Belajar PEMROGRAMAN LANJUT Dengan C++

Muhammad Taufik Dwi Putra
Munawir
Ana Rahma Yuniarti

Belajar PEMROGRAMAN LANJUT Dengan C++

Muhammad Taufik Dwi Putra
Munawir
Ana Rahma Yuniarti

Penerbit Widina

BELAJAR PEMROGRAMAN LANJUT DENGAN C++

Penulis:
Muhammad Taufik Dwi Putra, Munawir, Ana Rahma Yuniarti

Desain Cover:
Fawwaz Abyan

Sumber Ilustrasi:
www.freepik.com

Tata Letak:
Handarini Rohana

Editor:
Evi Damayanti

ISBN:
978-623-459-689-2

Cetakan Pertama:
September, 2023

PENERBIT:
WIDINA MEDIA UTAMA
Komplek Puri Melia Asri Blok C3 No. 17 Desa Bojong Emas
Kec. Solokan Jeruk Kabupaten Bandung, Provinsi Jawa Barat

Anggota IKAPI No. 360/JBA/2020
Website: www.penerbitwidina.com
Instagram: @penerbitwidina
Telepon (022) 87355370

PRAKATA PENULIS

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa dan tidak lupa ucapan terima kasih penulis sampaikan kepada seluruh pihak yang telah mendukung penulis dalam penyusunan buku ini sehingga buku ini dapat sampai di tangan para pembacanya. Penulis akhirnya dapat menyelesaikan buku dengan judul Belajar Pemrograman Lanjut dengan C++ setelah begitu banyak perbaikan dan penyempurnaan. Buku ini berisikan materi mengenai pemrograman C++ dan algoritma pemrograman secara umum.

Penyajian materi disusun dengan penggunaan bahasa yang lebih mudah dimengerti, karena penulis menyajikan berbagai pokok bahasan dengan mempertimbangkan tingkat kemampuan pembaca yang masih belum memiliki pengetahuan mengenai pemrograman agar dapat lebih dimengerti oleh masyarakat umum. Penulis juga menyajikan contoh-contoh dalam kehidupan sehari-hari agar para pembaca bisa lebih memahami bagaimana cara kerja dari materi yang disampaikan.

Pada akhirnya, buku yang ditulis oleh manusia biasa tentunya berisi berbagai kekurangan. Sehingga harapannya jika ditemukan kesalahan pada buku ini, baik kesalahan penulisan maupun kesalahan materi mohon dimaafkan.

Wassalam.

Bandung, September, 2023

Muhammad Taufik

Persembahan

Untuk Tim Peneliti Kecerdasan Buatan:

Haryanto Hidayat

Mochamad Aldi Sidik Maulana

Muhamad Hisyam Nugraha Solihin

Muhammad Aksyal Bambang Suseno

Muhammad Andhika Ramadhan

Rahmawati

Rizal Maulana Komarudin

Salman Abdul Jabbaar Wiharja

Syiva Awaliyah Maqdis

DAFTAR ISI

FUNGSI DAN PROSEDUR

A. DEFINISI FUNGSI DAN PROSEDUR

a. Definisi Fungsi

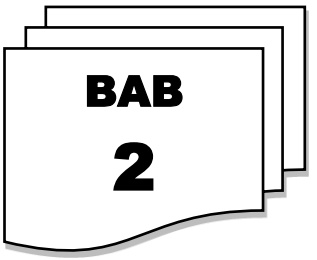
Fungsi merupakan blok program yang melakukan suatu aksi tertentu berdasarkan *input* yang diterima dan mengembalikan sebuah nilai ketika deklarasinya dipanggil dalam sebuah program. Fungsi dapat dipanggil/ digunakan di dalam blok kode yang sama maupun di blok kode program utama. Format sebuah fungsi adalah sebagai berikut:

```
1  tipe_data nama_fungsi(parameter_fungsi){
2  /*
3       blok program
4       fungsi
5  */
6       return nilai
7  }
```

Gambar 1.1 Format Penulisan Fungsi

Pada fungsi ini diperlukannya spesifikasi tipe data yang digunakan, kemudian diikuti dengan nama fungsi, parameter fungsi, isi fungsi, dan terakhir adalah nilai yang dikembalikan oleh fungsi. Perlu diperhatikan bahwa nilai yang dikembalikan harus memiliki tipe data yang sama dengan tipe data fungsi itu sendiri. Contoh penulisan fungsi:

```
int kuadrat(int angka){
    angka *= angka;
    return angka;
}
```



POINTER

A. PENGERTIAN POINTER

Ketika membuat sebuah program, apapun itu programnya, tidak lepas dengan penggunaan sebuah variabel. Setiap variabel yang telah dideklarasikan, variabel tersebut pasti disimpan di dalam sebuah memori. Dikarenakan telah disimpan di dalam sebuah memori, pastinya variabel tersebut memiliki sebuah alamat memori. Alamat memori ini berfungsi sebagai lokasi penyimpanan data pada sebuah RAM. Lalu apa hubungan antara alamat memori dengan sebuah pointer? Pointer merupakan suatu variabel penunjuk yang berisi alamat pada suatu lokasi memori tertentu. Sehingga, sebuah pointer tidak berisi nilai dari sebuah variabel yang telah dideklarasikan, melainkan berisi suatu alamat memori. Dengan menggunakan pointer, dimungkinkan untuk mendapatkan atau mengubah isi dari memori yang ditunjuk.

Misalnya, terdapat sebuah variabel X dengan tipe data integer bernilai 5, disimpan di memori dengan alamat 0xfffa. Kemudian ada sebuah pointer Y dengan alamat 0xfffb yang mengarah kepada variabel X. Maka nilai dari pointer Y (0xfffb) ini akan sama dengan variabel X (0xfffa) yaitu 5.

Penggunaan pointer akan meningkatkan performa secara signifikan dikarenakan tidak ada proses pengalokasian alamat memori baru, meskipun memang penggunaannya akan lebih sulit apabila dibandingkan tanpa menggunakan pointer.

BAB
3

SORTING

A. PENGERTIAN SORTING

Sorting adalah proses mengatur sekumpulan objek menurut urutan atau susunan tertentu. Urutan objek tersebut dapat menaik (*ascending*), yaitu urutan objek yang disusun mulai dari Nilai terkecil hingga terbesar atau menurun (*descending*), yaitu urutan objek yang disusun mulai dari Nilai terbesar hingga terkecil.

B. JENIS-JENIS SORTING

a. Bubble Sort

Bubble sort adalah metode pengurutan yang terus-menerus menukar dua buah elemen hingga pengurutan selesai. Cara ini sering dikatakan tidak efisien, namun sangat mudah untuk dipahami. Misalkan sebuah array berisi angka sebagai berikut:

Gambar 3.1 Sebuah array sebelum diurutkan

SEARCHING

A. PENGERTIAN SEARCHING

Searching adalah tindakan mengambil data dari kumpulan data berdasarkan kunci (*key*) atau referensi data. Di kehidupan sehari-hari, kita sering berurusan dengan pencarian. Misalnya, mencari nomor telepon seseorang di buku telepon, atau mencari kata pada kamus, dan masih banyak lagi. Pencarian sering dilakukan dalam aplikasi komputer. Misalnya, jika ingin menghapus atau mengubah data/record tertentu dalam suatu file, harus terlebih dahulu menentukan apakah data tersebut ada dalam file. Jika ada, maka data dapat dihapus atau diubah.

B. SEQUENTIAL SEARCH PADA C++

Sequential search adalah metode pencarian paling sederhana. Proses pencarian sequential yaitu dengan membandingkan setiap elemen array satu per satu secara beruntun, dimulai dengan elemen pertama, sampai elemen yang dicari ditemukan atau sampai elemen terakhir dari array. Pencarian sekuensial dapat dilakukan pada elemen array yang tidak diurutkan atau pada elemen array yang diurutkan. Perbedaan antara keduanya terletak pada efisiensi operasi perbandingan yang dilakukan. Dengan begitu, proses pencarian sequential akan singkat jika data yang diolah sedikit, dan jika data yang diolah banyak, prosesnya akan lama. Maka, metode ini direkomendasikan untuk data yang sedikit. Sebagai contoh diberikan suatu array dengan jumlah 8 elemen, seperti berikut:

BAB

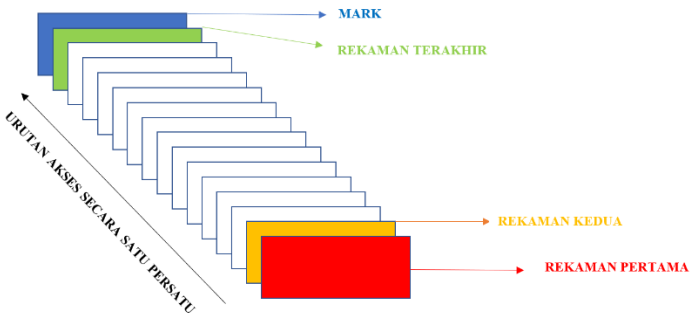
5

SEQUENTIAL FILE

A. PENGERTIAN SEQUENTIAL FILE

Sequential file merupakan kumpulan-kumpulan rekaman suatu data yang disimpan pada penyimpanan cadangan pada komputer, untuk mengakses rekaman yang diinginkan harus mulai dari urutan pertama hingga ke rekaman yang diinginkan dengan kata lain tidak dapat diakses secara acak. Selain itu, jika akan menambahkan rekaman baru maka peletakan rekaman tersebut berada di akhir file. Rekaman terakhir disini menandakan bahwa arsip terakhir dari rekaman tersebut bisa disebut juga sebagai EOF (End of File).

Adapun kelebihan dari penggunaan *sequential file* yaitu memiliki tingkat akurasi yang tepat dalam proses pengaksesan rekaman. Namun, selain memiliki kelebihan *sequential file* ini memiliki kekurangan yaitu tidak dapat mengakses rekaman yang diinginkan secara langsung ataupun mengakses secara acak. Karena rekaman-rekaman dari *sequential file* ini harus diakses secara berurutan. Maka dari ini proses yang paling sering digunakan adalah *batch processing* dibandingkan dengan *interactive processing*.



Gambar 5.1 Konsep sequential file

STRUCT

A. PENGERTIAN STRUCT

Struct dalam bahasa C++ merupakan struktur data yang memungkinkan terjadinya pembentukan tipe data baru dengan menggabungkan berbagai macam variabel dengan tipe data berbeda yang tersedia dalam C++. Tipe data yang baru, dapat dibentuk di luar tipe data yang sudah ada dengan menggabungkan beberapa tipe data tersebut sesuai dengan kebutuhan program atau aplikasi yang dirancang.

Berbeda dengan larik (array) yang memungkinkan penyimpanan beberapa tipe data yang sama, struct dapat menyimpan dan menggabungkan berbagai variabel dengan tipe data yang berbeda. Misalkan data mengenai nama, NIM, program studi, dan universitas. Keempat data tersebut memiliki tipe data yang berbeda tetapi masih dalam satu kelompok yaitu data mahasiswa.

Berbagai variabel yang membangun suatu struktur dapat juga disebut sebagai elemen atau anggota struktur. Penggunaan struct memudahkan pemetaan entitas dari sebuah kehidupan yang nyata ataupun suatu model dengan penyimpanan dan *handling* dalam C++.

B. NOTASI STRUCT

Notasi (syntax) struct dalam C++ terbilang mudah untuk dipahami dan digunakan. Tuliskan kata kunci “struct” tanpa petik pada awal kode dilanjutkan dengan nama struktur yang diinginkan seperti nama_struktur. Kemudian tambahkan berbagai variabel dengan berbagai tipe data berbeda yang diperlukan di dalam tanda kurung kurawal. Selanjutnya nama_struktur akan dianggap sebagai tipe data baru yang dapat digunakan dalam program. Notasi tersebut dapat ditulis sebagai berikut:

BAB
7

BILANGAN ACAK

Bilangan random (acak) dapat dibuat pada pemrograman C++ dengan memanfaatkan library yang sudah disediakan dari bahasa pemrograman C++ nya itu sendiri, yaitu menggunakan library stdlib.h yang memiliki fungsi untuk membangkitkan bilangan acak atau random tersebut. Untuk implementasi library stdlib.h dapat dilihat pada gambar dibawah ini:

```
1   #include <iostream>
2   #include <stdlib.h>
3   #include <time.h> //library untuk waktu
4   using namespace std;
5   int main(){
6       int random;
7       srand(time(0));
8       for(int a=0;a<=10;a++){
9           random = rand()%4;
10          cout << "Acak" << a << " = " << random << endl'
11      }
12  }
```

Gambar 7.1 Implementasi Library Stdlib.h

Alasan harus menggunakan rand() adalah bisa memudahkan ketika akan membuat sebuah program lempar dadu, atau program pembagian kelompok belajar.

A. PENJELASAN DAN PENGGUNAAN RAND() DALAM C++

Dalam C++ fungsi rand() digunakan untuk menghasilkan angka acak atau *random*. Untuk menulis program bilangan acak ini, cukup dengan menggunakan rand() akan menghasilkan angka *random*:

BAB

8

FILE EKSTERNAL

A. PENDAHULUAN

Masukkan dan keluaran atau *input* dan *output* erat kaitannya dan pasti selalu ada ketika membuat program menggunakan bahasa C++. *Stream Input/Output* mengandung *flow character* baik dari *input device* ataupun *output device* ke program. *Stream* I/O bersifat sementara, ketika program dihentikan maka data atau nilai tidak akan tersimpan. Variabel dalam program memang dapat menyimpan data dan nilai, namun hanya sementara dan nilai yang tersimpan kemudian akan menghilang ketika program dimatikan. Perhatikan contoh program catat nama berikut.

```
1  //Program CatatNama
2  //Membaca nama dari input dan menampilkan output ke layar
3  #include <iostream>
4  using namespace std;
5
6  int main() {
7      //Deklarasi
8      string nama;
9
10     //Algoritma
11     cout << "Silahkan catat namamu: " << endl;
12
13     cin >> nama;
14
15     cout << "Namamu adalah: " << nama << endl;
16
17     return 0;
18 }
```

Gambar 8.1 Program catat nama

BAB
9

PROPROCESSING, COMPILING, DAN LINKING

A. PREPROCESSING

Preprocessing adalah langkah pertama yang dilakukan program sebelum kompilasi. Membangun program di C++ dimulai dengan script program yang dimulai dengan karakter #.

Contoh script yang umum digunakan adalah #include. Script ini nantinya akan diterjemahkan ke dalam bentuk kode aktual yang akan diperluas. Misalnya, jika pada script terdapat pernyataan #include <iostream>, seluruh isi file “iostream” akan disertakan ke dalam kode program. C++ memiliki beberapa *preprocessor* di dalamnya, yaitu:

a. #include: Menyertakan isi file header ("*.h") dalam kode program.
b. #define: Menggantikan identifier dalam program dengan makro atau suatu nilai.
c. #undef: Menghapus identifier yang sudah dibuat dengan #define.
d. #Pragma: Perintah khusus compiler yang bisa berbeda untuk setiap compiler.
e. Conditional Preprocessor (#if, #elseif, #else): Menentukan bagian program yang akan di-compile berdasarkan suatu persyaratan seperti OS atau compilernya.

Sebelumnya, kita menggunakan #include untuk menyertakan header standar. Ada dua cara untuk menggunakan #include.

a. Jika #include menggunakan header C++ standar atau header default yang telah ditentukan, nama file diapit oleh "<" dan ">".

REFERENCE

A. PENGERTIAN REFERENCE

Reference atau disebut juga *reference variable* atau variabel referensi adalah variabel yang dijadikan acuan bagi variabel lain yang telah dibuat sebelumnya. Referensi memungkinkan membuat sebuah variabel yang dapat menyimpan alamat memori dan bertindak seperti variabel tetapi sebenarnya adalah alias dari alamat memori, objek, atau variabel. Saat mendeklarasikan variabel, pada dasarnya harus mengalokasikan alamat memori yang digunakan untuk menyimpan nilai. Tidak seperti saat mendeklarasikan variabel referensi, yaitu tidak membuat alokasi memori baru melainkan mereferensikan alamat memori yang sebelumnya dibuat oleh variabel lain. Referensi terdiri atas 2 jenis yaitu referensi lvalue yang mengacu pada variabel bernama dan referensi rvalue yang mengacu pada objek sementara. Operator & menandakan referensi lvalue dan && menandakan referensi value atau referensi universal (tidak rvalue atau lvalue).

B. PENULISAN REFERENCE

Untuk menggunakan variabel reference dibutuhkan tanda (&) “Operator Reference”, yang diletakan sebelum identitas dari sebuah variable

Bentuk penulisan
tipeData &nama_reference = variabel;

Contoh penulisan
int &ref = Var;

BAB
11

UNION

A. PENGERTIAN UNION

Union merupakan tipe data yang memungkinkan untuk menyimpan variabel-variabel pada lokasi memori yang sama, dengan kata lain union membagi memori untuk digunakan beberapa variabel. Tipe data union hampir sama dengan tipe data struct yaitu dapat menampung beberapa variabel dengan tipe data berbeda, namun perbedaannya terdapat pada penggunaan memori, union lebih hemat dibandingkan dengan struct. Variabel-variabel yang terdapat pada union jika salah satunya ada yang diubah, maka variabel lain yang disimpan pada memori yang sama juga akan terpengaruh, karena pada union variabel saling bertumpuk satu sama lain.

B. SINTAKS

```
4  union A{
5      int angka;
6      char huruf;
7      float desimal;
8  };
```

Gambar 11.1 Sintaks deklarasi union

Pendeklarasian union sama dengan pendeklarasian struct, diawali dengan union, kemudian diikuti oleh nama union, dan diisi dengan elemen-elemen union (dapat menampung semua jenis tipe data) yang di bungkus dengan kurung kurawal.

ENUM

Pada bahasa C++ tentunya banyak terdapat tipe data salah satunya adalah tipe data enum. Tipe data enum ini berisikan sekumpulan konstanta. Tipe data enum sendiri termasuk kedalam tipe data bentukan, hal ini dikarenakan tipe data enum dibuat oleh programer itu sendiri.

```
3   enum kesulitan {Mudah = 1, Sedang = 2, Sulit = 3};
4   enum jenis_kelamin {Pria, Wanita};
5   enum Game_Mobile {PUBG, Mobile Legend, Ludo};
```

Gambar 12.1 Contoh penulisan enum

Apabila pengguna tidak menentukan pilihan yang telah disediakan maka sistem akan memilihkannya sesuai dengan urutan penulisannya.

Contoh:

enum hari {Senin, Selasa, Rabu, Kamis, Jumat, Sabtu, Minggu};

Maka nilai dari semua konstanta diatas adalah Senin = 0, Selas = 1, Rabu = 2, Kamis = 3, Jumat = 4, Sabtu = 5, Minggu = 6.

A. CONTOH KODE PROGRAM TIPE DATA ENUM C++

Melanjutkan dari contoh enum hari di atas, berikut adalah kode program lengkap untuk menggunakan tipe data enum:

BAB
13

CASTING & COMMA OPERATOR

A. CASTING OPERATOR

Casting Operator atau tipe data casting merupakan tipe data yang bersifat sementara, artinya adalah apabila akan merubah tipe data sebuah variabel hanya dalam operasi aritmatika saja. Setelah operasi aritmatikanya diubah selanjutnya tipe data pada variabel tersebut akan tetap sesuai dengan deklarasinya diawal.

```
1   #include <iostream>
2   #include <conio.h>
3   using namespace std;
4
5   int main() {
6         int a,b,mod;
7         float hasil;
8         cout<<"Contoh penggunaan / dan %\n\n";
9         cout<<"Masukkan angka pertama (a) = ";cin>>a;
10        cout<<"Masukkan angka kedua (b) = ";cin>>b;
11        hasil=a/b;
12        mod=a%b;
13        cout<<"\nHasil pembagian = "<<hasil;
14       cout<<"\nSisa hasil bagi = "<<mod;
15  return 0;
16  }
```

Gambar 13.1 Program deklarasi casting operator

```
Masukkan angka kesatu (a) = 6
Masukkan angka kedua (b) = 7

Hasil pembagian = 0
Sisa hasil bagi    = 6
```

Gambar 13.2 Output program deklarasicasting operator

PREPROCESSOR DIRECTIVES

Preprocessor Directive adalah baris yang termasuk dalam kode program yang didahului oleh tanda hash (#). Baris merupakan arahan untuk preprocessor. Preprosesor memeriksa kode sebelum kompilasi kode yang sebenarnya dimulai dan menyelesaikan semua arahan ini sebelum kode apa pun benar-benar dihasilkan oleh pernyataan reguler.

Arahan preprosesor ini hanya diperluas melintasi satu baris kode. Segera setelah karakter baris baru ditemukan, *Preprocessor Directive* berakhir. Tidak ada titik koma (;) yang diharapkan pada akhir arahan preprocessor. Satu-satunya cara direktif preprosesor dapat meluas melalui lebih dari satu baris adalah dengan mendahului karakter baris baru di akhir baris dengan garis miring terbalik (\).

A. MACRO DEFINITIONS (#DEFINE, #UNDEF)

Untuk mendefinisikan makro preprosesor, dapat menggunakan #define. Sintaksnya adalah:

#define penggantian pengidentifikasi

Ketika preprocessor bertemu dengan *macro definitions*, preprocessor menggantikan setiap kemunculan *identifier* di sisa kode dengan *replacement*. *Replacement* ini bisa berupa ekspresi, pernyataan, blok atau lainnya. Preprocessor tidak mengerti C++ dengan benar, itu hanya menggantikan setiap kejadian pengenal dengan *replacement*.

BAB
15

NAMESPACE C++

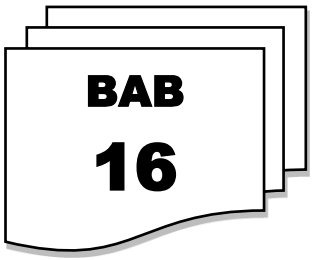
A. PENGERTIAN

Namespace atau dalam bahasa Indonesia bisa juga disebut sebagai ruang nama merupakan suatu ruang lingkup dengan nama yang digunakan sebagai pengelompokkan berbagai entitas seperti *class, variable, object,* dan fungsi. Membangun sebuah *namespace* membutuhkan suatu kata kunci namespace dan dilanjut dengan identitas dari *namespace* tersebut. Pengelompokkan entitas dapat disimpan di dalam tanda kurung kurawal { } setelah nama dari *namespace*. Penulisan umum dari *namespace* adalah sebagai berikut.

```
1  namespace nama_identitas{
2      //daftar entitas
3  }
```

Gambar 15.1 Bentuk Umum Namespace

Penggunaan *namespace* berfungsi sebagai pemisah beberapa entitas dari *global scope* dan mengelompokkannya ke dalam satu nama yang sama. Hal ini dapat mengurangi adanya benturan antara identitas yang berbeda dengan cara pengelompokkan. Karena suatu entitas akan dimasukkan ke dalam satu kelompok, maka suatu entitas tersebut hanya dapat dipanggil melalui nama dari *namespace* tersebut dengan menggunakan *sxope operator* : : atau titik dua sebanyak dua buah. Contoh penulisan pemanggilan entitas dalam *namespace* dan contoh programnya adalah sebagai berikut.



MULTIFILE

A. PENDAHULUAN

Pembuatan program yang besar menggunakan bahasa pemrograman C++ tentu memiliki beberapa perbedaan dengan pembuatan program yang kecil. Cara kerja, alur dan objek-objek di dalam program yang kecil lebih mudah dipahami berbeda dengan program yang besar. Untuk membuat program yang besar dibutuhkan strategi supaya struktur program tersebut lebih mudah dimengerti. Salah satunya adalah dengan membagi program besar tersebut menjadi bagian-bagian yang lebih kecil atau menjadi *multiple file*. Terdapat dua jenis file yang digunakan dalam multifile yaitu file header dan file source. Penggunaan multifile membuat program lebih terstruktur dan mudah dikembangkan. Selain itu, multifile juga memungkinkan kita untuk membuat beberapa library yang dapat digunakan dalam program lain.

B. FILE HEADER

File header atau file yang diakhiri dengan ekstensi ".h" adalah file yang berisi deklarasi fungsi, variabel, dan tipe data yang akan digunakan dalam program. File header digunakan untuk dibagikan ke beberapa file source yang lain, sehingga kita dapat menggunakan fungsi atau variabel yang sama di beberapa file source tanpa harus menuliskan ulang deklarasinya. File header juga digunakan untuk membuat library yang dapat digunakan dalam program lain. Library ini berisi fungsi-fungsi yang sudah ditentukan dan dapat digunakan oleh program lain tanpa harus mengetahui detail implementasinya.

File header juga menyediakan mekanisme untuk mencegah multiple inclusion, yaitu dengan menggunakan preprocessor directive #ifndef dan #endif. Ini digunakan untuk mencegah file header yang sama dari di-include lebih dari sekali. Sebagai contoh, jika kita memiliki file header yang berisi

TYPEDEF

A. PENGERTIAN TYPEDEF

Typedef memiliki fungsi yaitu sebagai memberikan sebuah penamaan tipe data yang isi dari tipe datanya melebihi dari satu jenis tipe data dan biasanya dipadukan dengan struct atau array dengan kata lain bukan membuat tipe data baru tetapi hanya mendefinisikan sebuah nama tipe data yang sudah ada sebelumnya agar tipe datanya dapat digunakan kembali. Typedef ini juga termasuk dalam ADT (Abstract Data Type) dengan pengertian kumpulan beberapa objek yang memuat beberapa operasi dan memiliki fungsi yaitu untuk merepresentasikan kumpulan data yang bermasalah. Tujuan dari penggunaan typedef ini adalah agar mudah diingat dan bisa memasukan seluruh tipe data dalam satu program tidak seperti #define yang memuat satu tipe data saja. Untuk penulisan dari tipe data ini sebagai berikut:

```
typedef tipe_data nama_tipe_data;
```

B. IMPLEMENTASI TYPEDEF

Berdasarkan dari penulisan typedef sebelumnya, berikut adalah contoh pengaplikasiannya secara sederhana sebagai berikut:

```
Typedef unsigned long ulong;
Typedef int nama;
Typedef char NIP[13];
```

BAB
18

TEMPLATE FUNGSI

A. PENGERTIAN TEMPLATE PADA C++

Template adalah fitur C++ yang memungkinkan membuat sebuah program lebih generik. Lebih jelasnya yaitu dapat membuat fungsi atau kelas yang dapat beradaptasi dan bekerja dengan tipe data yang berbeda. Dengan menggunakan template, dimungkinkan untuk membuat kode yang dapat digunakan berulang kali dan fleksibel dengan berbagai tipe data. Template juga memudahkan dalam melakukan pemeliharaan kode.

B. PENULISAN TEMPLATE C++

Bentuk penulisan template yang bisa digunakan adalah sebagai berikut:

```
#include <stdio.h>
#include <conio.h>
#include<iostream>
using namespace std;

template <typename T>
//atau
template <class T>

int main(){

    return 0;

}
```

Gambar 18.1 Penulisan template

AUTO

A. PENGERTIAN AUTO PADA C++

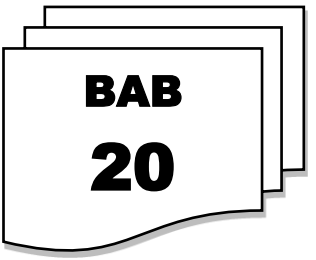
Auto pada C++ merupakan sebuah pintasan tipe data. Auto secara otomatis mendeteksi dan menetapkan tipe data ke variabel yang digunakannya. Kompiler menganalisis tipe data variabel dengan melihat inisialisasinya. Penting untuk menginisialisasi variabel saat mendeklarasikannya menggunakan kata kunci auto. Kata kunci auto memiliki banyak kegunaan, tidak terbatas pada variabel, fungsi, pointer, iterator, template, dan banyak lagi, dengan menggunakan auto sangat berguna untuk menghindari penginisialisasian tipe data yang panjang atau salah ketik sehingga membuat kode menjadi lebih efisien.

Selain itu, pada tipe data auto mendukung pergantian tipe data, jika ekspresi dalam variabelnya berubah, tipe data auto akan menyesuaikannya, termasuk pada hasil pengembalian dari sebuah fungsi yang ekspresi tipe datanya akan dievaluasi pada *runtime.* Penggunaan auto akan memberikan tipe data spesifik yang diperlukan oleh variabel yang dimaksud.

B. DEKLARASI AUTO

Pendeklarasian auto dapat dilakukan dengan beberapa bentuk, yaitu:

a. Inisialisasi universal
   auto a {20};
b. Inisialisasi dengan penugasan
   auto b = 86;
c. Gabungan bentuk universal dan penugasan
   auto c = { 1.945 };
d. Inisialisasi langsung
   auto d(8.234);



ERROR HANDLING

A. PENGERTIAN

Terdapat suatu metode dalam bahasa pemrograman C++ yang dinamakan sebagai *error handling* atau biasa disebut juga sebagai *exception handling*. *Error handling* digunakan untuk mengatasi error atau suatu keadaan yang terjadi ketika sedang menjalankan program. Ketika terdapat *error* saat program dijalankan (*runtime error*), program akan mengarahkan alur eksekusi program ke fungsi khusus yang disebut sebagai fungsi *handler*. Ketika tidak terdeteksi adanya *error* yang menyebabkan *exception*, maka semua *handler* akan diabaikan kemudian program akan berjalan secara normal.

Penggunaan *error handling* dapat memudahkan programmer untuk memisahkan blok kode untuk mengatasi kesalahan dengan blok kode utama dari program yang dibuat. Pemisahan tersebut akan membuat kode program menjadi terlihat lebih rapi dan mudah untuk dibaca. Terdapat tiga kata kunci yang digunakan dalam *error handling* C++ yaitu *throw, catch* dan *try*.

B. PENGGUNAAN METODE THROW, CATCH DAN TRY

Exception dapat dilempar kemana saja dalam blok kode menggunakan pernyataan *throw*. *Operand* dari pernyataan *throw* menentukan jenis *exception* dan dapat berupa bentuk apa pun dan jenis hasil ekspresi akan menentukan tipe *exception* yang dilempar. Berikut adalah contoh *throw exception* ketika pembagian dengan nilai nol terjadi.

QT

A. APA ITU QT

Dalam membuat sebuah aplikasi, *Graphical User Interface* atau GUI merupakan salah satu komponen yang penting dalam sebuah aplikasi. GUI akan memudahkan interaksi antara manusia dengan komputer. Dengan adanya UI, antarmuka sebuah akan jadi lebih menarik. Pada akhirnya, interaksi apapun yang dilakukan oleh pengguna aplikasi dengan mesin menjadi lebih jelas dan terarah.

Qt, adalah sebuah *framework* yang dapat digunakan untuk membuat sebuah GUI pada pemrograman C++. Qt merupakan *framework* yang siap pakai yang bisa memudahkan *programmer* dalam membuat sebuah aplikasi berbasis C++ dan menampilkan program visual. Qt merupakan *framework* yang paling banyak digunakan dan populer untuk proses pengembangan program-program visual di lingkungan Linux dan bersifat *open-source.* Google Earth, VLC Media Player dan Virtual Box merupakan contoh aplikasi yang dikembangkan dengan menggunakan *framework* Qt.

B. INSTALASI QT

Sebelum menggunakan Qt, diharuskan untuk menginstall terlebih dahulu aplikasi yang diperlukannya. Untuk mengunduhnya, bisa dengan mengunjungi https://www.qt.io/download. Panduan Qt yang dibahas pada buku ini merupakan versi Qt 5, sehingga pengguna dapat mengunjungi https://www.qt.io/offline-installers atau bisa juga mengunjungi https://download.qt.io/new_archive/qt/5.4/5.4.1/ jika ingin memiliki versi yang sama dengan buku ini dengan pilih yang paling bawah.

PROFIL PENULIS

Muhammad Taufik Dwi Putra

Penulis merupakan seorang dosen Teknik Komputer Universitas Pendidikan Indonesia. Penulis menyelesaikan pendidikan Magister di Universitas Bina Nusantara, pada tahun 2020. Fokus riset penulis antara lain di bidang *Computer Vision, Image Processing* dan *Deep Learning.*

Munawir

Penulis merupakan dosen di Program Studi Teknik Komputer, Universitas Pendidikan Indonesia. Penulis menyelesaikan pendidikan Doctoral di Kitakyshu University, Japan pada tahun 2020. Fokus riset penulis antara lain di bidang *Urban Informatics, Data Mining* dan Kecerdasan Buatan.

Ana Rahma Yuniarti

Penulis merupakan seorang dosen Teknik Komputer Universitas Pendidikan Indonesia. Penulis menyelesaikan pendidikan Magister di Kumoh National *Institute of Technology*, Korea Selatan pada tahun 2017. Fokus riset penulis antara lain di bidang Biomedis, Kecerdasan Buatan dan *Signal Processing*.

Belajar PEMROGRAMAN LANJUT Dengan C++

Buku yang sangat tepat untuk pembaca yang sedang atau akan mempelajari pemrograman lanjut. Buku ini berisi pelajaran mengenai logika dan algoritma untuk menyelesaikan berbagai masalah yang di eksekusi oleh komputer.

Materi yang dibahas di dalam buku ini meliputi: Pengenalan konsep dasar algoritma dan pemrograman, variabel, nilai, tipe data, operator, percabangan, pengulangan, fungsi string, operasi dasar, larik (array) dan matriks dalam array. Setiap materi dalam buku ini dilengkapi dengan berbagai contoh pemecahan masalah beserta kode program dalam bahasa C++. Bahasa yang digunakan dalam buku ini dirancang agar pembaca dapat mempelajarinya secara mandiri tanpa melibatkan bantuan orang lain.

Esensi dari mempelajari pemrograman dan algoritma adalah merancang suatu solusi untuk menyelesaikan suatu permasalahan yang ada. Maka diperlukan kemampuan penyelesaian masalah menggunakan berbagai pola atau teknik dan logika yang tepat untuk memahami esensi tersebut. Besar harapan bahwa buku ini mampu memberikan ilmu bagi pembacanya supaya dapat mempelajari dasar-dasar pemrograman dengan mengikuti aturan yang benar.

Penerbit widina
www.penerbitwidina.com

ISBN 978-623-459-689-2