama parameter tidak ada dalam deklarasi di bawah)

```
// C++ Program to show function that takes
// two integers as parameters and returns
// an integer
int max(int, int);

// A function that takes an int
// pointer and an int variable
// as parameters and returns
// a pointer of type int
int* swap(int*, int);

// A function that takes
// a char as parameter and
// returns a reference variable
char* call(char b);

// A function that takes a
// char and an int as parameters
// and returns an integer
int fun(char, int);
```

Gambar 6.4 penulisan deklarasi function

1.3 JENIS FUNGSI



Gambar 6.5 gambar jenis function

- FUNGSI BUATAN PENGGUNA

Fungsi yang ditentukan pengguna adalah blok kode yang ditentukan pengguna/pelanggan yang disesuaikan secara khusus untuk mengurangi kompleksitas program besar. Mereka juga umumnya dikenal sebagai “ fungsi yang dibuat khusus ” yang dibangun hanya untuk memenuhi kondisi di mana pengguna menghadapi masalah sekaligus mengurangi kompleksitas keseluruhan program.

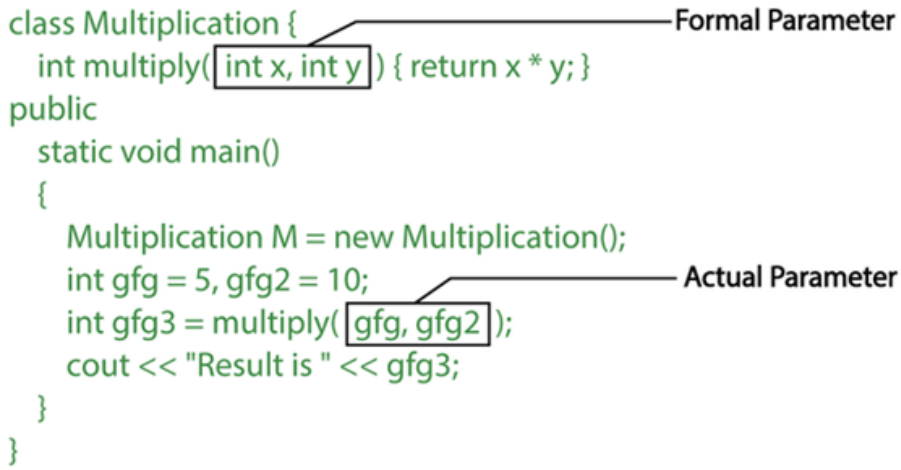
- FUNGSI PERPUSTAKAAN

Fungsi perpustakaan juga disebut “ Fungsi bawaan ”. Fungsi-fungsi ini merupakan bagian dari paket compiler yang sudah terdefinisi dan terdiri dari fungsi khusus dengan arti khusus dan berbeda. Fungsi Bawaan memberi kita keunggulan karena kita dapat langsung menggunakannya tanpa mendefinisikannya sedangkan dalam fungsi yang ditentukan pengguna kita harus mendeklarasikan dan mendefinisikan suatu fungsi sebelum menggunakannya. Misalnya: sqrt(), setw(), strcat(), dll.

Pembahasan II.

2.0 PARAMATER MENERUSKAN KE FUNGSI

Parameter yang diteruskan ke fungsi disebut parameter aktual . Misalnya, pada program di bawah ini, 5 dan 10 adalah parameter sebenarnya. Parameter yang diterima oleh suatu fungsi disebut parameter formal . Misalnya, pada program di atas x dan y adalah parameter formal.Misalnya, pada program di atas x dan y adalah parameter formal



Gambar 6.6 parameter pada function

ADA DUA CARA YANG PALING POPULAR UNTUK MENERUSKAN PARAMATER:

1. Lewati Nilai: Dalam metode penerusan parameter ini, nilai parameter aktual disalin ke parameter formal fungsi. Parameter aktual dan formal disimpan di lokasi memori yang berbeda sehingga setiap perubahan yang dilakukan pada fungsi tidak tercermin dalam parameter aktual pemanggil.

2. Lewati Referensi: Parameter aktual dan formal merujuk ke lokasi yang sama, sehingga setiap perubahan yang dilakukan di dalam fungsi akan tercermin dalam parameter aktual pemanggil.

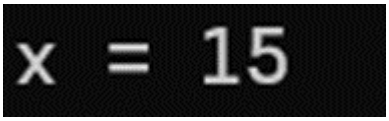
DEFINISI FUNGSI

Nilai pass by digunakan jika nilai x tidak diubah menggunakan fungsi fun().

```
#include <iostream>
using namespace std;

void fun(int x)
{
    x= 25;
}
int main ()
{
    int x = 15;
    fun(x);
    cout << "x = "<< x;
    return 0;
}
```

Gambar 6.7 penulisan function dalam C++

HASIL:



Gambar 6.8 hasil program function

FUNGSI MENGGUNAKAN POINTER

Fungsi fun() mengharapkan penunjuk ptr ke bilangan bulat (atau alamat bilangan bulat). Ini mengubah nilai di alamat ptr. Operator dereferensi * digunakan untuk mengakses nilai pada suatu alamat. Pernyataan pada '*ptr = 30', nilai yang ada dalam alamat "ptr" diubah menjadi 30. Operator alamat & digunakan untuk mendapatkan alamat variabel bertipe data apa pun. Dalam pernyataan pemanggilan fungsi 'fun(&x)', alamat

x dilewatkan sehingga x dapat dimodifikasi menggunakan alamatnya.

```
#include <iostream>
using namespace std;

void fun(int* ptr) {*ptr = 30; }

int main()
{
    int x =10;
    fun(&x);
    cout<< "x = " << x;

    return 0;
}
```

Gambar 6.9 penulisan function menggunakan pointer

HASIL:

```
x = 30
```

Gambar 6.10 hasil penulisan function menggunakan pointer

PERBEDAAN ANTARA PANGGILAN BERDASARKAN NILAI DENGAN REFRENSI DI DALAM C++

PANGGILAN BERDASARKAN NILAI

PANGGILAN DENGAN REFERENSI

Salinan nilai diteruskan ke fungsi

Alamat nilai ditegurkan ke fungsi

Perubahan yang dilakukan di fungsi tidak terceminkan pada fungsi lainnya

Perubahan yang dilakukan dalam fungsi juga tercerminkan di luar fungsi

Argumen actual dan formal akan dibuat di dalam lokasi memori yang berbeda

Argumen actual dan formal akan dibuat di lokasi memori yang sama

PANGGILAN BERDASARKAN NILAI	PANGGILAN DENGAN REFERENSI
Salinan nilai diteruskan ke fungsi	Alamat nilai ditegurkan ke fungsi
Perubahan yang dilakukan di fungsi tidak terceminkan pada fungsi lainnya	Perubahan yang dilakukan dalam fungsi juga tercerminkan di luar fungsi
Argumen actual dan formal akan dibuat di dalam lokasi memori yang berbeda	Argumen actual dan formal akan dibuat di lokasi memori yang sama

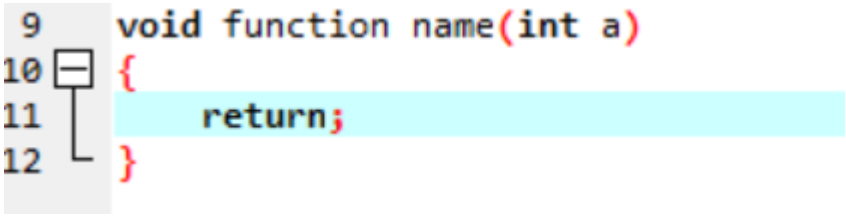
Gambar 6.11 tabel panggilan berdasarkan nilai dan dengan refrensi

POIN YANG PERLU DIINGAT TENTANG FUNGSI DI DALAM C++

1. Kebanyakan program C++ memiliki fungsi bernama main() yang dipanggil oleh sistem operasi saat pengguna menjalankan program.

2. Setiap fungsi memiliki tipe kembalian. Jika suatu fungsi tidak mengembalikan nilai apa pun, maka void digunakan sebagai tipe pengembalian. Selain itu, jika tipe kembalian dari fungsi tersebut tidak berlaku, kita

masih dapat menggunakan pernyataan return dalam isi definisi fungsi dengan tidak menentukan konstanta, variabel, dll., dengan hanya menyebutkan 'return;' pernyataan yang melambangkan penghentian fungsi seperti yang ditunjukkan di bawah ini:



Gambar 6.12 penulisan function void

3. Untuk mendeklarasikan fungsi yang hanya bisa dipanggil tanpa parameter apa pun, kita harus menggunakan " void fun(void) ". Sebagai catatan tambahan, dalam C++, daftar kosong berarti suatu fungsi hanya dapat dipanggil tanpa parameter apa pun. Di C++, void fun() dan void fun(void) sama.

FUNGSI UTAMA

Fungsi utama merupakan fungsi khusus. Setiap program C++ harus berisi fungsi bernama main. Ini berfungsi sebagai titik masuk untuk program ini. Komputer akan mulai menjalankan kode dari awal fungsi utama.

JENIS FUNGSI

1. Tanpa Paramater

```
int main()
{ ...return 0;}
```

Gambar 6.13 penulisan function tanpa parameter

2. Dengan Paramater

```
int main(int argc,char* const argv[])
{... return 0; }
```

Gambar 6.14 penulisan parameter dengan function

Alasan memiliki opsi parameter untuk fungsi utama adalah untuk mengizinkan input dari baris perintah. Saat Anda menggunakan fungsi utama dengan parameter, ia menyimpan setiap grup karakter (dipisahkan dengan spasi) setelah nama program sebagai elemen dalam array bernama argc . Karena fungsi utama memiliki tipe pengembalian int , pemrogram harus selalu memiliki pernyataan pengembalian dalam kodenya. Nomor yang dikembalikan digunakan untuk menginformasikan program pemanggil mengenai hasil eksekusi program tersebut. Mengembalikan 0 menandakan bahwa tidak ada masalah.

REKURSI C++

Ketika fungsi dipanggil dalam fungsi yang sama, ini dikenal sebagai rekursi dalam C++. Fungsi yang memanggil fungsi yang sama disebut fungsi rekursif. Fungsi yang memanggil dirinya sendiri, dan tidak melakukan tugas apa pun setelah pemanggilan fungsi, dikenal sebagai rekursi ekor. Dalam rekursi ekor, kita biasanya memanggil fungsi yang sama dengan pernyataan return.

```
recursionfunction()
{
    recursionfunction(); // calling self function
}
```

Gambar 6.15 penulisan pemanggilan function di function itu sendiri

KEBUTUHAN DARI REKURSI

Rekursi adalah teknik luar biasa yang dengannya kita dapat mengurangi panjang kode dan membuatnya lebih mudah untuk dibaca dan ditulis. Memiliki keunggulan tertentu dibandingkan teknik iterasi, rekursi adalah salah satu solusi terbaik untuk itu. Misalnya; Faktorial suatu bilangan.

INTERPRESTASI MATEMATIKA

Terdapat berbagai metode untuk menentukan jumlah n bilangan asli pertama, namun pendekatan paling mudah adalah dengan menjumlahkan bilangan dari 1 hingga n. Dengan begitu, fungsinya akan terlihat seperti ini.

```
pendekatan(1) – Cukup tambahkan satu per satu

f(n) = 1 + 2 + 3 +.....+ n
```

Gambar 6.16 metode interprestasi matematika

Lalu ada pendekatan matematis lain yang dapat mewakili hal seperti ini,

```
pendekatan(2) – Penambahan rekursif

f(n) = 1 n=1

f(n) = n + f(n-1) n>1
```

Gambar 6.17 metode interprestasi matematika

Pentingnya perbedaan antara pendekatan (1) dan (2) terletak pada fakta bahwa dalam pendekatan (2), fungsi “f()” dipanggil kembali di dalam dirinya sendiri, yang disebut sebagai rekursi. Fenomena ini membentuk dasar dari apa yang dikenal sebagai fungsi rekursif, yang merupakan alat yang kuat bagi para pemrogram untuk menyelesaikan masalah dengan cara yang lebih sederhana dan efisien.

KONDISI DASAR DALAM REKURSI

Konsep utama dalam pemrograman rekursif adalah menyediakan solusi untuk kasus dasar, lalu mengungkapkan solusi untuk masalah yang lebih besar dalam istilah masalah yang lebih kecil.

```
ke dalam fakta(int n)
{
    if (n < = 1) // kasus dasar
        kembali 1;
    kalau tidak
        kembalikan n*fakta(n-1);
}
```

Gambar 6.18 konsep kondisi dasar rekursi

Pada contoh sebelumnya, kita mendefinisikan kasus dasar ketika n ≤ 1, dan untuk menyelesaikan nilai yang lebih besar dari suatu bilangan, kita mengurai masalah itu menjadi nilai yang lebih kecil sampai kita mencapai kasus dasar

BAGAIMANA SUATU MASALAH DAPAT DISELESAIKAN DENGAN REKURSI.

Dalam rekursi, kita memecah masalah menjadi masalah yang lebih kecil dan menetapkan kondisi dasar yang akan menghentikan rekursi. Sebagai contoh, untuk menghitung faktorial dari n, kita menggunakan faktorial dari (n-1). Kasus dasar dalam perhitungan faktorial adalah ketika n = 0, di mana kita mengembalikan nilai 1.

CARA MEMANGGIL FUNGSI YANG BERBEDA DALAM REKURSI.

Ketika suatu fungsi dipanggil dari main(), memori dialokasikan padanya di stack. Fungsi rekursif memanggil dirinya sendiri, memori untuk fungsi yang dipanggil dialokasikan di atas memori yang dialokasikan untuk fungsi pemanggil dan salinan variabel lokal yang berbeda dibuat untuk setiap pemanggilan fungsi. Ketika kasus dasar tercapai, fungsi mengembalikan nilainya ke fungsi yang memanggilnya dan memori dibatalkan alokasinya dan proses dilanjutkan.

```
#include <bits/stdc++.h>
using namespace std;

void printFun(int test)
{
    if (test < 1)
        return;
    else {
        cout << test << " ";
        printFun(test - 1);
        cout << test << " ";
        return;

    }
}

int main()
{
    int test = 3;
    printFun(test);
}
```

Gambar 6.19 cara pemanggilan function di class Main

HASIL:

```
3 2 1 1 2 3
```

Gambar 6.20 hasil pemanggilan function di class ma

Ketika printFun(3) dipanggil dari main(), memori dialokasikan ke printFun(3) dan pengujian variabel lokal diinisialisasi ke 3 dan pernyataan 1 hingga 4 dimasukkan ke dalam tumpukan seperti yang ditunjukkan dalam

diagram di bawah. Ini pertama kali mencetak ‘3’. Dalam pernyataan 2, printFun(2) dipanggil dan memori dialokasikan ke printFun(2) dan pengujian variabel lokal diinisialisasi ke 2 dan pernyataan 1 hingga 4 dimasukkan ke dalam tumpukan. Demikian pula, printFun(2) memanggil printFun(1) dan printFun(1) memanggil printFun(0) . printFun(0) menuju ke pernyataan if dan kembali ke printFun(1) . Pernyataan printFun(1) yang tersisa dieksekusi dan kembali ke printFun(2) dan seterusnya. Pada output, nilai dari 3 hingga 1 dicetak dan kemudian 1 hingga 3 dicetak.

PARAMETER FUNCTION

Melewati fungsi sebagai argumen adalah konsep yang berguna dalam C++. Konsep ini telah digunakan saat meneruskan fungsi pembanding khusus sebagai argumen di std::sort() untuk mengurutkan urutan objek sesuai kebutuhan. Pada artikel ini, kita akan membahas berbagai cara mendesain fungsi yang menerima fungsi lain sebagai argumen.

SUATU FUNCTION BISA DITERUSKAN SEBAGAI PARAMETER DENGAN MENGGUNAKAN 3 PENDEKATAN:

- Lulus sebagai Pointer
- Menggunakan std;fungsi <>
- Menggunakan Lamda

1. MENERUSKAN POINTER KE SUATU FUNGSI.

Suatu fungsi juga dapat diteruskan ke fungsi lain dengan meneruskan alamatnya ke fungsi tersebut; Secara sederhana, hal ini dapat dicapai melalui pointer.

Contoh Program:

```
1   #include <iostream>
2   using namespace std;
3
4   int add(int x, int y) {return x + y;
5   }
6   int multiply(int x, int y) {return x * y;
7   }
8   int invoke( int x, int y, int (*func)(int, int))
9   {
10      return func(x,y);
11  }
12  int main()
13  {
14      cout<< "Penambahan 20 dan 10 adalah ";
15      cout<< invoke(20,10,&add)<<'\n';
16
17      cout<<"perkalian  20"
18          <<" dan 10 adalah ";
19      cout<< invoke(20,10, &multiply) <<'\n';
20      return 0;
21  }
```

Gambar 6.21 penulisan pemanggilan pointer ke dalam function

HASIL:

```
Penambahan 20 dan 10 adalah 30
perkalian  20 dan 10 adalah 200
```

Gambar 6.22 hasil pemanggilan pointer ke dalam function

2. MENGGUNAKAN STD;FUNCTION<>

Di C++ terdapat std;function<> yang memungkinkan untuk meneruskan fungsi sebagai objek. Objek std;function<> dapat dibuat seperti ini:

Contoh Program :

```
#include <functional>
#include <iostream>
using namespace std;

int invoke(int x, int y, function<int(int, int)> func)
{
    return func(x, y);
}

int main()
{
    cout << "Addition of 20 and 10 is ";
    int k = invoke(20, 10, [](int x, int y) -> int {
        return x + y;
    });
    cout << k << '\n';

    cout << "Multiplication of 20 and 10 is ";
    int l = invoke(20, 10, [](int x, int y) -> int {
        return x * y;
    });
    cout << l << '\n';

    return 0;
}
```

Gambar 6.23 penulisan std;function<>

HASIL:

```
Penambahan 20 dan 10 adalah 30
Perkalian 20 dan 10 adalah 200
```

Gambar 6.24 hasil penulisan std;function<>

3. MENGGUNAKAN LAMDA

Lambdas di C++ menyediakan cara untuk mendefinisikan objek fungsi anonim satu kali saja. Lambda ini dapat didefinisikan di tempat yang mengharuskan meneruskan fungsi sebagai argumen.

Contoh Program :

```
#include <functional>
#include <iostream>
using namespace std;

int invoke(int x, int y
function<int(int,int)> func)
{
    return func(x, y);
}

int main()
{
    cout << "addition of 20 and 10 is ";
    int k = invoke(20, 10,[](int x, int y) -> int{
    return x + y;});

    cout<< k << '\n';

    cout<< " Multiplication of 20"
    <<" and 10 is ";
    int 1 = invoke (20, 10, [](int x, int y) -> int {
    return x * y;});

    cout << 1 << '\n';
    return 0;
}
```

Gambar 6.25 penulisan function dengan lamda

HASIL:

```
Penambahan 20 dan 10 adalah 30
Perkalian 20 dan 10 adalah 200
```

Gambar 6.26 hasil penulisan function dengan lamda

Pembahasan III.

RETURN FUNGSI

RETURN DALAM C++

Pernyataan return mengembalikan aliran eksekusi ke fungsi tempat ia dipanggil. Pernyataan ini tidak wajib memerlukan pernyataan bersyarat apa pun. Segera setelah pernyataan dijalankan, aliran program segera berhenti dan mengembalikan kontrol dari tempat pemanggilannya. Pernyataan return mungkin atau mungkin tidak mengembalikan apa pun untuk fungsi void, tetapi untuk fungsi non-void, nilai kembalian harus dikembalikan.

Lalu Ada Berbagai Cara Untuk Menggunakan Pernyataan return:

1. Metode tidak mengembalikan nilai.

Ketika suatu fungsi tidak mengembalikan apa pun, tipe pengembalian void digunakan. Jadi jika ada tipe pengembalian void dalam definisi fungsi, maka tidak akan ada pernyataan return di dalam fungsi tersebut (secara umum).

Contoh Program :

```
#include <iostream>
using namespace std;

void Print()
{
    cout << "Welcome to SIEGA Unika Soegijapranata ";
}

int main ()
{
    Print();

    return 0;
}
```

Gambar 6.27 penulisan function dengan return 0

HASIL:

```
Welcome to SIEGA Unika Soegijapranata
```

Gambar 6.28 hasil penulisan function dengan return 0

2. Metode mengembaliikan nilai.

Metode yang mendefinisikan tipe kembalian, pernyataan return harus segera diikuti dengan nilai kembalian dari tipe kembalian yang ditentukan.

Contoh Program :

```
1   #include <iostream>
2   using namespace std;
3
4   int SUM(int a, int b)
5   {
6       int s1 = a + b;
7
8       return s1;
9   }
10  int main()
11  {
12      int num1 = 10;
13      int num2 = 10;
14      int sum_of = SUM(num1, num2);
15      cout << "The sum is " << sum_of;
16      return 0;
17  }
```

Gambar 6.29 penulisan mengembalikan nilai

HASIL:

```
The sum is 20
```

Gambar 6.30 hasil penulisan function yang mengembalikan nilai

Pembahasan IV.

ARRAY KE FUNGSI

ARRAY

Array adalah struktur data yang digunakan untuk menyimpan beberapa nilai dari tipe data serupa di lokasi memori yang berdekatan.jika kita harus menyimpan nilai 4 atau 5 siswa, maka kita dapat dengan mudah menyimpannya dengan membuat 5 variabel berbeda, tetapi bagaimana jika kita ingin menyimpan nilai 100 siswa atau katakanlah 500 siswa, maka akan menjadi sangat sulit untuk membuat angka tersebut variabel dan mengelolanya. Sekarang, array muncul dalam gambaran yang dapat melakukannya dengan mudah hanya dengan membuat array dengan ukuran yang diperlukan.

Properti Yang Digunakan Di Array C++

- Array adalah sekumpuan data dengan tipe data yang sama, dan disimpan didalam lokasi memori yang berdekatan.
- Pengindeksan suatu array dimulai dari 0. Dengan artian elemen pertama yang disimpan pada indeks ke-

0,elemen kedua pada indeks ke-1, dan seterusnya.

- Elemen indeks dapat diakses dengan menggunakan indeksnya.
- Setelah array dideklarasikan, ukurannya tetap konstan selama program berlangsung.
- Sebuah array dapat memiliki banyak dimensi.
- Jumlah elemen dalam array dapat ditentukan menggunakan operator sizeof.
- Kita bisa menemukan ukuran tipe elemen yang disimpan dalam array dengan mengurangkan alamat yang berdekatan.

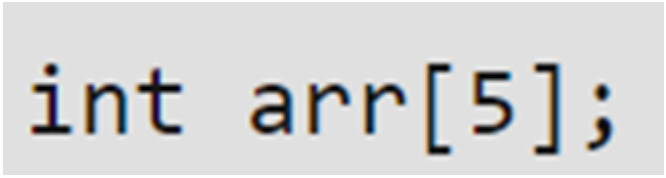
Deklarasi Array C++

Di C++, kita dapat mendeklarasikan array hanya dengan menentukan tipe datanya terlebih dahulu, lalu nama array beserta ukurannya.

```
tipe_data nama_array[Ukuran_array];
```

Gambar 6.31 deklarasi array dalam function

Dengan contoh :



Gambar 6.32 penulisan deklarasi array dalam function

Penjelasan :

- int: Ini adalah tipe data yang akan disimpan dalam array. Kita juga bisa menggunakan tipe data lain seperti char, float, dan double.
- arr: Ini adalah nama arraynya.
- 5: Ini adalah ukuran array yang berarti hanya 5 elemen yang dapat disimpan dalam array.

Mengakses Elemen Array C++

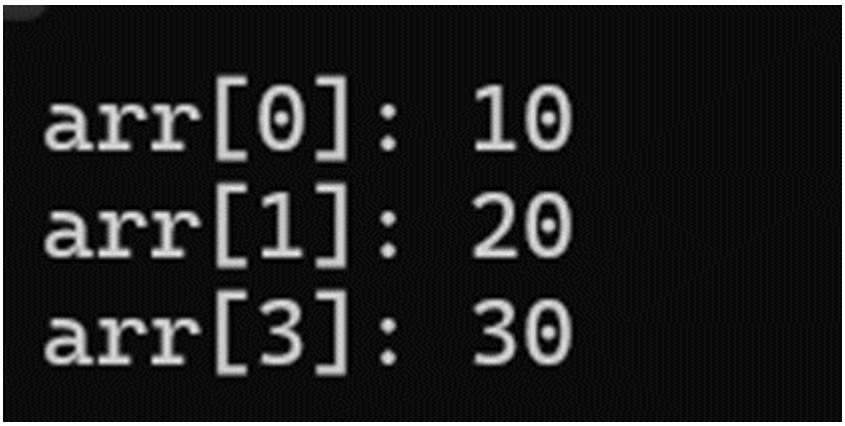
Elemen suatu array dapat diakses dengan menentukan nama array, kemudian indeks elemen tersebut diapit oleh operator subskrip array []. Misalnya, arr[i].

Contoh Program :

```cpp
#include <iostream>
using namespace std;

int main()
{
    int arr[3];

    arr[0] = 10;
    arr[1] = 20;
    arr[2] = 30;

    cout<< "arr[0]: " << arr [0] << endl;
    cout<< "arr[1]: " << arr [1] << endl;
    cout<< "arr[3]: " << arr [2] << endl;

    return 0;
}
```

Gambar 6.33 penulisan untuk mengakses elemen array

HASIL :



Gambar 6.34 hasil penulisan untuk mengakses elemen array

Perbarui Elemen Array

Untuk memperbarui elemen dalam array, kita dapat menggunakan indeks yang ingin kita perbarui yang disertakan dalam operator subskrip array dan menetapkan nilai baru.

```
arr[i] = nilai_baru;
```

Gambar 6.35 penulisan pembaruan array

Melintasi Array

Melintasi array dengan bantuan loop menggunakan pengindeksan di C++. Pertama, kita menginisialisasi array ‘table_of_two’ dengan kelipatan 2. Setelah itu, kita menjalankan loop for dari 0 hingga 9 karena dalam sebuah array pengindeksan dimulai dari nol. Oleh karena itu, dengan menggunakan indeks kami mencetak semua nilai yang disimpan dalam array.

Contoh Program :

```cpp
#include <iostream>
using namespace std;

int main()
{
    int table_of_two[10]
    ={ 2, 4, 6, 8, 10, 12, 14, 16. 18. 20};

    for (int i=0; i<10; i++){
        cout<< table_of_two[i]<< " ";
        }
    return 0;
}
```

Gambar 6.36 penulisan function melintasi array

HASIL:

```
2 4 6 8 10 12 14 16 18 20
```

Gambar 6.37 hasil penulisan function melintasi array

MENERUSKAN ARRAY KE FUNGSI C++

Melewati array ke fungsi dilakukan untuk melakukan berbagai operasi pada elemen array tanpa mengacaukan kode utama. Array dapat diteruskan dalam suatu fungsi menggunakan pointer atau referensi. Memahami berbagai pendekatan untuk meneruskan array penting untuk menulis kode sesuai kebutuhan.

Metode Melewati Array ke Fungsi di C++

- Sebagai array berukuran.
- Sebagai array yang tidak berukuran.
- Sebagai petunjuk.
- Sebagai referensi.

1. Melewati Sebagai Array Berukuran.

Meneruskan array dengan cara yang sama seperti mendeklarasikannya dengan tipe, nama, dan ukuran array. Seperti yang bisa dilihat, masih harus meneruskan ukuran array sebagai parameter lain karena pada akhirnya, array akan diperlakukan sebagai pointer dalam fungsi.

Contoh Program :

```
#include <iostream>
using namespace std;

void printarray(int a[10])
{
    for (int i = 0; i < 5; i++)
    a[i] = a[i] + 5;
}

int main()
{
    int a[5] = {1, 2, 3, 4, 5 };
    printarray(a);

    for (int i = 0; i < 5; i++)
    cout << a[i] << " ";
    return 0;
}
```

Gambar 6.38 penulisan function melewati sebagai array berukuran

HASIL:

Gambar 6.39 hasil penulisan function melewati sebagai array berukuran

2. Melewati Sebagai Array Tak Berukuran.

Metode ini sama dengan yang diatas. Tetapi yang membedakan hanya tidak menentukan ukuran arraynya.

Contoh Program :

```
1   #include <iostream>
2   using namespace std;
3
4   void printarray(int a[], int size)
5   {
6       for (int i = 0; i < size; i++)
7       a[i] = a[i] +5;
8
9   }
10
11  int main()
12  {
13      int a[5] ={1, 2, 3, 4, 5 };
14      int n=5;
15      printarray(a,n);
16
17      for (int i = 0; i < n; i++)
18      cout << a[i] << " ";
19       return 0;
20  }
```

Gambar 6.40 penulisan function melewati array yang beraturan

HASIL:

Gambar 6.41 hasil penulisan function melewati array yang beraturan

3. Melewati Array Sebagai Pointer.

Di dalam metode ini, kita meneruskan alamat memori elemen pertama array. Metode ini juga memungkinkan ukuran array dinamis.

Contoh Program :

```
1   #include <iostream>
2   using namespace std;
3
4   void printarray (int* a)
5   {
6       for(int i = 0; i < 5; i++)
7           *(a + i) = *(a + i) + 5;
8   }
9   int main()
10  {
11      int a[5] = {1, 2, 3, 4, 5 };
12      printarray(a);
13      for (int i = 0 ; 1 < 5; i++)
14      cout<< a[i] << " ";
15
16      return 0;
17  }
18
```

Gambar 6.42 penulisan function melewati array sebagai pointer

HASIL:

Gambar 6.43 hasil penulisan function melewati array sebagai pointer

4. Melewati Array Sebagai Referensi.

Meneruskan array ke suatu fungsi sebagai referensi dan juga secara eksplisit meneruskan ukurannya, Yakni meneruskan referensi ke elemen pertama dan ukuran array.

Contoh Program:

```
1   #include <iostream>
2   using namespace std;
3
4   void modifyArray (int (&arr)[5])
5   {
6       int size = sizeof(arr) / sizeof(int);
7       for (int i = 0; i<size; ++i){
8           arr[i] *=2;
9       }
10  }
11
12  int main()
13  {
14      int arr[] = {1, 2, 3, 4, 5 };
15      int n=5;
16      modifyArray(arr);
17      for (int i= 0; i < n; i++){ cout << arr[i] << " ";
18      }
19      return 0;
20  }
```

Gambar 6.44 penulisan function lewati array sebagai refrensi

HASIL;

Gambar 6.45 hasil penulisan function melewati array sebagai refrensi

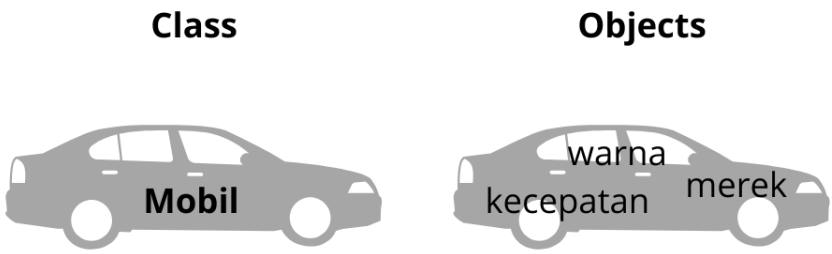
7

Object-Oriented Programming

1. Pengertian *Object-Oriented Programming*

Object-Oriented Programming atau sering disebut OOP adalah sebuah metode pemrograman yang menggunakan konsep objek sebagai unit dasar untuk membuat sebuah program perangkat lunak. Dalam OOP, objek itu seperti benda nyata didalam program yang memiliki ciri-ciri, misalnya bentuk dan warna, dan bisa melakukan sesuatu, misalnya bergerak. Objek ini dibuat berdasarkan model yang disebut Class / kelas. Class / kelas ini seperti pembentukan yang menentukan bagaimana objek itu akan terlihat dan apa yang bisa dilakukannya. OOP membantu dalam mengelola kode menjadi unit yang lebih kecil, terstruktur dan mudah dipahami. OOP juga mendukung konsep pengembangan perangkat lunak yang lebih terstruktur. Contohnya, jika kita memikirkan sebuah objek "mobil", kita bisa mendefinisikan Class bernama "Mobil" yang memiliki atribut seperti warna, merk, dan kecepatan, dan juga

perilaku seperti “maju”, “mundur”, atau “berhenti”.IstilahObjectdalamObject Oriented Programming (OOP) C++, sebenarnya terdiri dari Class, data member, member functiondanObject.



Gambar 7.1 gambar perumpamaan perbedaan class dan object

Dengan menggunakan OOP, kita dapat memodelkan entitas dunia nyata ke dalam program perangkat lunak dengan cara yang lebih terstruktur dan mudah dipahami. Hal ini tentu membantu dalam mengorganisasi kode menjadi bagian yang lebih kecil, sehingga memudahkan pengembangan, pemeliharaan dan memperluas program nantinya. Jadi, OOP adalah pendekatan dan

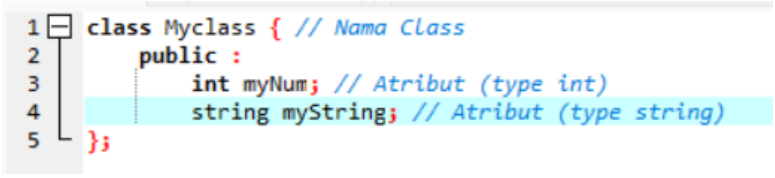
pengembangan perangkat lunak yang menggunakan konsep objek dan kelas untuk mengelola dan menstrukturkan kode secara efisien. Object-Oriented Programming terlihat sulit untuk di pahami, tetapi begitu kita paham konsep Class dan Object, maka kita akan mengerti bahwa betapa kuat dan berguna konsep OOP di dalam pengembangan perangkat lunak. Kita tidak hanya bisa meniru keadaan di dunia nyata, tetapi juga

A. Class/Object

Class/ Kelas adalah tempat yang berisi abstraksi dari suatu objek atau benda, yang mendeskripsikan data dan fungsi yang dimiliki oleh objek tersebut. Karena Class adalah tempat yang akan digunakan untuk menciptakan objek, jelas bahwa kita harus membuat Class terlebih dahulu sebelum membuat objek. Objek menjadi wujud nyata dari sebuah Class. Jika Class secara umum mempresentasikan sebuah objek, sebuah Instance adalah representasi nyata dari dari Class itu sendiri.

Didalam Object-Oriented Programming ada juga yang namanya Atribut. Atribut adalah data yang di simpan di dalam objek, kita bisa menganggap atribut ini sebagai variabel yang menentukan keadaan dari dari sebuah objek. Misalnya di dalam Class "Mobil", atributnya itu terdiri dari warna, merk, type dll.

Untuk cara membuat Class dan Attribut bisa dilihat gambar berikut ini :



Gambar 7.2 penulisan class dan attribute

Untuk cara pembuatan namaClassaturannya adalah huruf pertama adalah huruf kapital, contohMy,jika ada dua kata seperti, contohMy Classmaka kata kedua juga ditulis dengan huruf kapital dan tidak boleh ada spasi, menjadiMyClass.Kata kuncipublicadalah penentu akses, yang menentukan bahwa anggota (attributedanmethod)Classdapat diakses dari luasClass.DidalamClassada variabel integer"MyNum"dan variabel

string”MyString“,ketika variabel dideklarasikan dalam suatuClass,mereka bisa disebutattribute.Dan untuk menutupClassmaka harus diberisemicolon(;)diakhir kodenya.

```cpp
#include <iostream>

using namespace std;

class MyClass { // Nama Class
    public :
        int myNum; // Atribut (type int)
        string myString; // Atribut (type string)
};

int main() {
    MyClass myObj;

    myObj.myNum = 20;
    myObj.myString = "Helloworld!";

    cout << myObj.myNum << "\n";
    cout << myObj.myString;

    return 0;
}
```

Gambar 7.3 penulisan object pada class main

Hasilnya seperti ini :

```
20
Helloworld!
--------------------------------
Process exited after 0.08595 seconds with return value 0
Press any key to continue . . .
```

Gambar 7.4 hasil penulisan object pada class main

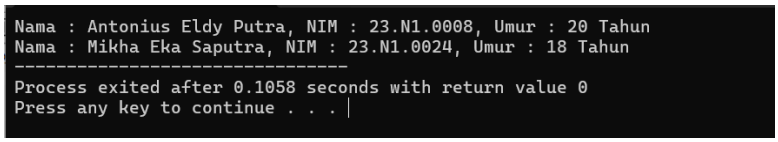
Penerapan pembuatan Class pada program C++ bisa dilihat pada gambar

diatas. Didalam fungsimain(), Class MyClass dideklarasikan dengan nama myObj atau bisa disebut Object, yang nantinya ObjectmyObj akan memanggil atribut-atribut yang sudah dibuat dalam MyClass. Untuk mengakses atribut Class (myNum dan myString), gunakan titik(.) pada Object. Atribut myNum dan MyString bisa diisi dengan value berdasarkan dengan variabel atribut tersebut didalam MyClass. Didalam satu Class, tidak hanya bisa membuat satu Object saja, tetapi bisa lebih dari satu, namun nama Object didalam satu Class tidak boleh sama antara satu dengan yang lain. Sebagai contoh bisa lihat gambar di bawah :

```cpp
#include <iostream>

using namespace std;

class Mahasiswa { // Nama Class
    public :
        string nama; // Atribut (type string)
        string nim; // Atribut (type string)
        int umur;    // Atribut (type in)
};

int main() {
    // Object 1
    Mahasiswa mahasiswa1;

    mahasiswa1.nama = "Antonius Eldy Putra";
    mahasiswa1.nim = "23.N1.0008";
    mahasiswa1.umur = 20;

    // Object 2
    Mahasiswa mahasiswa2;

    mahasiswa2.nama = "Mikha Eka Saputra";
    mahasiswa2.nim = "23.N1.0024";
    mahasiswa2.umur = 18;

    cout << "Nama : " << mahasiswa1.nama << ", NIM : " << mahasiswa1.nim << ", Umur : " << mahasiswa1.umur << " Tahun" << "\n";
    cout << "Nama : " << mahasiswa2.nama << ", NIM : " << mahasiswa2.nim << ", Umur : " << mahasiswa2.umur << " Tahun";

    return 0;
}
```

Gambar 7.5 penulisan class dan cara memanggilnya

Hasilnya seperti ini :



Gambar 7.6 hasil penulisan class dan cara memanggilnya

Dalam pembuatan lebih dari satu *Object,penamaan harus dibedakan satu sama lain agar atribut yang ada pada Class Mahasiswa bisa digunakan lebih dari satu kali pada Object yang berbeda. Misalnya Object satu bisa diberi nama mahasiswa1 dan Object kedua bisa diberi nama mahasiswa2.*

B. Class Method

Method juga bisa disebut sebagai function dan procedure. Kenapa bisa disebut juga sebagai function dan procedure? Karena disebut function jika method tersebut melakukan suatu proses dan mengembalikan nilai (return value), dan dikatakan procedure jika method tersebut melakukan sebuah proses dan tidak

mengembalikan nilai (void). Fungsi dari method ini di dalam Object-Oriented Programming adalah digunakan untuk memodularisasi sebuah program melalui pemisahan tugas di dalam suatu Class. Pemanggilan method yaitu dengan menspesifikasikan nama method dan menyediakan parameter yang diperlukan untuk proses di dalam method. Ini sangat membantu dalam membangun aplikasi yang besar dan kompleks dengan kemungkinan untuk mengelompokkan kode ke dalam unit-unit yang terorganisir dan mudah diatur. Dalam C++ method memiliki dua jenis, yaitu Instance Method dan Static Method.

a. Instance Method

Method ini berhubungan dengan objek tertentu dan dapat mengakses serta memanipulasi atribut objek. Method ini bisa dipanggil menggunakan objek spesifik dengan operator titik (.) dan setiap objek memiliki salinan instance method nya sendiri.

```
class MyClass {

public :
```

void namaMethod () {

// Memanipulasi data anggota objek

}

};

MyClass obj;

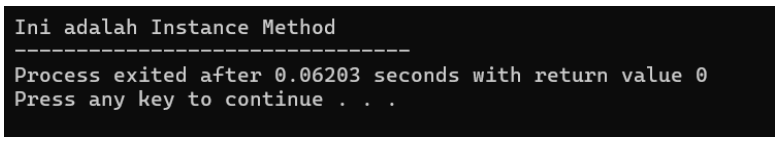
obj.namaMethod();

Contoh program :

```
#include <iostream>

using namespace std;

class MyClass {
public:
    static void staticMethod() {
        cout << "Ini adalah Static Method";
    }
};

int main() {
    MyClass::staticMethod();
}

```

Gambar 7.7 penulisan instance method

Hasil program :



Gambar 7.8 hasil penulisan instance method

b. Static Method

Static Method ini tidak berkaitan dengan instance tertentu dan seringkali digunakan untuk operasi yang berkaitan dengan Class secara keselurahan, bukan instance individ, juga tidak memiliki akses langsung dengan ke data bagian objek. Static Method dipanggil menggunakan nama Class, bukan objek, dengan operator dua titik dua (::) dan hanya memiliki satu salinan untuk seluruh Class tanpa harus membuat objek.

```
class MyClass {

public :

        static void namaMethod () {

                //          Operasi
                tanpa  akses  ke
```

data anggota
objek

}

};

MyClass::namaMethod();

Contoh program :

```
#include <iostream>

using namespace std;

class MyClass {
public:
    static void staticMethod() {
        cout << "Ini adalah Static Method";
    }
};

int main() {
    MyClass::staticMethod();
}

```

Gambar 7.9 penulisan static method

Hasil program :

```
Ini adalah Static Method
--------------------------------
Process exited after 0.05538 seconds with return value 0
Press any key to continue . . .
```

Gambar 7.10 hasil penulisan static method

Parameter

Di dalam Instance method maupun Static method dapat di tambahkan parameter untuk menerima sebuah value untuk di proses dalam method tersebut. Untuk penulisan parameter di dalam method bisa diawali dengan type data parameter yang terletak di dalam kurang, selain itu dalam method bisa di isi dengan lebih dari satu parameter, dengan syarat harus di beri koma (,) di setiap parameter. Untuk pola nya seperti berikut :

```
class NamaClass {

        public :

        namaMethod(parameter1,
        parameter2, ..) {

                //          Proses
                eksekusi

        }

};

MyClass obj;

obj.namaMethod(parameter          value,
parameter value);
```

Contoh program :

```cpp
#include <iostream>

using namespace std;

class MyClass {
public:
    luasPersegiPanjang(double panjang, double lebar) {
        return panjang * lebar;
    }
};

int main() {
    MyClass obj;
    cout << obj.luasPersegiPanjang(7, 10);
}
```

Gambar 7.11 cara penulisan parameter di method

Hasil program :

```
70
--------------------------------
Process exited after 0.04835 seconds with return value 0
Press any key to continue . . .
```

Gambar 7.12 hasil penulisan parameter di method

Return

Suatu method akan menggunakan method apabila method tersebut ingin melengkapi semua statement atau kode program di

dalam block nya, ketika kita ingin mendeklarasikan return value ke dalam deklarasi method, kita harus meletakan dalam body dari method tersebut. Perlu kita ketahui, bahwa tidak semua method memerlukan return value. Method yang menggunakan keyword void tidak memerlukan sebuah return value, apabila memaksa menggunakannya maka akan terjadi error pada program. Untuk mendapatkan deklarasi return value kita perlu menuliskan sintak seperti di bawah ini di dalam isi method :

return *returnValue;*

Perlu diperhatikan untuk type data yang digunakan yang digunakan dalam return value tersebut harus sama dengan type data yang digunakan dalam deklarasi method, kita tidak dapat return nilai string jika mendeklarasikan methodnya menggunakan interger atau yang lainnya.

Contoh program :

```cpp
#include <iostream>

using namespace std;

class MyClass {
public:
    luasPersegiPanjang(double panjang, double lebar) {
        cout << panjang * lebar;
    }
};

int main() {
    MyClass obj;
    obj.luasPersegiPanjang(7, 10);
}

```

Gambar 7.13 penulisan return pada method

Hasil program :

```
70
--------------------------------
Process exited after 0.07955 seconds with return value 0
Press any key to continue . . .
```

Gambar 7.14 hasil penulisan return ada method

C. Constructor

Constructor adalah spesial method khusus yang akan di eksekusi terlebih dahulu saat pembuatan suatu objek, yaitu saat proses instansiasi. Constructor biasa dipakai untuk membuat proses awal dalam mempersiapkan objek, seperti memberi nilai kepada data member, memanggil member function internal

serta beberapa proses yang perlu dilakukan. Di dalam program C++, cara membuat constructor yaitu dengan cara menulis sebuah member function yang namanya sama dengan nama kelas. Sebagai contoh jika kita membuat class ***Mobil****, maka function constructor juga harus ditulis* ***Mobil()****. Sebuah constructor tidak mengembalikan nilai sehingga tidak perlu menulis tipe data sebelum nama function. Constructor juga harus di set sebagai public, jika tidak maka tidak bisa meng-instansiasi class tersebut. Syarat lain dari constructor ini adalah hanya boleh ada satu constructor di dalam sebuah kelas. Pola dari constructor sebagai berikut :*

```
class NamaClass {

        NamaConstructor() {

                //          Proses
                eksekusi

        }

};


int main() {
```

NamaClass obj;

return 0;

}

Contoh program :

```cpp
#include <iostream>

using namespace std;

class Laptop {

  public:
    Laptop() {
      cout << "Satu object Laptop sudah di buat" << endl;
    }

};

int main()
{
  Laptop laptopYogi;
  Laptop laptopMikha;
  Laptop laptopAlfon;

  return 0;
}
```

Gambar 7.15 penulisan constructor

Hasil Program :

```
Satu object Laptop sudah di buat
Satu object Laptop sudah di buat
Satu object Laptop sudah di buat

--------------------------------
Process exited after 0.07639 seconds with return value 0
Press any key to continue . . .
```

Gambar 7.16 hasil penulisan constructor

Parameter pada Constructor

Constructor juga bisa memiliki sebuah parameter sama seperti pada function biasa yang dapat berguna untuk mengatur nilai inisial dari atribut.

```
class NamaClass {

    NamaConstructor(paramete
    r1, parameter2, ..) {

        // Proses eksekusi

    }

};


int main() {

    NamaClass   obj(parameter1,
    parameter2, ..);
    return 0;

}
```

Contoh program :

```cpp
#include <iostream>

using namespace std;

class Laptop {

  public:
    string returnMerk;
    string returnType;

    Laptop(string merk, string type) {
      returnMerk = merk;
      returnType = type;
    }

};

int main()
{
  Laptop laptopYogi("Acer", "Aspire 5");
  Laptop laptopMikha("Asus", "A Series");
  Laptop laptopAlfon("Lenovo", "Yoga");

  cout << laptopYogi.returnMerk << " " << laptopYogi.returnType << endl;
  cout << laptopMikha.returnMerk << " " << laptopMikha.returnType << endl;
  cout << laptopAlfon.returnMerk << " " << laptopAlfon.returnType << endl;

  return 0;
}
```

Gambar 7.17 cara penulisan parameter dalam constructor

Hasil program :

```
Acer Aspire 5
Asus A Series
Lenovo Yoga

--------------------------------
Process exited after 0.07062 seconds with return value 0
Press any key to continue . . .
```

Gambar 7.18 hasil penulisan parameter dalam constructor

Constructor di luar Class

Sama seperti function, constructor juga dapat di definisikan di luar class. Pertama, deklarasikan constructor di dalam class, kemudian definisikan di luar class dengan menentukan nama dari

class, di ikuti oleh operator dua titik dua (::) di ikuti nama dari constructor.

```
class NamaClass {

        NamaConstructor(paramete
        r1, parameter2, ..);

};


NamaConstructor::NamaConstructor(par
ameter1, parameter2, ..) {

        // Proses eksekusi

}


int main() {

        NamaClass   obj(parameter1,
        parameter2, ..);
        return 0;

}
```

Contoh program :

```cpp
#include <iostream>

using namespace std;

class Laptop {

  public:
    string returnMerk;
    string returnType;

    Laptop(string merk, string type);

};

Laptop::Laptop(string merk, string type) {
      returnMerk = merk;
      returnType = type;
    }

int main()
{
  Laptop laptopYogi("Acer", "Aspire 5");
  Laptop laptopMikha("Asus", "A Series");
  Laptop laptopAlfon("Lenovo", "Yoga");

  cout << laptopYogi.returnMerk << " " << laptopYogi.returnType << endl;
  cout << laptopMikha.returnMerk << " " << laptopMikha.returnType << endl;
  cout << laptopAlfon.returnMerk << " " << laptopAlfon.returnType << endl;

  return 0;
}
```

Gambar 7.19 penulisan constructor di luar class

Hasil program :

```
Acer Aspire 5
Asus A Series
Lenovo Yoga

--------------------------------
Process exited after 0.08092 seconds with return value 0
Press any key to continue . . .
```

Gambar 7.20 hasil penulisan constructor di luar class

D. Destructor

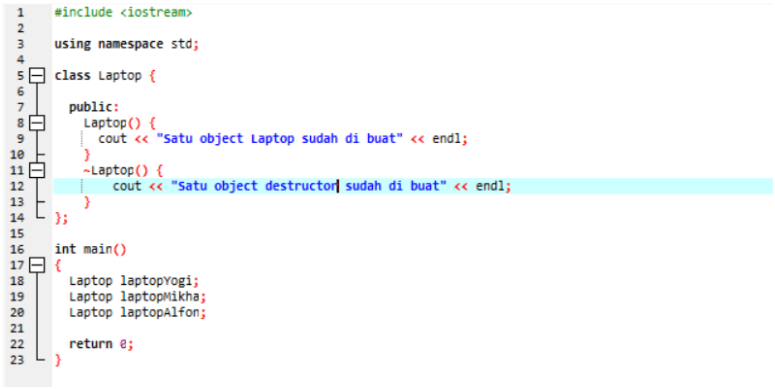
Destructor adalah method khusus yang dijalankan secara otomatis pada saat sebuah objek dihapus. Sebenarnya seluruh objek sudah otomatis dihapus ketika program selesai di proses, tetapi kita juga

bisa menghapus objek secara manual. Destructor biasa dipakai untuk membersihkan beberapa variabel atau menjalankan proses tertentu sebelum objek dihapus. Untuk pola nya seperti berikut :

```
class NamaClass {

    NamaConstructor() {

        // Proses eksekusi

    }


    ~NamaDestructor() {

        // Proses eksekusi

    }

};


int main() {

    NamaClass obj;
    return 0;

}
```

Ketika objek dibuat maka constructor otomatis terpanggil, ketika program berakhir destructor dipanggil secara otomatis. Diperlukan operator (~) untuk membuat sebuah destructor, di ikuti dengan nama destructor, tetapi nama destructor harus sama dengan nama constructor.

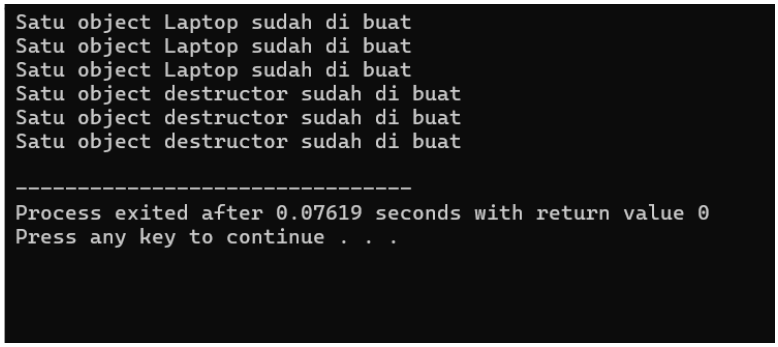
Contoh program :



```
#include <iostream>

using namespace std;

class Laptop {

  public:
    Laptop() {
      cout << "Satu object Laptop sudah di buat" << endl;
    }
    ~Laptop() {
        cout << "Satu object destructor sudah di buat" << endl;
    }
};

int main()
{
  Laptop laptopYogi;
  Laptop laptopMikha;
  Laptop laptopAlfon;

  return 0;
}
```

Gambar 7.21 penulisan destructor

Hasil Program :



Gambar 7.22 hasil penulisan destructor

D. Encapsulation

Encapsulation adalah suatu konsep Object-Oriented Programming yang digunakan untuk membungkus data atau fungsi, agar tetap terjaga agar tidak adanya penyalahgunaan dari pengguna. Konsep Encapsulation menyebabkan sebuah konsep Object-Oriented Programming bernama Data Hidding atau Abstraction. Encapsulation juga sebuah teknik untuk membuat antar muka dan menyembunyikan mekanisme atau isi seluruh yang berhubungan dengan pengguna, hal ini pengguna tidak diperbolehkan untuk mengakses data yang disembunyikan secara langsung oleh programer, tapi bisa menggunakan dan memahami dengan mudah berdasarkan antarmuka yang telah disediakan.

```
class MyClass {
        private :
        int content;

        public :

        void setContent(int c) {
                content = c;
        }

        int getContent() {
                return content;
        }
};

int main() {
        MyClass obj;
        obj.setContent(2500);
        cout << getContent();
```

return 0;

}

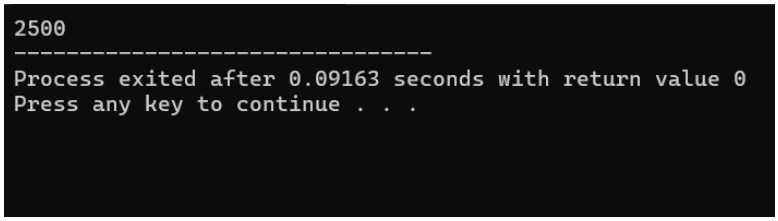
Disini atribut "content" bersifat private yang memiliki akses terbatas. Method "setContent" mengambil parameter dan dimasukan ke atribut "c" (content = c), sedangkan method "getContent" publik mengembalikan nilai atribut dari "content". Di dalam "main()" setelah membuat objek "obj", kita bisa menggunakan method "setContent" untuk menyetel nilai atribut private menjadi 2500 karna tipe data yang digunakan atribut adalah integer. Kemudian kita bisa memanggil method "getContent" pada objek untuk mengembalikan nilai.

Contoh program :

```
#include <iostream>

using namespace std;


class Mobil {
    private :
        int cc;

    public :
        void setMobil(int cubicCentimeter) {
            cc = cubicCentimeter;
        }

        int getMobil() {
            return cc;
        }
};

int main() {
    Mobil obj;
    obj.setMobil(2500);
    cout << obj.getMobil();
}
```

Gambar 7.23 penulisan encapsulation

Hasil program :



Gambar 7.24 hasil penulisan encapsulation

D. Inheritance

Inheritance atau pewarisan adalah sebuah konsep Object-Oriented Programming dimana sebuah Class dapat menurunkan data member dan member function yang di miliki kepada Class lain. Konsep Inheritance dipakai untuk memanfaatkan fitur code reuse, yaitu menghindari terjadinya duplikasi kode pada program, dan membuat sebuah struktur Class ke dalam program. Class yang diturunkan bisa di sebut sebagai Parent Class atau kelas induk, sedangkan Class yang di warisi bisa disebut sebagai Child Class atau kelas anak. Tidak semua data member dan member function Parent

Class ke Child Class. Data dan function dengan hak akses privat tidak akan di turunkan ke Child Class, hanya data dan function yang memiliki sifat public saja yang bisa di akses dari Child Class.

```
class ParentClass {

        // proses eksekusi

}


class ChildClass: public ParentClass{

        // kode untuk class turunan
        Child Class

}


int main() {

        ParentClass parentClass;

        ChildClass childClass;


        return 0;

}
```

Contoh program :

```
#include <iostream>

using namespace std;


class Mobil {

};

class Motor: public Mobil {

};

int main() {
    Mobil mobilObj;
    Motoro motorObj;
}
```

Gambar 7.25 penulisan class turunan

Gambar diatas adalah cara untuk membuat sebuah Child Class, hanya saja hasil programnya masih kosong karena belum diberi proses apapun di dalam Parent Class maupun Child Class. Seperti ini adalah cara untuk mengakses data Parent Class :

Contoh program :

```cpp
#include <iostream>

using namespace std;


class Mobil {
    public :
        string merk = "Honda";

        string cekMobil() {
            return "Mobil Honda";
        }
};

class Motor: public Mobil {
    public :
        string jenis = "Motor";

        string cekMotor() {
            return "Motor Honda";
        }
};

int main() {
    Motor motorku;

    cout << motorku.merk << endl;
    cout << motorku.cekMobil() << endl;
    cout << motorku.jenis << endl;
    cout << motorku.cekMotor();
}
```

Gambar 7.26 penulisan cara mengakses class induk

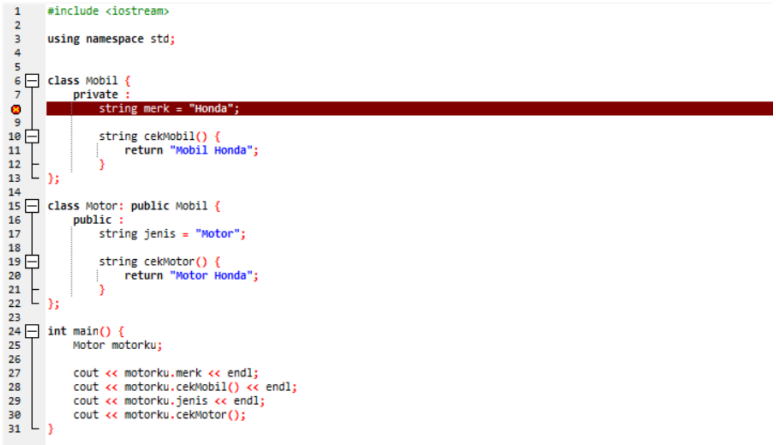
Hasil program :

```
Honda
Mobil Honda
Motor
Motor Honda
--------------------------------
Process exited after 0.07783 seconds with return value 0
Press any key to continue . . .
```

Gambar 7.27 hasil penulisan cara mengakses class induk

Kode diatas, Class "Mobil" memiliki satu data member dan satu data function, sedangkan Class "Motor" juga memiliki satu data member dan satu data function. Class "Motor" adalah Class turunan dari

Parent Class "Mobil". Di dalam function main() membuat objek "motorku" yang di instansiasi dari class "Motor", dan objek "motorku" bisa mengakses semua isi Class "Mobil". Jika Class "Mobil" di set menjadi privat, maka program akan menjadi error seperti ini :



```
#include <iostream>

using namespace std;


class Mobil {
    private :
        string merk = "Honda";

        string cekMobil() {
            return "Mobil Honda";
        }
};

class Motor: public Mobil {
    public :
        string jenis = "Motor";

        string cekMotor() {
            return "Motor Honda";
        }
};

int main() {
    Motor motorku;

    cout << motorku.merk << endl;
    cout << motorku.cekMobil() << endl;
    cout << motorku.jenis << endl;
    cout << motorku.cekMotor();
}
```

Gambar 7.28 penulisan class induk dengan di set private

Log error :

		D:\PERKULIAHAN\SEMESTER 2\PENULISAN KREATIF\...	In function 'int main()':
8	17	D:\PERKULIAHAN\SEMESTER 2\PENULISAN KREATIF\PR...	[Error] 'std::string Mobil::merk' is private
27	18	D:\PERKULIAHAN\SEMESTER 2\PENULISAN KREATIF\PR...	[Error] within this context
10	10	D:\PERKULIAHAN\SEMESTER 2\PENULISAN KREATIF\PR...	[Error] 'std::string Mobil::cekMobil()' is private
28	27	D:\PERKULIAHAN\SEMESTER 2\PENULISAN KREATIF\PR...	[Error] within this context

Gambar 7.29 log error karna class turunan mengakses class induk yang di

set private

Seperti yang kita lihat pada gambar diatas, ada terjadi error karena Child Class gagal mengakses Parent Class yang di set privat.

Multilevel Inheritance

Multilevel Inheritance adalah penurunan Class berjenjang dari satu Class ke Class lain. Misalnya kita punya Class A yang diturunkan ke Class B, kemudian Class B ini diturunkan lagi ke Class C, sehingga itu sudah terjadi Multilevel Inheritance. Dalam contoh ini Class A adalah Grand Parent Class dari Class C, maka semua data member dan data function dari Class A juga bisa di akses di Class C selama tidak di set privat.

```
class GrandParentClass {

        public :

        int atribut1;

}
```

```
class ParentClass: public GrandParentClass {
    public :
    int atribut2;
}

class ChildClass: public ParentClass {
    public :
    int atribut3;
}

int main() {
    ChildClass obj3;

    cout << obj3.atribut1 + obj3.atribut2 + obj.atribut3;
}
```

Contoh program :

```cpp
#include <iostream>

using namespace std;


class Manusia {
    public :
        string manusia = "ini makhluk hidup";
};
class Hewan: public Manusia {
    public :
        string hewan = "ini makhluk hidup";


};
class Tumbuhan: public Hewan {
    public :
        string tumbuhan = "ini makhluk hidup";
};

int main() {
    Tumbuhan tumbuhan;

    cout << tumbuhan.manusia << endl;
    cout << tumbuhan.hewan << endl;
    cout << tumbuhan.tumbuhan;
}
```

Gambar 7.30 penulisan multilevel inheritance

Hasil program :

```
ini makhluk hidup
ini makhluk hidup
ini makhluk hidup
--------------------------------
Process exited after 0.05016 seconds with return value 0
Press any key to continue . . .
```

Gambar 7.31 hasil penulisan multilevel inheritance

Dari hasil program diatas bisa dilihat bahwa Grand Parent Class bisa diakses oleh Child Class yang menerima warisan dari Parent Class, dan Grand Parent Class

yang sudah mewariskan data member dan data function nya ke Parent Class.

Multiple Inheritance

Konsep Multiple Inheritance hampir sama dengan Multilevel Inheritance, sama-sama bisa mengakses dua Class dari Grand Parent Class dan Parent Class, tetapi beda nya Multiple Inheritance ini adalah penurunan Class dimana sebuah Class bisa memiliki lebih dari satu Parent Class, yang artinya tidak ada Grand Parent Class tetapi bisa langsung mengakses dua atau lebih Parent Class. Untuk penulisannya antar Parent Class yang ingin mewariskan dipisah dengan tanda koma (,) saat membuat Child Class.

```
class ParentClass1 {

        // proses eksekusi

}


class ParentClass2 {

        // proses eksekusi

}
```

class ChildClass: public ParentClass1, public ParentClass2 {

// proses eksekusi

}

int main() {

ChildClass obj;

cout << obj;

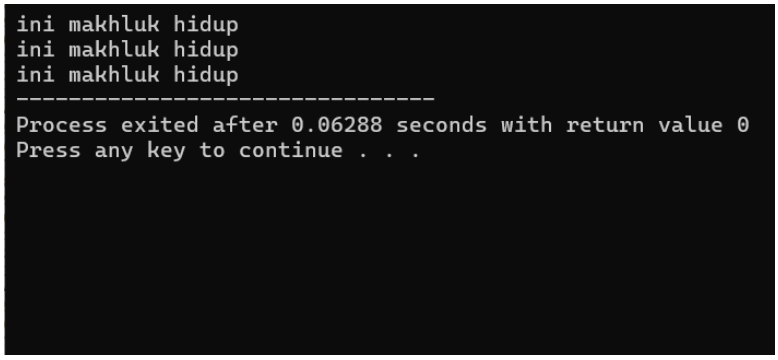
}

Contoh program :

```cpp
#include <iostream>

using namespace std;


class Manusia {
    public :
        string manusia = "ini makhluk hidup";
};
class Hewan {
    public :
        string hewan = "ini makhluk hidup";
};
class Tumbuhan: public Manusia, public Hewan {
    public :
        string tumbuhan = "ini makhluk hidup";
};
int main() {
    Tumbuhan tumbuhan;

    cout << tumbuhan.manusia << endl;
    cout << tumbuhan.hewan << endl;
    cout << tumbuhan.tumbuhan;
}
```

Gambar 7.32 penulisan multiple inheritance

Hasil program :



Gambar 7.33 hasil penulisan multiple inheritance

Access Specifiers

*Access Specifiers atau penentu akses adalah salah satu bagian penting di dalam konsep Object-Oriented Programming untuk melakukan Data Hiding atau bisa disebut penyembunyian data. Fitur ini memungkinkan kita untuk mengatur hak akses dari member class, yang digunakan agar tidak sembarangan perintah dapat mengakses member data dan member function dalam sebuah Class, atau tidak bisa diakses secara langsung. Ada tiga buah tipe akses yang bisa digunakan di dalam konsep Object-Oriented Programming, k*etiga penentu

akses mempunyai sifatnya masing-masing, dan sifat default adalah*private*, jadi jika tidak didirikan sebuah*Access Specifiers*di Class nya maka akan otomatis ter set menjadi*private*.

1. public

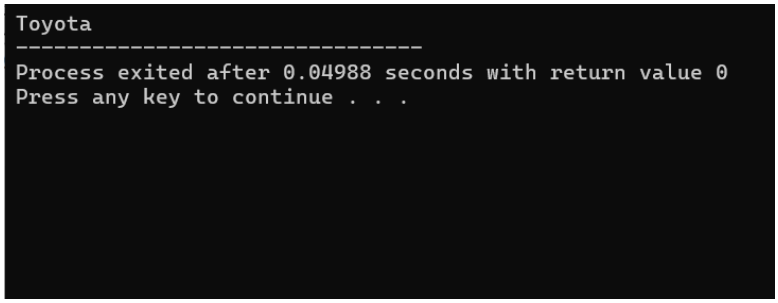
public adalah label yang berfungsi untuk menentukan sifat akses ke semua data member dari Child Class, sehingga Child Class maupun class main bebas mengakses data member dan data function dari sebuah Parent Class yang di set public.

Contoh program :

```
#include <iostream>

using namespace std;

class Mobil {
    public :
        string merk;
};

int main() {

    Mobil mobil;
    mobil.merk = "Toyota";
    cout << mobil.merk;

}
```

Gambar 7.34 penulisan access specifiers public

Hasil program :



Gambar 7.35 hasil penulisan access specifiers public

2. private

private adalah label yang berfungsi untuk menentukan sifat akses ke semua member yang mengikutinya menjadi memiliki sifat yang tidak dapat di akses dari manapun kecuali friend function dan dari dalam Class itu sendiri.

Contoh program :

```cpp
#include <iostream>

using namespace std;

class Mobil {
    private :
        string merk;
};

int main() {

    Mobil mobil;
    mobil.merk = "Toyota";
    cout << mobil.merk;

}
```

Gambar 7.36 penulisan access specifiers private

Log error :

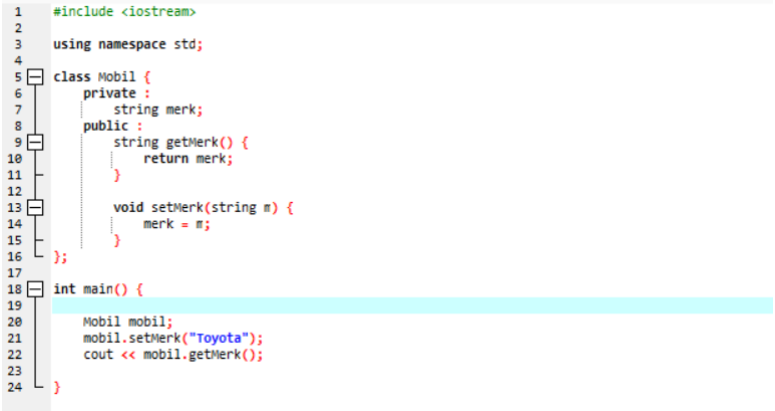
Line	Col	File	Message
		D:\PERKULIAHAN\SEMESTER 2\PENULISAN KREATIF\...	**In function 'int main()':**
7	10	D:\PERKULIAHAN\SEMESTER 2\PENULISAN KREATIF\PR...	[Error] 'std::string Mobil::merk' is private
13	8	D:\PERKULIAHAN\SEMESTER 2\PENULISAN KREATIF\PR...	[Error] within this context
7	10	D:\PERKULIAHAN\SEMESTER 2\PENULISAN KREATIF\PR...	[Error] 'std::string Mobil::merk' is private
14	16	D:\PERKULIAHAN\SEMESTER 2\PENULISAN KREATIF\PR...	[Error] within this context

Gambar 7.37 log error hasil penulisan access specifers private

Hasil program yang dijalankan di atas terjadi error karena Child Class mengakses data member dari Parent Class yang di set private. Pesan itu

membuktikan bahwa member "merk" mengikuti sifat level private, yang tidak dapat di akses dari luar Class. Private memiliki sifat hanya dapat di akses oleh Class itu sendiri. Tetapi ada cara untuk mengakses data member "merk" dari luar Class dengan cara membuat perantara di dalam Class tersebut, yaitu dengan function yang bersifat public, cara ini disebut dengan konsep setter and getter.

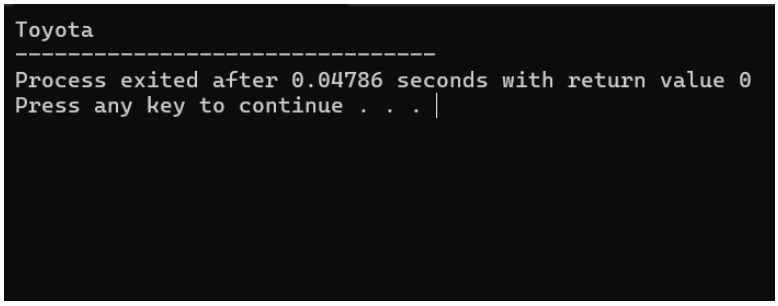
Contoh Program :



```
#include <iostream>

using namespace std;

class Mobil {
    private :
        string merk;
    public :
        string getMerk() {
            return merk;
        }

        void setMerk(string m) {
            merk = m;
        }
};

int main() {

    Mobil mobil;
    mobil.setMerk("Toyota");
    cout << mobil.getMerk();

}
```

Gambar 7.38 penulisan access specifiers private

Hasil program :



Gambar 7.39 hasil penulisan access specifiers private

3. protected

Protected adalah label yang berfungsi untuk menentukan sifat akses semua data member dan data function yang mengikutinya, sehingga memiliki sifat yang tidak dapat di akses dari luar Class, tetapi masih dapat di akses dari dalam Class maupun Child Class.

Contoh program :

```cpp
#include <iostream>

using namespace std;

class Mobil {
    protected :
        string merk;

};

class Motor: public Mobil {
    public :
        string getMerk() {
            return merk;
        }
};

int main() {

    Motor motor;
    motor.merk = "Toyota";
    cout << motor.merk;
    return 0;
}
```

Gambar 7.40 penulisan access specifiers protected

Hasil program :

Line	Col	File	Message
		D:\PERKULIAHAN\SEMESTER 2\PENULISAN KREATIF\...	**In function 'int main()':**
7	10	D:\PERKULIAHAN\SEMESTER 2\PENULISAN KREATIF\PR...	[Error] 'std::string Mobil::merk' is protected
21	8	D:\PERKULIAHAN\SEMESTER 2\PENULISAN KREATIF\PR...	[Error] within this context
7	10	D:\PERKULIAHAN\SEMESTER 2\PENULISAN KREATIF\PR...	[Error] 'std::string Mobil::merk' is protected
22	16	D:\PERKULIAHAN\SEMESTER 2\PENULISAN KREATIF\PR...	[Error] within this context

Gambar 7.41 hasil penulisan access specifiers protected

Dari hasil program di atas terjadi error atribut "merk" yang pesannya berisi "merk is protected", yang berarti atribut "merk" dari Parent Class tidak bisa di akses di Main Class, tetapi saat atribut "merk" di panggil di Child Class tidak ada

tanda-tanda terjadi error. Inti nya adalah atribut "merk" hanya bisa di akses di Class turunannya saja, dan tidak bisa di panggil di Main Class.

Penutup

Pemrograman Berorientasi Objek atau Object-Oriented Programming ini memang sulit di pelajari, jadi perlu untuk memahami dasar-dasar pemrograman seperti variabel, type data, function dll. Harus ada urutan-urutannya untuk bisa memahami konsep Object-Oriented Programming ini. Jika masih belum paham tentang konsep Object-Oriented Programming, kami sarankan untuk mempelajari terlebih dahulu dasar-dasar pemrograman agar mudah untuk mempelajari konsep Object-Oriented Programming ini.

Daftar Pustaka

SUMBER:

1. “Data Structures and Algorithms in C++” oleh Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser: http://1605928573 66.free.fr/joe/ebooks/ShareData/Data%20Structures%20and%20Algorithms%20in

2. “C++ Programming Language” oleh Bjarne Stroustrup: https://scholar.google.com/citations?user=Rr9Y8acAAAAJ&hl=en

3. “Array - GeeksforGeeks”: https://www.geeksforgeeks.org/array-data-structure/

4. “C++ Classes” : https://www.w3schools.com/cpp/cpp_oop.asp

Profil Penulis

Bernardinus Harnadi

adalah dosen Sistem Informasi Unika Soegijapranata di Unika Soegijapranata

Antonius Eldy Putra

adalah mahasiswa Sistem Informasi angkatan 2023 di Unika Soegijapranata

Wilibrordus Endra Bima

adalah mahasiswa Sistem Informasi angkatan 2023 di Unika Soegijapranata

Alfonso Praditya Galuh Mahesa

adalah mahasiswa Sistem Informasi angkatan 2023 di Unika Soegijapranata

Ignatius Yogyawan Dwi

adalah mahasiswa Sistem Informasi angkatan 2023 di Unika Soegijapranata

Mikha Eka Saputra

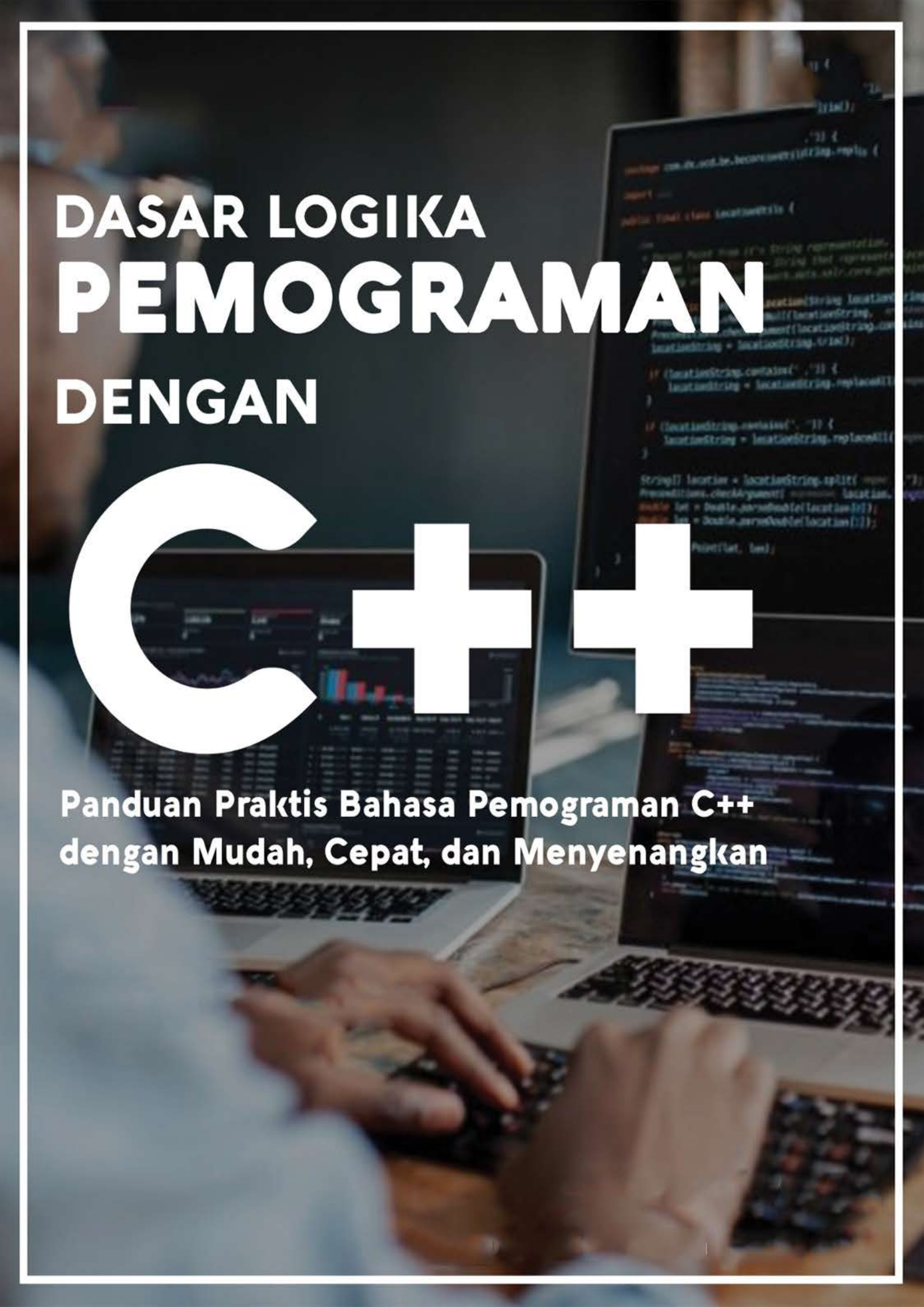
adalah mahasiswa Sistem Informasi angkatan 2023 di Unika Soegijapranata

Tegar Heru Saputra

adalah mahasiswa Sistem Informasi angkatan 2023 di Unika Soegijapranata

Irwan Wirawan Gulo

adalah mahasiswa Sistem Informasi angkatan 2023 di Unika Soegijapranata



DASAR LOGIKA PEMOGRAMAN DENGAN C++

Panduan Praktis Bahasa Pemograman C++ dengan Mudah, Cepat, dan Menyenangkan