BAB II

Mengenal Dev C++ (Dasar Pemrograman C++)

A. Teori dasar C++

Sejarah dan Konsep C++

C++ adalah salah satu dari banyak jenis bahasa pemrograman. Bahasa pemrograman atau yang biasa disebut dengan bahasa komputer, merupakan instruksi standar untuk meng-*command* (memerintah) komputer. Bahasa pemrograman sendiri adalah sebuah himpunan dari aturan sintaks dan semantik yang digunakan untuk mendefinisikan program komputer. Bahasa ini memungkinkan seorang programmer mampu menentukan secara persis data mana yang akan diolah oleh komputer, bagaimana data tersebut akan diteruskan/disimpan, dan jenis langkah apa yang akan dijalankan dalam berbagai macam keadaan (situasi).

Bahasa C++ ini merupakan pengembangan dari bahasa C (yang dikembangkan di Bong laboratorium oleh Dennis Ritchie diawal tahun 1970-an) yang merupakan turunan dari bahasa sebelumnya yaitu bahasa B. Pada awalnya, bahasa tersebut dirancang sebagai bahasa pemrograman yang dijalankan pada sistem unix. Pada perkembangannya, versi ANSI (American National Standart Institute) bahasa pemrograman C menjadi versi dominan, meskipun versi tersebut sekarang jarang dipakai dalam pengembangan sistem dan jaringan maupun untuk sistem embedded.

Bahasa pemrograman C++ sendiri diciptakan pada tahun 1979 ketika Bjarne Stroustrup mengerjakan praktek untuk tesis Ph.D nya. Salah satu bahasa *Stroustrup* memiliki kesempatan untuk bekerja sama dengan adalah bahasa yang disebut Simula, seperti namanya simula adalah bahasa yang terutama dirancang untuk simulasi. Bahasa Simula 67 – yang merupakan varian yang bekerja dengan Stroustrup – dianggap sebagai bahasa pertama yang mendukung paradigma pemrograman berorientasi objek. *Stroustrup* menemukan bahwa paradigma ini sangat berguna untuk pengembangan perangkat lunak, namun bahasa Simula terlalu lambat untuk penggunaan praktis.

Tak lama kemudian, mulai dikerjakan “peng-kelasan C”, yang seperti namanya dimaksudkan untuk menjadi *superset* dari bahasa C. Tujuannya adalah untuk menambahkan pemrograman berorientasi objek ke dalam bahasa C, yang dulu dan masih merupakan bahasa yang dihormati karena portabilitasnya memperhatikan kecepatan atau fungsionalitas tingkat rendah. Bahasa yang diciptakannya termasuk *classes*, *basic inheritance*, *inlining*, argumen fungsi default, dan pemeriksaan tipe yang kuat selain semua fitur bahasa C.

Bahasa C pertama dengan compiler disebut C front, yang diturunkan dari kompiler C yang disebut Cpre. Ini adalah program yang dirancang untuk menerjemahkan C dengan kode Kelas ke C biasa. Hal yang cukup menarik adalah bahwa sebagian besar penulisan ditulis dalam C dengan Kelas, menjadikannya kompiler self-hosting (kompiler yang dapat mengkompilasi sendiri). Cfront kemudian ditinggalkan pada tahun 1993 karena sulit untuk mengintegrasikan fitur-fitur baru ke dalamnya, dengan pengecualian C ++. Meskipun demikian, Cfront membuat dampak besar pada implementasi kompiler masa depan dan pada sistem operasi Unix.

Pada tahun 1983, nama bahasanya diubah dari C dengan peng-kelasan menjadi C++. Operator ++ di bahasa C adalah sebuah operator variabel tambahan, dimana ini memberi penjelasan bahwa *stroustrup* menghormati bahasa tersebut. Banyak keistimewaan baru yang ditambahkan sekarang, yang paling terkenal diantaranya adalah virtual function, function overloading, referensi dengan simbol &, the const keyword dan single line comments menggunakan dua forward slashes (keistimewaan ini diambil dari bahasa BCPL).

Di tahun 1985, *Stroustrup* mempublikasikan "*Bahasa pemrograman C++"*. Di tahun yang sama, C++ diimplementasikan sebagai produk komersial. Bahasa C sendiri belum resmi terstandarisasi, membuat bukunya sangat penting menjadi referensi. Bahasa ini di perbaiki lagi di tahun 1989 untuk memasukkan keamanan dan *static members*, seperti warisan dari beberapa kelas. Pada tahun 1990, TheAnotated C++ Reference Manual dirilis. Di tahun yang sama pula kompiler Turbo C++ Borland akan dirilis sebagai produk komersial. turbo C++ menambahkan plethora dari libraries tambahan dimana mempertimbangkan dampak dari perkembangan C++. Walaupun turbo C++ terakhir yang di rilis tahun 2006, kompiler ini masih banyak digunakan.

Pada tahun 1998, Diterbitkan standar internasional pertama untuk C++ ISO / IEC 14882:1998, Yang secara informal dikenal sebagai C++98. dalam standar yang diterbitkan The Annotated C++ Reference Manual dikatakan memiliki pengaruh besar dalam pengembangan standar tersebut. Pada tahun 2003, komite standar C++ merespon beberapa masalah yang dilaporkan dengan standar mereka yang diterbitkan pada tahun 1998, dan merevisinya. kemudian Bahasa C++ yang direvisi dijuluki Bahasa C++03.

Pada tahun 2005, komite standar C++ merilis laporan teknis (dijuluki TR1) merinci berbagai fitur yang mereka rencanakan untuk menambah fitur yang ada di C++ standar terbaru. Standar baru itu secara informal dijuluki C++0x, standar terbaru tersebut diharap akan dirilis sebelum akhir dekade pertama. Namun, ironisnya, standar baru tersebut belum dirilis sampai pertengahan 2011.

Pada pertengahan 2011, C++ dengan standar baru (dijuluki C++11) selesai dibuat (disetujui oleh ISO/IEC pada 12 Agustus 2011, diterbitkan sebagai 14882:11). Standar ini meningkatkan Library yang ada dalam C++, sehingga standar yang baru dikeluarkan membuat dampak (perubahan) yang besar pada standar C++.

Definisi variabel, library, operator dan manipulator di C++

1. **Variabel**

variabel adalah sebuah tempat untuk menampung data dimemori dimana tempat tersebut dapat menampung nilai (data) yang dapat berubah-ubah selama proses program. atau variabel juga disebut sebagai sebuah identifier yang mempunyai nilai dinamis, arti kata dinamis disini bermaksud bahwa nilai variabel tersebut dapat kita ubah sesuai kebutuhan dalam program. Dari pengertian variabel diatas dapat disimpulkan bahwa fungsi variabel adalah sebagai tempat yang akan digunakan untuk menampung data dimemori dimana tempat tersebut dapat menampung nilai (data) yang dapat berubah-ubah selama proses program.

Terdapat beberapa jenis variabel yang digunakan dalam pemrograman C++, yaitu : Variabel Auto, Variabel Statis, Variabel Register, dan Variabel Eksternal. Klasifikasi ini sebenarnya hanya didasarkan pada ruang penyimpanannya saja.

a**. Variabel auto**, merupakan variabel normal yang dideklarasikan di dalam lingkup (scope) atau blok program tertentu. Variabel jenis ini sebenarnya merupakan nama lain dari variabel lokal. Dengan kata lain, variabel ini hanya akan dikenal dalam suatu blok program saja, misalnya blok pemilihan, pengulangan, maupun fungsi. Meskipun sama dengan variabel lokal, tapi kita bisa saja secara eksplisit menambahkan kata kunci auto di depan pendeklarasiannya. Jenis variabel ini akan dialokasikan di memori pada saat program mengeksekusi badan blok dan didealokasikan secara otomatis ketika eksekusi blok berakhir.

b. Variabel Statis, adalah variabel yang menempati ruang memori kumputer secara permanen, artinya nilai terakhir dari variabel ini akan terus disimpan. Dalam C++, untuk menyatakan variabel statis adalah dengan menggunakan kata kunci static. Pada implementasinya, variabel statis dapat diperankan sebagai variabel global (disebut variabel statis global) maupun variabel lokal (disebut variabel statis lokal)

c. Variabel Statis Lokal, ini diterapkan di dalam suatu fungsi / prosedur, sehingga nama variabelnya hanya akan dikenali di dalam fungsi tempat pendeklarasiannya. Namun perlu diperhatikan bahwa nilai terakhir yang dihasilkan akan terus disimpan. Dengan demikian setiap pemanggilan fungsi yang sama pasti akan memberikan hasil yang berbeda.

2. Library

Library pada C++ merupakan suatu pustaka yang disediakan oleh C++ untuk mempermudah programmer dalam pembuatan suatu program. Library di C++ biasanya di tandai dengan header dengan simbol # di depan nya

contoh #inlcude <iostream.h>

Baris yang ditandai dengan simbol # disebut *Preprocessor directive*. Bertugas untuk mengarahkan prepocessor yang akan digunakan untuk membaca file header atau bisa dikatakan sebagai pengatur proses kompilasi.

a. iostream

Digunakan untuk menampilkan perintah:

- cin

  Fungsi masukan yang digunakan untuk memasukkan data ke suatu variabel. Bentuk umumnya cin>>x;
- cout

  Fungsi keluaran yang digunakan untuk menampilkan data ataupun tulisan. Bentuk umumnya cout<<“tulisan”;
- endl

  Digunakan untuk pindah baris atau enter. Bentuk umumnya cout<<“tulisan”<<endl;
- ends

  Fungsi manipulator yang digunakan untuk menambah karakter null (nilai ASCII NOL) ke deretan suatu karakter. Fungsi ini akan berguna untuk mengirim sejumlah karakter ke file di disk atau modem dan mangakhirinya dengan karakter NULL.

b. conio.h

Digunakan untuk menampilkan perintah:

- getch

  Berfungsi untuk menahan tampilan. Bentuknya umumnya getch();
- clrscr

  Berfungsi untuk membersihkan layar. Bentuk umumnya clrscr;
- getche

  Fungsi yang dipakai untuk membaca sebuah karakter dengan sifat karakter yang dimasukkan tidak perlu diakhiri dengan menekan tombol ENTER, dan karakter yang dimasukan ditampilkan di layar.

- putch

  Akan menampilkan karakter ASCII dari nilai x ke layer monitor tanpa memindahkan letak kursor ke baris berikutnya.
- clreol

  Fungsi ini digunakan untuk membersihkan layar mulai dari posisi kursor hingga kolom terakhir, posisi kursor tiak berubah.
- gotoxy

  Fungsi gotoxy digunakan untuk memindahkan kursor ke kolom x, baris y.
- wherex

  Fungsi wherex digunakan untuk mengembalikan posisi kolom kursor.
- wherey

  Fungsi wherey digunakan untuk mengembalikan posisi baris kursor.
- window

  Fungsi window digunakan untuk mendefinisikan sebuah window berdasarkan koordinat kiri atas dan kanan bawah.\

c. stdio.h

Digunakan untuk menampilkan perintah:

- printf

  Merupakan fungsi keluaran. Bentuk umumnya printf (“tulisan”);
- scanf

  Merupakan fungsi masukan. Bentuk umumnya scanf (“%c”, &karakter);
- gets

  Fungsi inputan yang bisa membaca spasi. Bentuk umumnya gets(var x);

d. string.h

Digunakan untuk menampilkan perintah:

- strcpy

  Digunakan untuk menyailn nilai string.
- strlen

  Dntuk mengetahui panjang string.
- strupr

  Digunakan untuk membuat string menjadi capital.

- strcmp

  Digunakan untuk membandingkan dua buah string. Hasil dari fungsi ini bertipe integer dengan nilai:

  Negative, jika string pertama kurang dari string kedua. Nol, jika string pertama sama dengan string kedua Positif, jika string pertama lebih besar dari string kedua Bentuk umumnya strcmp(string1, string2);
- strlwr

  Digunakan untuk mengubah huruf menjadi kecil semua. Bentuk umumnya strlwr(string);
- strcat

  Digunakan untuk menggabungkan string. Bentuk umumnya strcat(string1,string2);

e. math.h

Digunakan untuk menampilkan perintah:

- sqrt

  Fungsi akar
- pow

  Fungsi pangkat
- sin(), cos(), tan()
- Masing-masing digunakan untuk menghitung nilai sinus, cosinus dan tangens dari suatu sudut. Bentuk umumnya Sin(sudut); Cos(sudut); Tan(sudut);
- max

  Digunakan untuk menghitung hasil pembagian dan sisa pembagian. Bentuk umumnya max(bilangan1, bilangan2);
- min

  Digunakan untuk menentukan bilangan terkecil dari dua buah bilangan. Bentuk umumnya min(bilangan1, bilangan2).

f. windows

Digunakan untuk menampilkan perintah:

- system

  Digunakan untuk memberi warna.

g. iomanip.h

Digunakan untuk menampilkan perintah:

- setiosflags()

  Digunakan untuk mengatur jumlah digit decimal dibelakang koma.

h. stdlib.h

Digunakan untuk menampilkan perintah:

- atof()

  Digunakan untuk mengonfersi nilai string menjadi bilangan bertipe double. Bentuk umumnya atof(char x);
- atoi()

  Digunakan untuk merubah tipe data string menjadi integer.
- pow()

  Digunakan untuk pemangkatan suatu bilangan.Bentuk umum : pow(bilangan, pangkat).

i. float.h

Mendefinisikan konstanta makro menentukan implementasi khusus properti dari floating-point library.

3. Operator

Operator sendiri berarti simbol khusus yang akan memberitahu kepada compiler untuk melakukan operasi aritmatika dan logika tertentu dalam c++ ada beberapa jenis operator terdiri dari

a. Operator aritmatika, berfungsi untuk perhitungan matematika seperti pembagian,perkalian, penambahan, pengurangan berikut adalah tabel dari operator aritmatika.
b. Operator relasional, sebuah operator yang bernilai true dan false. Untuk mengevaluasi antara 2 ekspresi, dapat digunakan operator Relasional. Hasil dari operator ini adalah nilai bool yaitu hanya berupa true atau false, atau dapat juga dalam nilai int, 0 untuk mereprensentasikan “false” dan 1 untuk merepresentasikan “true”.
c. Operator logika, ekuivalen dengan operasi boolean NOT, hanya mempunyai 1 operand, berguna untuk membalikkan nilai dari operand yang bersangkutan dan operator Logika && dan || digunakan untuk mengevaluasi 2 ekspresi dan menghasilkan 1 nilai akhir. mempunyai arti yang sama dengan operator logika Boolean AND dan OR.
d. Operator Bitwase, memodifikasi variabel menurut bit yang merepresentasikan nilai yang disimpan, atau dengan kata lain dalam representasi binary.

e. Operator Assignment(penugasan), di tandai dengan tanda(“=”)
f. Operator Precedence,operator yang didahulukan menentukan pengelompokan istilah dalam ekspresi. misal ada perintah perkalian dan penjumlahan maka akan didahulukan yang perkalian.

4. Manipulator

Manipulator adalah fungsi pembantu yang memungkinkan untuk mengontrol input / output stream. Manipulator pada umumnya digunakan untuk mengatur tampilan layar. Contohnya untuk mengatur supaya suatu nilai ditampilkan dengan lebar 8 karakter dan diatur rata kiri terhadap lebar tersebut.

Dalam C++, terdapat beberapa manipulator yang merupakan fitur baru, yang baru ditambahkan. hal ini karena compiler C++ klasik (belum distandarisasi) tidak mendukung adanya manipulator. Adapun manipulator yang dimaksud disini adalah seperti yang terlihat pada tabel dibawah ini:

Manipulator	**Kegunaan**	**Operasi**
boolalpha	Mengaktifkan flag	Input / Output
dec	Mengaktifkan flag dec	Input / Output
endl	Menampilkan baris baru dan membuat stream	Output
ends	Menampilkan null	Output
fixed	Mengaktifkan flag fixed	Output
flush	Membuang stream	Output
hex	Mengaktifkan flag hex	Input / Output
internal	Mengaktifkan flag internal	Output
left	Mengaktifkan flag left	Output
noboolalpha	Mematikan flag boolalpha	Input / Output
noshowbase	Mematikan flag showbase	Output
noshowpoint	Mematikan flag showpoint	Output
noshowpos	Mematikan flag showpos	Output
noskipws	Mematikan flag skipws	Input

nounitbuf	Mematikan flag unitbuf	Output
nouppercase	Mematikan flag uppercase	Output
oct	Mematikan flag oct	Input / Output
resetiosflags (fmtflags f)	Mematikan flag yang dituliskan (f)	Input / Output
right	Mengaktifkan flag right	Output
scientific	Mengaktifkan flag scientific	Output
setbase (int base)	Mengaktifkan basis base	Input / Output
setfill (int ch)	Mengisi karakter dengan ch	Output
setiodflags (fmtflags f)	Mengaktifkan flag yang dituliskan (f)	Input / Output
setprecision (int p)	Menentukan presisi digit	Output
setw (int w)	Menentukan lebar kolom	Output
showbase	Mengaktifkan flag show base	Output
showpoint	Mengaktifkan flag show point	Output
showpos	Mengaktifkan flag showpos	Output
skipws	Mengaktifkan flag skipws	Input
unitbuf	Mengaktifkan flag unitbuf	Output
uppercase	Mengaktifkan flag uppercase	Output
ws	Mengabaikan karakter white-space	Input

Untuk menggunakan manipulator yang tercantum diatas sobat harus menyertakan file header <iomanip> dan <ionsteam>. Pada kesempatan kali ini kita hanya akan membahas 10 fungsi manipulator yang disediakan oleh Dev-C++, antara lain: endl, ends, setw( ), dec( ), oct( ), hex( ), setbase( ), setfill( ), setprecision( ) dan setiosflags( ).

- **Tipe data**

Pada dasarnya tipe data dibagi menjadi 3 jenis:

1. Tipe Data Angka – untuk angka dan berhubungan dengan aritmetika.
2. Tipe Data Karakter – untuk karakter dan angka bukan untuk operasi aritmetika.
3. Tipe Data Logika – untuk logika benar (*true*) atau salah (*false*).

Pada dasarnya C++ memiliki beberapa tipe data *built-in* yang langsung anda gunakan. Dan berikut 7+ tipe data primitif dalam bahasa pemrograman C++:

Tipe Data	**Keyword**
Boolean	bool
Character	char
Integer	int
Floating point	float
Double	double
Valueless	void
Wide character	wchar_t

Tipe data berdasarkan ukuran Memori dan Jangkauannya

Beberapa tipe data dasar diatas dapat dimodifikasi dengan *type modifiers* berikut −

- signed
- unsigned
- short
- long

Setiap tipe data memiliki besar memori yang berbeda antara satu dengan yang lain. Untuk membuktikannya, anda bisa memeriksanya dengan fungsi kode sizeof()

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

int main() {
   cout << "Size of char : " << sizeof(char) << endl;
   cout << "Size of int : " << sizeof(int) << endl;
   cout << "Size of short int : " << sizeof(short int) << endl;
   cout << "Size of long int : " << sizeof(long int) << endl;
   cout << "Size of float : " << sizeof(float) << endl;
   cout << "Size of double : " << sizeof(double) << endl;
   cout << "Size of wchar_t : " << sizeof(wchar_t) << endl;

   return 0;
}
```

Ketika kode diatas di complie maka kita akan mendapat output sebagai berikut:

```
Size of char : 1
Size of int : 4
Size of short int : 2
Size of long int : 4
Size of float : 4
Size of double : 8
Size of wchar_t : 4
```

Dan disini kami akan merangkum macam-macam tipe variabel, berapa banyak memori yang digunakan dalam memori, dan berapa jangkauan nilai yang mampu digunakan.

Tipe Data	**Ukuran Memori**	**Jangkauan**
char	1 byte	-127 to 127 or 0 to 255
unsigned char	1 byte	0 to 255
signed char	1 byte	-127 to 127
int	4 bytes	-2147483648 to 2147483647
unsigned int	4 bytes	0 to 4294967295
signed int	4 bytes	-2147483648 to 2147483647
short int	2 bytes	-32768 to 32767
unsigned short int	Range	0 to 65,535
signed short int	Range	-32768 to 32767
long int	4 bytes	-2,147,483,648 to 2,147,483,647
signed long int	4 bytes	-2,147,483,648 to 2,147,483,647
unsigned long int	4 bytes	0 to 4,294,967,295
float	4 bytes	+/- 3.4e +/- 38 (~7 digits)
double	8 bytes	+/- 1.7e +/- 308 (~15 digits)
long double	8 bytes	+/- 1.7e +/- 308 (~15 digits)
wchar_t	2 atau 4 bytes	1 wide character

Tipe Data Baru dengan typedef

Dalam Bahasa C++ kita dapat mendeklarasikan variabel menggunakan tipe data kustom yang sebelumnya telah kita buat dengan menggunakan sintaks typedef.

Cara membuatnya seperti ini:

```
typedef tipedata namabaru;
```

Sebagai contoh:

Kita ingin memberi tahu kepada compiler bahwa angka merupakan nama lain dari int.

```
typedef int angka;
```

Maka sekarang anda bisa menggunakan angka sebagai tipe data baru anda:

```
angka jarak;
```

Enumerasi dalam C++

Pada C++, enum biasa digunakan saat suatu data yang sudah kita ketahui jumlahnya, dan tidak banyak. Seperti halnya untuk menyatakan beberapa hal berikut:

- Nama hari,
- Nama bulan,
- Jenis kelamin, dan lainnya.

Berikut cara penggunaannya:

```
enum bulan{Januari, Februari, Maret, April, Mei, Juni, Juli, Agustus, September, Oktober, November, Desember};
```

Secara default, nilai dari nama pertama adalah 0, nama kedua adalah 1, dan seterusnya. Namun anda bisa memberikan sebuah nilai sepesifik, dengan menambahakn identifier.

Contoh:

```
enum color { red, green = 5, blue};
```

Sehingga red bernilai 0 (tanpa identifier, selalu mulai dari nol), gree bernilai 5, dan blue bernilai 6 karena setiap nama akan selalu 1 tingkat lebih besar dari nama sebelumnya.

Berikut cara penggunaannya:

```
enum color { red, green, blue };
c = blue;
```

#1 Tipe Data Boolean (bool)

Boolean adalah salah satu tipe data yang hanya memiliki dua pilihan yaitu True (1) atau False (0). Tipe data ini biasanya digunakan untuk memberikan kondisi pada program atau bisa juga memastikan kebenaran dari sebuah operasi. Besarnya memori yang dibutuhkan tipe data bool yaitu 1 byte atau 8 bit. Berikut ini contoh program C++ menggunakan tipe data bool:

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

int main()
{
  int angka;
  bool hasil;
  cout << "Masukan angka = "; cin >> angka;
  hasil = angka > 10;
  cout << hasil;
}
```

Pada contoh program diatas, kita menggunakan 2 buah variabel yaitu variabel angka dengan tipe data integer, dan variabel hasil dengan tipe data boolean. Nah, disini saya akan mengambil nilai/value untuk variabel hasil dengan membandingkan nilai pada variabel angka terhadap bilangan 10. Apabila nilai pada variabel angka lebih dari 10 maka hasil bernilai 1 (true) dan jika angka lebih kecil dari 10 maka hasilnya bernilai 0 (false).

#2 Tipe Data Character (char)

Character adalah salah satu tipe data yang memungkinkan kita untuk memesan memori berformat text (huruf, angka, dan simbol) dengan karakter tunggal. Besarnya memori yang dibutuhkan tipe data **char** yaitu 1 byte atau 8 bit. Berikut ini contoh program C++ menggunakan tipe data char:

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

int main(){
  char nilai;

  cout << "Masukan nilai (A/B/C/D): "; cin>>nilai;
  cout << "Nilai anda:" << nilai;

}
```

Perlu diingat bahwa tipe data char hanya dapat menyimpan data berbentuk karakter dan hanya satu karakter, oleh karena itu apabila anda memasukan lebih dari 1 karakter maka nilai yang akan tersimpan hanya karakter pertama.

#3 Tipe Data Integer (int)

Integer adalah salah satu tipe data numerik yang memungkinakan kita untuk menyimpan data dalam bentuk bilangan bulat. Besarnya memori yang dibutuhkan tipe data int yaitu 4 byte atau 32 bit. Berikut ini contoh program C++ menggunakan tipe data int:

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

int main(){
  int x,y,z;
  x=3; y=4;

  z=x*y;
  cout << "Hasil perkalian: " << z;
}
```

Dengan menggunakan tipe data integer hal ini memungkinkan kita untuk melakukan sejumlah operasi aritmetika seperti perkalian dan lain sebagainya.

Pada contoh diatas, saya menggunakan 3 buah variabel beripe integer sebagai berikut: X bernilai 3, Y bernilai 4, dan Z sebagai hasil hasil perkalian x dan y.

#4 Tipe Data Floating Point (float)

Floating Point adalah tipe data numerik yang memungkinkan untuk menyimpan nilai dalam memori bersifat bilangan pecahan atau real, maupun eksponensial. Besarnya memori yang dibutuhkan tipe data float yaitu 4 byte atau 32 bit. Berikut ini contoh program C++ menggunakan tipe data float:

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

int main(){
  float jari, hasil ;
  const float p=3.14;

  cout << "Masukan Jumlah jari-jari = "; cin >> jari;
  hasil = (jari * p) * 2;
```

```
  cout << "Keliling dari Lingkaran adalah " << hasil;

}
```

#5 Tipe Data Double Floating Point (double)

Double Floating Point sama seperti float yaiut salah satu tipe data yang bersifat menyatakan bilangan pecahan atau real, maupun eksponensial. Bedanya adalah penyimpanan angka masimal lebih besar daripada float dan otomatis double juga akan membutuhkan memori yang lebih besar. Besarnya memori yang dibutuhkan tipe data double yaitu 8 byte atau 64 bit. Berikut ini contoh program C++ menggunakan tipe data double:

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

int main(){
  double jari, hasil ;
  const double p=3.1428;

  cout << "Masukan Jumlah jari-jari = "; cin >> jari;
  hasil = jari*(jari * p);
  cout << "Luas lingkaran: " << hasil;
}
```

#6 Tipe Data String (string)

String merupakan tipe data *text* (huruf, angka, dan simbol) yang memungkinkan kita menyimpan nilai dengan bentuk text, kumpulan dari character. Besarnya memori yang dibutuhkan tipe data string yaitu 4 byte atau 32 bit. Berikut ini contoh program C++ menggunakan tipe data string:

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

int main(){
  string nohp;

  cout << "Masukan nomor HP: "; cin >> nohp;
  cout << "Nomor HP anda: " << nohp;
}
```

Sama seperti halnya tipe data char, dalam tipe data string kita bisa menggunakan karakter dan angka dengan ketentuan tidak dapat dilakukan operasi aritmetika. Namun perbedaannya, jika dalam tipe data char kita hanya mampu menyimpan nilai satu karakter untuk tiap variabel, hal

ini tidak berlaku pada tipe data string. Hal ini di karenakan String merupakan tipe data composite.

#7 Tipe Data Valueless (void)

Valueless adalah salah satu tipe data yang berarti "tidak ada" atau "tidak mempunyai tipe data". Namun disini kita belum akan membahasnya lebih detail. Void termasuk katagori tipe data namun kita tidak bisa menggunakanya pada variabel biasa, void biasanya digunakan pada function yang tidak mempunyai return value. Besarnya memori yang dibutuhkan tipe data void yaitu 1 byte atau 8 bit.

- **Format penulisan konstanta dan variabel**

a. Konstanta

Konstanta adalah jenis identifier yang bersifat konstan atau tetap, artinya nilai dari konstanta di dalam program tidak dapat dirubah / bersifat tetap. konstanta juga sering dianggap seperti variabel, namun nilainya tetap / tidak dapat diubah-ubah

Konstanta berguna untuk menentukan nilai yang merupakan tetapan, misalnya nilai pi (π), kecepatan cahaya, ketetapan gravitasi dan lainnya. Dengan mendefinisikan konstanta yang bersifat global, maka kita dapat menggunakannya di setiap bagian program. Ada dua cara untuk mendeklarasikan konstanta, yaitu :

Menggunakan kata kunci const

Dalam C++ kita dapat mendefinisikan sebuah konstanta dengan menggunakan kata kunci const. Berikut ini adalah bentuk umumnya:

const tipe_data nama_konstanta = nilai_tetapan;

Contoh pendeklarasian konstanta dengan kata kunci const adalah sebagai berikut:

const int lebar = 100;

const char tab = '\t'

const zip = 1212;

Perlu diketahui konstanta yang terdiri dari satu karakter (char) atau dikenal sebagai konstanta karakter harus diawali dan diakhiri dengan tanda petik tunggal(') sedangkan konstanta yang terdiri dari beberapa karakter (string) atau dikenal dengan konstanta string harus diawali dan diakhiri dengan tanda petik ganda(").

Menggunakan Preprocessor Directive #define

Kita dapat menggunakan preprocessor directive #define untuk mendefinisikan sebuah konstanta. Penggunaan #define sendiri memiliki kelebihan apabila dibandingkan dengan const. Yaitu pada sisi kecepatan kompilasi, karena sebelum kompilasi dilakukan, kompiler pertama kali mencari symbol #define (oleh karena itu mengapa "#" dikatakan prepocessor directive), sehingga dari segi kompilasi penggunaan #define lebih cepat dibanding const.

Berikut adalah bentuk umum penggunaan #define

#define nama_konstanta nilai_tetapan;

Contoh pendeklarasian konstanta dengan #define adalah sebagai berikut:

#define Newline '\n'

#define phi 3.14159265

#define lebar 100

Pendeklarasian dengan #define tanpa diperlukan adalanya tanda = untuk memasukkan nilai kedalam pengenal dan juga tanpa diakhiri tanpa semicolon.

b. Variabel

Variabel Adalah sebuah tempat untuk menampung data dimemori dimana tempat tersebut dapat menampung nilai (data) yang dapat berubah-ubah selama proses program. atau variabel juga disebut sebagai sebuah identifier yang mempunyai nilai dinamis, arti kata dinamis disini bermaksud bahwa nilai variabel tersebut dapat kita ubah sesuai kebutuhan dalam program.

Dari pengertian variabel diatas dapat disimpulkan bahwa fungsi variabel adalah sebagai tempat yang akan digunakan untuk menampung data dimemori dimana tempat tersebut dapat menampung nilai (data) yang dapat berubah-ubah selama proses program.

Terdapat beberapa **jenis variabel** yang digunakan dalam pemrograman C++, yaitu : Variabel Auto, Variabel Statis, Variabel Register, dan Variabel Eksternal. Klasifikasi ini sebenarnya hanya didasarkan pada ruang penyimpanannya saja.

1. Variabel Auto

Variabel auto sebenarnya merupakan variabel normal yang dideklarasikan di dalam lingkup (scope) atau blok program tertentu. variabel jenis ini sebenarnya merupakan nama lain dari variabel lokal. Dengan kata lain, variabel ini hanya akan dikenal dalam suatu blok program saja, misalnya blok pemilihan, pengulangan, maupun fungsi. Meskipun sama dengan variabel lokal, tapi kita bisa saja secara eksplisit menambahkan kata kunci auto di depan pendeklarasiannya. Jenis variabel ini akan dialokasikan di memori pada saat program mengeksekusi badan blok dan didealokasikan secara otomatis ketika eksekusi blok berakhir.

Perhatikan contoh kode dibawah ini:

```
{
int A;
auto int B;
…..
}
```

Pada potongan kode diatas, variabel A dan B sama-sama merupakan variabel lokal / variabel auto yang hanya dikenal di dalam blok program bersangkutan.

2. Variabel Statis

Variabel statis adalah variabel yang menempati ruang memori kumputer secara permanen, artinya nilai terakhir dari variabel ini akan terus disimpan. Dalam C++, untuk menyatakan variabel statis adalah dengan menggunakan kata kunci **static**.

Bentuk umum pendeklarasian variabel statis adalah sebagai berikut:

static tipe_data nama_variabel;

Contoh:

```
static int ABC;
static char DEF;
```

Pada implementasinya, variabel statis dapat diperankan sebagai variabel global (disebut variabel statis global) maupun variabel lokal (disebut variabel statis lokal)

Variabel Statis Lokal

Variabel statis lokal ini diterapkan di dalam suatu fungsi / prosedur, sehingga nama variabelnya hanya akan dikenali di dalam fungsi tempat pendeklarasiannya. Namun perlu diperhatikan bahwa nilai terakhir yang dihasilkan akan terus disimpan. Dengan demikian setiap pemanggilan fungsi yang sama pasti akan memberikan hasil yang berbeda.

Variabel Statis Global

Dalam pembuatan program dengan bahasa C++, kita diizinkan untuk melakukan pembuatan fungsi-fungsi dalam file yang terpisah dari program utama. Untuk kasus-kasus tertentu dimana variabel statis lokal tidak dapat digunakan, kita dapat membuat file terpisah yang mempunyai variabel statis global.

Sebagai gambaran, perhatikanlah contoh penggunaannya dalam potongan kode program dibawah ini.

```
// Mendeklarasikan variabel statis global
// Dengan nama “A” static int A;
void set_NilaiA (int A);
int get_NilaiA ();
void set_NilaiA (int aa){
A = aa;
}
int get_NilaiA (){
A = A + 10;
return A;
}
```

3. Variabel Register

Tidak seperti variabel biasa yang berada di memori, variabel register ini akan disimpan di register CPU. Dengan demikian, untuk mengisikan atau mengubah nilai dari variabel register tentunya tidak memerlukan akses memori sehingga prosesnya juga akan lebih cepat. Dalam C++, variabel register hanya dapat diisi oleh tipe data **char**, **int** dan **pointer** saja serta hanya boleh dideklarasikan sebagai variabel lokal ataupun parameter dari sebuah fungsi. Untuk mendeklarasikan variabel register, kita harus menggunakan kata kunci **register**.

Bentuk umum dari pendeklarasian variabel register adalah sebagai berikut:

```
register tipe_data nama_variabel;
```

Berikut ini adalah contoh program yang akan menunjukkan cara mendeklarasikan dan mengubah variabel register.

```
#include <iostream>
using namespace std;
// Membuat fungsi untuk menghitung M pangkat eint Hitung(register int M, register int e)
        {
        register int temp;
        temp = 1;
        for (;e;e–){
        temp = temp * M;
        }
        return temp;
        }
        // Fungsi Utama
        int main(){
        int MD;
        MD = Hitung(4,5); // 4 pangkat 5
        cout<<MD<<endl;
        return 0;
        }
```

4. Variabel Eksternal

Variabel eksternal adalah variabel global yang ada atau sudah dideklarasikan di dalam file lain. variabel jenis ini biasa digunakan apabila program yang kita tulis berjumlah lebih dari satu file. Cara membuat variabel eksternal adalah dengan menyertakan kata kunci **extern** di depan deklarasi variabel bersangkutan.

Variabel Eksternal 1.cpp

```
#include <iostream>
using namespace std;
int a;
// Mendeklarasikan prosedur eksternal extern void TulisNilai();
int main(){
a = 99;
TulisNilai();
return 0;
}
```

- **Input dan Output pada C++**

Input Dan Output pada C++ – Pada bahasa C operasi input dan output dilakukan dengan memakai fungsi-fungsi yang ada di header file "stdio.h". contohnya untuk input dan output ke layar monitor digunakan perintah seperti scanf, printf, putch, dll. Untuk input dan output ke file digunakan perintah seperti fwrite, fread, fputc, dll.

Sedangkan bahasa pemrograman C++ memiliki teknik input dan output (I/O) yang baru, yaitu : menggunakan stream. Header file untuk input dan output stream adalah "iostream.h" dan beberapa file lain, seperti fstreamm.h, strstream.h dan constream.h.

Stream adalah suatu perintah logika (logikal device) yang berguna untuk mendapatkan atau memberikan informasi.Stream akan dihubungkan dengan perangkat fisik (misalnya keyboard, screen / layar, maupun printer) melalui sistem input dan output (I/O). Semua stream mempunyai perilaku yang sama, sehingga suatu fungsi I/O dapat dioperasikan ke peralatan fisik yang berbeda. Sebagai contoh, jika kita akan melakukan penulisan data, maka cara yang digunakan untuk menuliskan ke layar maupun ke printer adalah sama. Dalam bahasa C++ untuk melakukan hal-hal yang berhubungan dengan proses input dan output data digunakan file header "iostream.h" (dalam bahasa C digunakan file header stdio.h)

Pada saat program C++ memulai proses eksekusi, terdapat empat buah stream yang secara otomatis akan terbuka, yaitu seperti yang terlihat pada tabel dibawah ini.

Nama Stream	Kegunaan	Peralatan Standar
cin	Input standar	Keyboard
cout	Output standar	Layar (Screen)
cerr	Kesalahan output standar	Layar (Screen)
clog	Cerr yang ter-buffer melalui file log	Layar (Screen)

Atau Stream dapat diartikan sebagai nama umum untuk menampung aliran data (contoh : file, mouse, keyboard), maupun untuk keluaran (contoh : printer, layer). Dalam C++ input berarti membaca dari stream dan output berarti menulis ke stream.

Apa itu Input ?

Perintah masukan / input adalah perintah yang berfungsi untuk mamasukan data pernyataan kedalam memori program, yang biasanya akan diproses dan dikeluarkan dalam bentuk peintah keluaran (output) atau kemungkinan diolah/diproses terlebih dahulu sebelum dikeluarkan. Perintah standar yang disediakan di bahasa pemrograman C++ untuk melakukan input adalah cin() sedangkan dalam bahasa C adalah scanf().

Bentuk umum Input Operator C++ :

cin>>variable;

Apa itu Output ?

Perintah keluaran / output adalah perintah yang berfungsi untuk menampilkan pernyataan sehingga muncul ke layar / hasil consol program. Perintah standar yang disediakan di bahasa pemrograman C++ adalah cout() sedangkan dalam bahasa C adalah printf().

Bentuk Umum Output Operator C++ :

cout<<ekspresi;

Dalam bahasa pemrograman C++ kita dapat menggunakan Escape Sequences untuk merepresentasikan sebuah karakter yang tidak ada dalam tradisional symbol. Berikut beberapa contohnya:

- \n : baris baru / linefeed
- \” : petik ganda
- \b : back space

- **Jenis manipulator di C++**

Manipulator adalah fungsi pembantu yang memungkinkan untuk mengontrol input / output stream. Manipulator pada umumnya digunakan untuk mengatur tampilan layar. Contohnya untuk mengatur supaya suatu nilai ditampilkan dengan lebar 8 karakter dan diatur rata kiri terhadap lebar tersebut.

Dalam C++, terdapat beberapa manipulator yang merupakan fitur baru, yang baru ditambahkan. hal ini karena compiler C++ klasik (belum distandarisasi) tidak mendukung adanya manipulator. Adapun manipulator yang dimaksud disini adalah seperti yang terlihat pada tabel dibawah ini:

Manipulator	**Kegunaan**	**Operasi**
boolalpha	Mengaktifkan flag	Input / Output
dec	Mengaktifkan flag dec	Input / Output
endl	Menampilkan baris baru dan membuat stream	Output
ends	Menampilkan null	Output
fixed	Mengaktifkan flag fixed	Output
flush	Membuang stream	Output
hex	Mengaktifkan flag hex	Input / Output
internal	Mengaktifkan flag internal	Output
left	Mengaktifkan flag left	Output
noboolalpha	Mematikan flag boolalpha	Input / Output
noshowbase	Mematikan flag showbase	Output
noshowpoint	Mematikan flag showpoint	Output
noshowpos	Mematikan flag showpos	Output
noskipws	Mematikan flag skipws	Input
nounitbuf	Mematikan flag unitbuf	Output
nouppercase	Mematikan flag uppercase	Output
oct	Mematikan flag oct	Input / Output
resetiosflags (fmtflags f)	Mematikan flag yang dituliskan (f)	Input / Output

right	Mengaktifkan flag right	Output
scientific	Mengaktifkan flag scientific	Output
setbase (int base)	Mengaktifkan basis base	Input / Output
setfill (int ch)	Mengisi karakter dengan ch	Output
setiodflags (fmtflags f)	Mengaktifkan flag yang dituliskan (f)	Input / Output
setprecision (int p)	Menentukan presisi digit	Output
setw (int w)	Menentukan lebar kolom	Output
showbase	Mengaktifkan flag show base	Output
showpoint	Mengaktifkan flag show point	Output
showpos	Mengaktifkan flag showpos	Output
skipws	Mengaktifkan flag skipws	Input
unitbuf	Mengaktifkan flag unitbuf	Output
uppercase	Mengaktifkan flag uppercase	Output
ws	Mengabaikan karakter white-space	Input

Untuk menggunakan manipulator yang tercantum diatas sobat harus menyertakan file header <iomanip> dan <ionsteam>. Pada kesempatan kali ini kita hanya akan membahas 10 fungsi manipulator yang disediakan oleh Dev-C++, antara lain: endl, ends, setw( ), dec( ), oct( ), hex( ), setbase( ), setfill( ), setprecision( ) dan setiosflags( ).

endl

endl adalah sebuah fungsi manipulator yang berguna untuk memasukkan karakter NewLine atau mengatur pindah baris, dengan kata lain fungsi manipulator ini serupa dengan “\n”. Fungsi endl sangat dibutuhkan untuk piranti keluaran berupa file di disk. Untuk menggunakan manipulator ini, sobat harus menyertakan file header <iostream>.

ends

ends adalah sebuah fungsi manipulator yang berguna untuk menambah karakter null (nilai ASCII NOL) kederetan suatu karakter. Fungsi seperti ini seringkali diperlukan, misalnya untuk mengirim sejumlah karakter ke file di disk atau modem dan mengakhirinya dengan karakter NULL. Untuk menggunakan manipulator ini, sobat harus menyertakan file header <iostream>.

setw( )

setw( ) adalah sebuah fungsi manipulator yang berguna untuk mengatur lebar dari suatu tampilan data. Seandainya sobat akan menggunakan manipulator ini, sobat harus menyertakan file header <iomanip>. Bentuk umum penulisan setw() adalah sebgai berikut: setw(int n); n = adalah nilai lebar tampilan data, yang bernilai integer.

dec( ), oct( ) dan hex( )

dec, oct dan **hex** adalah sebuah fungsi manipulator yang berguna untuk memunculkan data dalam bentuk hexadesimal (bilangan berbasis 16), oktal (bilangan berbasis 8) dan desimal (bilangan berbasis 10). Seandainya sobat akan menggunakan manipulator ini, sobat harus menyertakan file header <iomanip>.

setbase( )

setbase( ) adalah sebuah fungsi manipulator yang berguna untuk konversi bilangan desimal, oktal dan hexadesimal. Seandainya sobat akan menggunakan manipulator setbase(), sobat harus menyertakan file header <iomanip>. Bentuk penulisannya setbase() adalah sebgai berikut:

setbase(base bilangan);

setfill( )

setfill( ) adalah sebuah fungsi manipulator yang berguna untuk menampilkan suatu karakter yang diletakkan didepan nilai yang diatur oleh fungsi setfill(). Untuk menggunakan manipulator setfill(), sobat harus menyertakan file header <iomanip>. Bentuk penulisannya setfill() adalah sebgai berikut:

setfill(charakter);

setprecision( )

setprecision( ) adalah sebuah fungsi manipulator yang berguna untuk mengatur jumlah digit desimal yang ingin ditampilkan. biasanya setprecision() digunakan jika sobat bekerja dengan menggunakan bilangan pecahan, dengan setprecision() sobat dapat mengatur jumah digit pecahan yang ingin ditampilkan, Untuk menggunakan manipulator setprecision(), sobat harus menyertakan file header <iomanip>

setiosflags( )

setiosflags( ) adalah sebuah fungsi manipulator yang dipakai untuk mengatur berbagai format keluaran data, Untuk menggunakan manipulator setiosflags(), sobat harus menyertakan file header <iomanip>. Terdapat berbagai format keluaran (outout) untuk fungsi setiosflags(), diantaranya adalah sebagai berikut:

Tebel Tanda format untuk menampilkan setiosflags() dan resetiosflags()

Tanda Format	Keterangan
ios::right	Menyetel rata kanan pada lebar field yang diatur melalui setw()
ios::left	Menyetel rata kiri pada lebar field yang diatur melalui setw()
ios::fixed	Memformat keluaran dalam bentuk notasi desimal
ios::scientific	Memformat keluaran pada notasi eksponensial
ios::oct	Memformat keluaran pada basis 8 (oktal)
ios::hex	Memformat keluaran pada basis 16 (heksadesimal)
ios::dec	Memformat keluaran pada basis 10 (desimal)
ios::uppercase	Memformat huruf dalam notasi heksadesimal pada bentuk huruf kapital
ios::showpoint	Menampilkan titik desimal dalam bilangan pecahan yang tidak mempunyai bagian pecahan
ios::showbase	Menampilkan awalan 0x bagi bilangan heksadesimal atau 0 (nol) bagi bilangan oktal
ios::showpos	Untuk menampilkan tanda + pada bilangan positif

- **Percabangan**

Salah satu permasalahan yang pasti akan dijumpai dalam pembuatan program adalah suatu percabangan. Percabangan yang dimaksud di sini tidak lain adalah suatu pemilihan statemen yang akan di eksekusi dimana pemilihan tersebut berdasarkan kondisi tertentu. Di dalam C++, terdapat 2 jenis struktur blok (blok program) yang digunakan untuk mengimplementasikan suatu percabangan, yaitu dengan menggunakan struktur if dan struktur switch. Struktur if sendiri pada artikel ini akan dibagi menjadi 4 yaitu : pernyataan if satu kondisi, pernyataan if dua kondisi / if-else, pernyataan if lebih dari dua kondisi / if-else majemuk dan pernyataan if bersarang / nested if.

Statement-statement yang ada dalam sebuah blok percabangan akan dieksekusi hanya jika kondisi yang didefinisikan bernilai benar (terpenuhi). Artinya jika kondisi bernilai salah (tidak terpenuhi), maka statemen-statemen tersebut tidak akan dieksekusi atau akan diabaikan oleh compiler.

Untuk memahami konsep percabangan, perhatikan kalimat dibawah ini:
“Jika Budi mendapat ranking satu maka Budi akan dibelikan sepatu baru”

Coba sobat amati, pada kalimat diatas yang merupakan kondisi adalah mendapat ranking satu. Pada kasus ini sepatu baru hanya akan dibeli jika Budi mendapat ranking satu. Sebaliknya, jika tidak mendapat ranking satu, maka sepatu baru pun tidak akan dibelikan.

5 Macam Operasi Percabangan C++

1. Pernyataan IF Satu Kondisi

Seperti yang sudah kita ketahui, pernyataan percabangan dipakai untuk memecahkan persoalan dengan cara mengambil suatu keputusan dari berbagai pernyataan yang ada. Untuk keperluan pengambilan keputusan, Dev-C++ menyediakan beberapa perintah salah satunya adalah if satu kondisi. Pernyataan if satu kondisi mempunyai pengertian, "Jika kondisi bernilai benar, maka perintah akan dikerjakan dan jika kondisi bernilai salah, maka perintah akan diabaikan".

- Jika kondisi bernilai benar, maka perintah akan dikerjakan.
- Jika kondisi bernilai salah, maka perintah tidak akan dikerjakan

Pengertian tersebut dapat dicerminkan melalui diagram alir berikut ini:

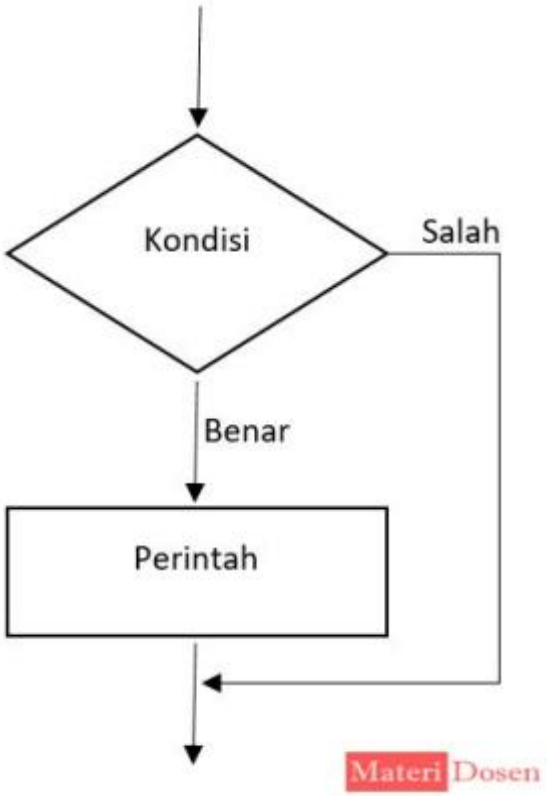


Diagram alir pernyataan if satu kondisi

Struktur If Satu Kondisi

Struktur if satu kondisi merupakan struktur yang paling sederhana karena hanya melibatkan sebuah ekspresi yang akan diperiksa.

Bentuk umum dari struktur percabangan if satu kondisi adalah sebagai berikut:

```
// Jika terdapat lebih dari satu pernyataan / statemen
if (kondisi){
   Statemen/Pernyataan;
   Statemen/Pernyataan;
}
// Jika hanya terdapat satu statemen
// Dapat ditulis sebagai berikut
     if (kondisi) Statemen;
```

2. Pernyataan IF Dua Kondisi

Struktur percabangan if dua kondisi / if-else sedikit lebih kompleks bila dibandingkan dengan struktur if yang hanya memiliki satu buah kondisi. Sebenarnya konsepnya juga sederhana, yaitu pada struktur jenis ini terdapat sebuah statemen khusus yang berfungsi untuk mengatasi kejadia apabila kondisi yang didefinisikan tersebut bernilai salah (tidak terpenuhi). Artinya dalam pernyataan if dua kondisi "Jika kondisi bernilai benar, maka perintah-1 akan dikerjakan dan jika kondisi bernilai salah (tidak terpenuhi) maka akan mengerjakan perintah-2".

Dari pengertian tersebut dapat dicerminkan melalui diagram alir berikut ini.

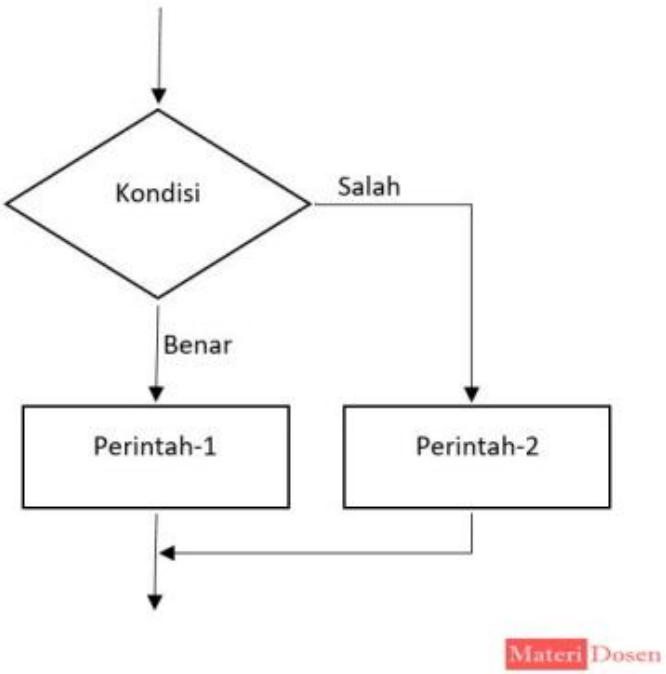


diagram alir pernyataan if dua kondisi

Struktur If Dua Kondisi (If Else)

Bentuk umum dari struktur percabangan dengan dua kondisi adalah sebagai berikut:

```
if (kondisi){
Statemen_jika_kondisi_terpenuhi;
}
else{
Statemen_jika_kondisi_tidak_terpenuhi;
}
```

3. Pernyataan IF Lebih Dari Dua Kondisi

Struktur percabangan if lebih dari dua kondisi / if-else majemuk merupakan struktur percabangan yang biasanya membingunggkan para programmer pemula. Percabangan If-else majemuk sebenarnya merupakan pengembangan dari struktur if dua kondisi, karena percabangan ini akan menambahkan (menyisipkan) satu atau lebih kondisi ke dalamnya.

Artinya dalam pernyataan if lebih dari dua kondisi: Jika kondisi1 bernilai benar, maka perintah-1 akan dikerjakan, jika kondisi1 salah maka akan mengecek kondisi2 dan jika kondisi2 bernilai benar, maka perintah-2 akan dikerjakan, jika kondisi2 juga salah maka akan mengecek kondisi berikutnya dan akan mengerjakan perintah pada struktur blok yang memiliki kondisi bernilai benar. Seandainya tidak ada kondisi yang bernilai benar, maka akan mengerjakan perintah yang berada pada struktur blok else. untuk lebih jelasnya mari kita perhatikan struktur if-else majemuk dibawah ini.

Struktur If Lebih Dari Dua Kondisi / If-Else Majemuk

Bentuk umum dari struktur percabangan If yang memiliki lebih dari dua kondisi adalah sebagai berikut:

```
if(kondisi1){
Statemen_jika_kondisi1_terpenuhi;
}
else if(kondisi2){
Statemen_jika_kondisi2_terpenuhi;
}
else if(kondisi3){
Statemen_jika_kondisi3_terpenuhi;
}
else if(kondisi4){
Statemen_jika_kondisi4_terpenuhi;
}
…..
else{
Statemen_jika_semua_kondisi_tidak_terpenuhi;
}
```

4. Pernyataan IF Bersarang

Struktur percabangan if bersarang / nested if merupakan struktur if yang paling komplek, karena merupakan perluasan dan kombinasi dari berbagai struktur if lainnya. Konsep dari percabangan ini adalah terdapat Struktur If yang berada didalam Struktur If lainnya. Artinya dalam pernyataan If bersarang jika kondisi If yang paling luar (paling atas) bernilai benar, maka kondisi If yang berada didalamnya baru akan dilihat (di cek).

Struktur If Bersarang / Nested If

Bentuk umum dari struktur If bersarang / struktur if yang berada di dalam struktur if lainnya adalah sebagai berikut:

```
if(kondisi1){
if(kondisi1a){
Statemen_jika_kondisi1_dan_1a_terpenuhi;
}
else if(kondisi1b){
Statemen_jika_kondisi1_dan_1b_terpenuhi;
}
…..
else{
Statemen_jika_hanya_kondisi1_yang_terpenuhi;
}
}
else if(kondisi2){
if(kondisi2a){
Statemen_jika_kondisi2_dan_2a_terpenuhi;
}
else if(kondisi2b){
Statemen_jika_kondisi2_dan_2b_terpenuhi;
}
…..
else{
Statemen_jika_hanya_kondisi2_yang_terpenuhi;
}
}
else if(kondisi3){
Statemen_jika_kondisi3_terpenuhi;
}
…..
else{
Statemen_jika_semua_kondisi_tidak_terpenuhi;
}
```

5. Pernyataan Switch-Case

Selain menggunakan pernyataan If, C++ juga menawarkan kepada kita untuk dapat melakukan percabangan (pemilihan) dengan menggunakan pernyataan Switch-Case. Sama seperti pernyataan If-Else, pernyataan Switch-Case juga merupakan pernyataan yang digunakan untuk menjalankan salah satu pernyataan dari beberapa kemungkinan pernyataan, Namun penggunaan pernyataan Switch-Case lebih sempit, karena perintah ini haya digunakan untuk memeriksa data yang bertipe integer atau karakter.

Struktur Switch-Case

Bentuk umum penggunaan pernyataan Switch-Case adalah sebagai berikut:

```
switch(ekspresi){
case nilai_konstanta1:
Statemen_atau_Perintah;
break;
case nilai_konstanta2:
Statemen_atau_Perintah;
break;
case nilai_konstanta3:
Statemen_atau_Perintah;
break;
…..
case nilai_konstantaN:
Statemen_atau_Perintah;
break;
default:
Statemen_alternatif;
}
```

Seperti yang sudah disinggung diatas, tipe data dari nilai_konstanta pada struktur pernyataan switch-case harus berupa tipe ordinal, seperti bilangan bulat atau karakter. Selain itu statemen default: pada struktur switch-case berguna untuk mengeksekusi statemen alternatif, yaitu jika nilai yang kita masukkan ternyata tidak sesuai dengan nilai-nilai konstanta yang telah didefinisikan. Sedangkan statement break pada struktur switch-case digunakan untuk menunjukan bahwa perintah siap keluar dari struktur switch-case. Jika pernyataan break tidak ada, maka program akan diteruskan ke pilihan-pilihan berikutnya. Sehingga setiap pilihan akan di cek dan dijalankan jika syarat nilai konstanta terpenuhi, termasuk statemen default juga akan dijalankan jika semua cabang diatasnya tidak memiliki pernyataan break.

Perlu untuk diketahui, dalam bahasa C standar kita di izinkan untuk menuliskan 257 buah statemen case dalam sebuah struktur switch-case, sedangkan dalam C++ mengizinkan 16384 buah statemen case dalam sebuah struktur switch-case. Namun dalam prakteknya sebaiknya kita membatasi pemilihan tersebut untuk efisiensi program yang kita buat.

- **Perulangan**

Perulangan adalah suatu proses eksekusi statemen-statemen dalam sebuah program secara terus-menerus sampai terdapat kondisi untuk menghentikannya. Operasi perulangan / looping selalu dijumpai didalam berbagai bahasa pemrograman, hal tersebut karena struktur perulangan akan sangat membantu dalam efisiensi program.

Struktur Perulangan For

Struktur perulangan / pengulangan jenis for biasanya digunakan untuk melakukan perulangan yang telah diketahui banyaknya. Biasanya jenis perulangan for dianggap sebagai jenis perulangan yang paling mudah dipahami.

Untuk melakukan perulangan dengan menggunakan struktur perulangan for, kita harus memiliki sebuah variabel sebagai indeksnya. Namun perlu sekali untuk diperhatikan bahwa tipe data dari variabel yang akan digunakan sebagai indeks haruslah tipe data yang mempunyai urutan yang teratur, misalnya tipe data int (0,1,2, …) atau char (‘a’ , ‘b’ , ‘c’ , …).

Adapun **bentuk umum dari struktur perulangan for** adalah seperti yang tampak dibawah ini:

```
// Untuk perulangan yang sifatnya menaik (increment)
// Pastikan nialai awal < kondisi saat berjalan
for(variabel = niali_awal ; kondisi_saat_berjalan ; variable++)
{
Statemen_yang_akan_diulang;
}
// Untuk perulangan yang sifatnya menurun (decrement)
// Pastikan nialai awal > kondisi saat berjalan
for(variabel = nilai_awal ; kondisi_saat_berjalan ; variable–)
{
Statemen_yang_akan_diulang;
}
```

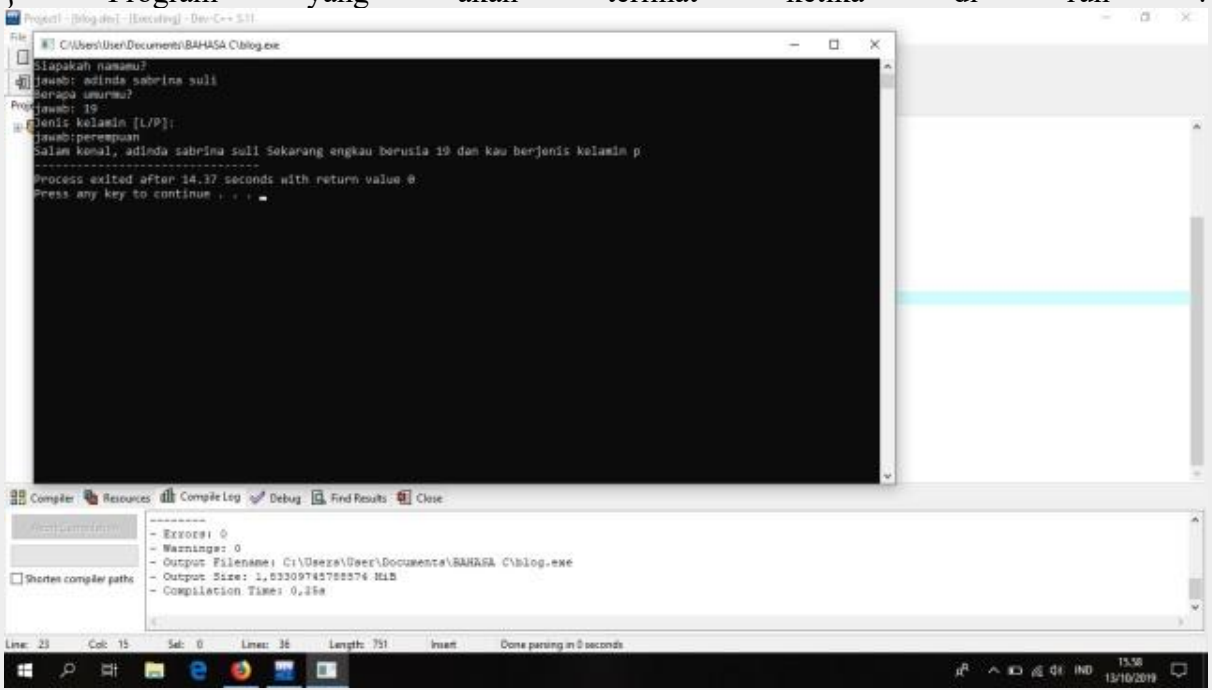
Sebagai catatan bahwa jika kita melakukan perulangan yang sifatnya menaik (increment) maka nilai awal dari variabel yang kita definisikan haruslah lebih kecil dari nilai akhir yang dituliskan dalam kondisi (kondisi saat berjalan). Sebaliknya jika kita akan melakukan perulangan yang sifatnya menurun (decrement) maka nilai awal harus lebih besar dari nilai akhir.

B. Contoh Program

Contoh program untuk variabel

```
#include <iostream>

using namespace std;

int main(){

  // deklarasi tipe data variabel
  string nama;
  int umur;
  char jenis_kelamin;

  // --- proses input ---
  cout << "Siapakah namamu?" << endl;
  cout << "jawab: ";
  // menyimpan data ke variabel
  getline(cin,nama);

  cout << "Berapa umurmu?" << endl;
  cout << "jawab: ";
  // menyimpan data ke variabel
  cin >> umur;

  cout << "Jenis kelamin [L/P]: " << endl;
  // menyimpan data ke variabel
  cin >> jenis_kelamin;

  // --- proses output ---
  cout << "Salam kenal, " << nama << " Sekarang engkau berusia ";
  cout << umur << " dan kau berjenis kelamin "<< jenis_kelamin;

  return 0;
```

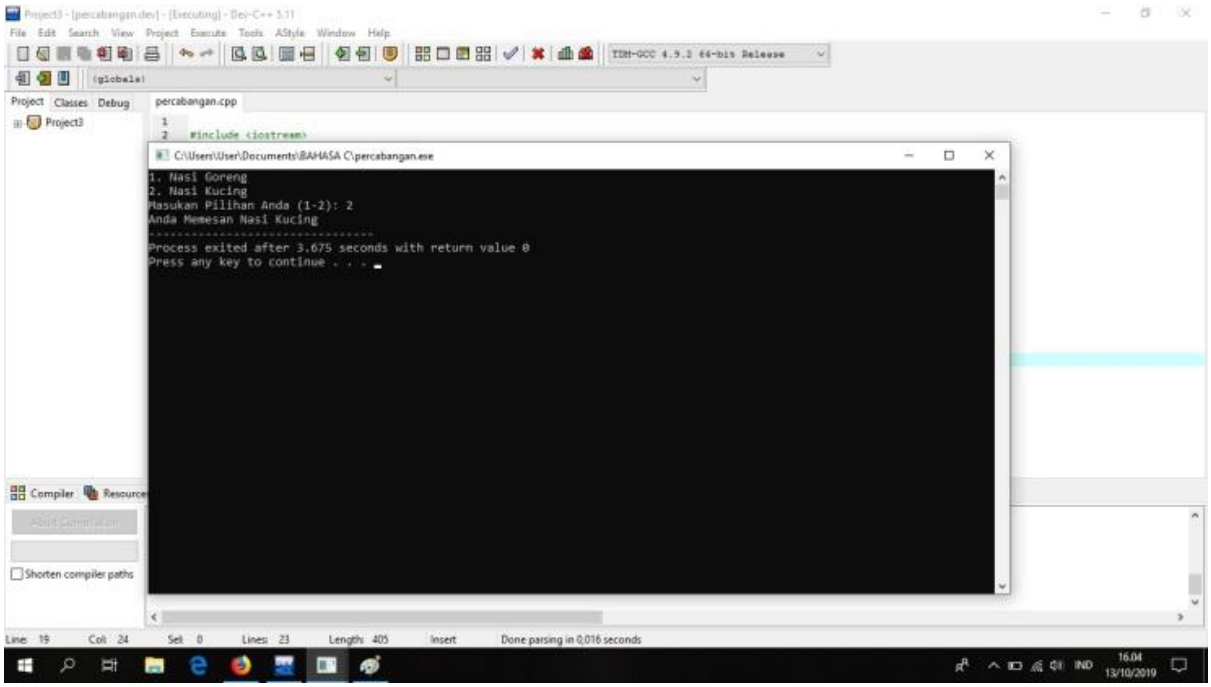
} Program yang akan terlihat ketika di run :



Contoh program untuk percabangan

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

void pilihan (int x){
   if (x==1) {
      cout << "Anda Memesan Nasi Goreng";
   }
   else {
      cout << "Anda Memesan Nasi Kucing";
   }
}

int main(){
   int pilih;

   cout << "1. Nasi Goreng" << '\n';
   cout << "2. Nasi Kucing" << '\n';
   cout << "Masukan Pilihan Anda (1-2): "; cin >> pilih;

   pilihan (pilih);
}
```

program diatas akan terlihat :

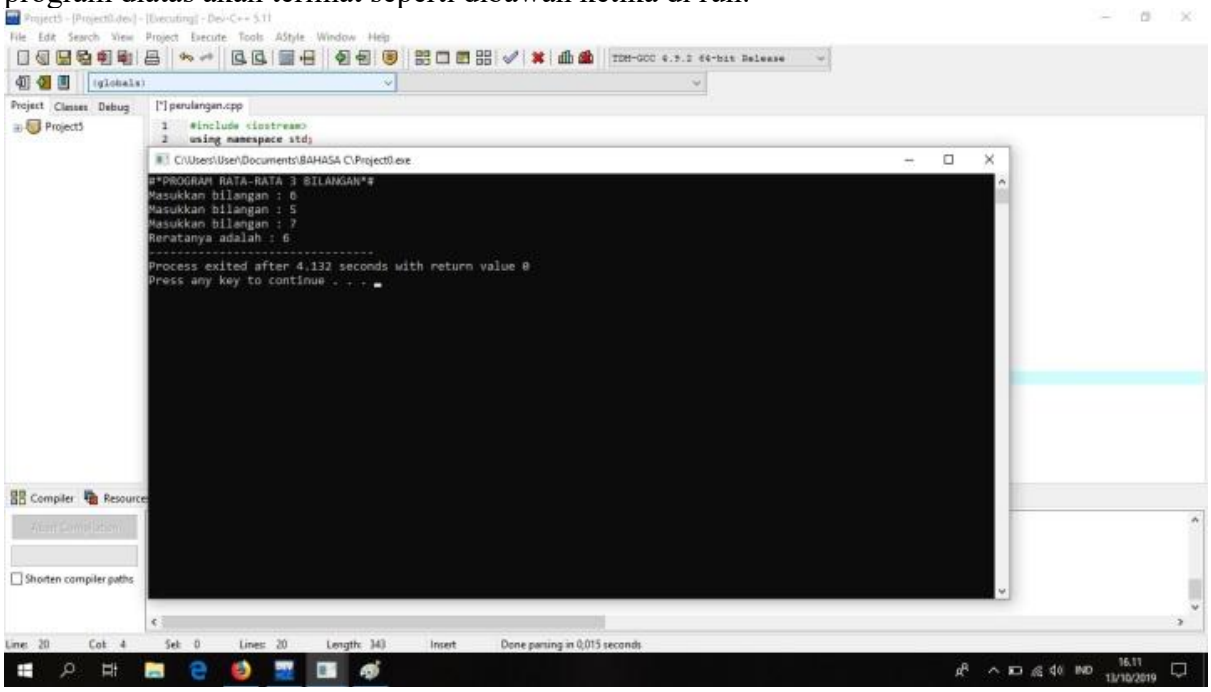


Contoh program untuk perulangan

Program perulangan menghitung rata-rata

```
#include <iostream>
using namespace std;
int main() {
float jumlah, x, rerata;
int bilangan;
jumlah=0;
bilangan=3;
cout<<“#*PROGRAM RATA-RATA 3 BILANGAN*# \n”;
do {
cout<<“Masukkan bilangan : “;
cin>>x;
jumlah = jumlah + x;
bilangan–;
} while (bilangan>0);
rerata = jumlah / 3;
cout<<“Reratanya adalah : “<<rerata;
}
```

program diatas akan terlihat seperti dibawah ketika di run:



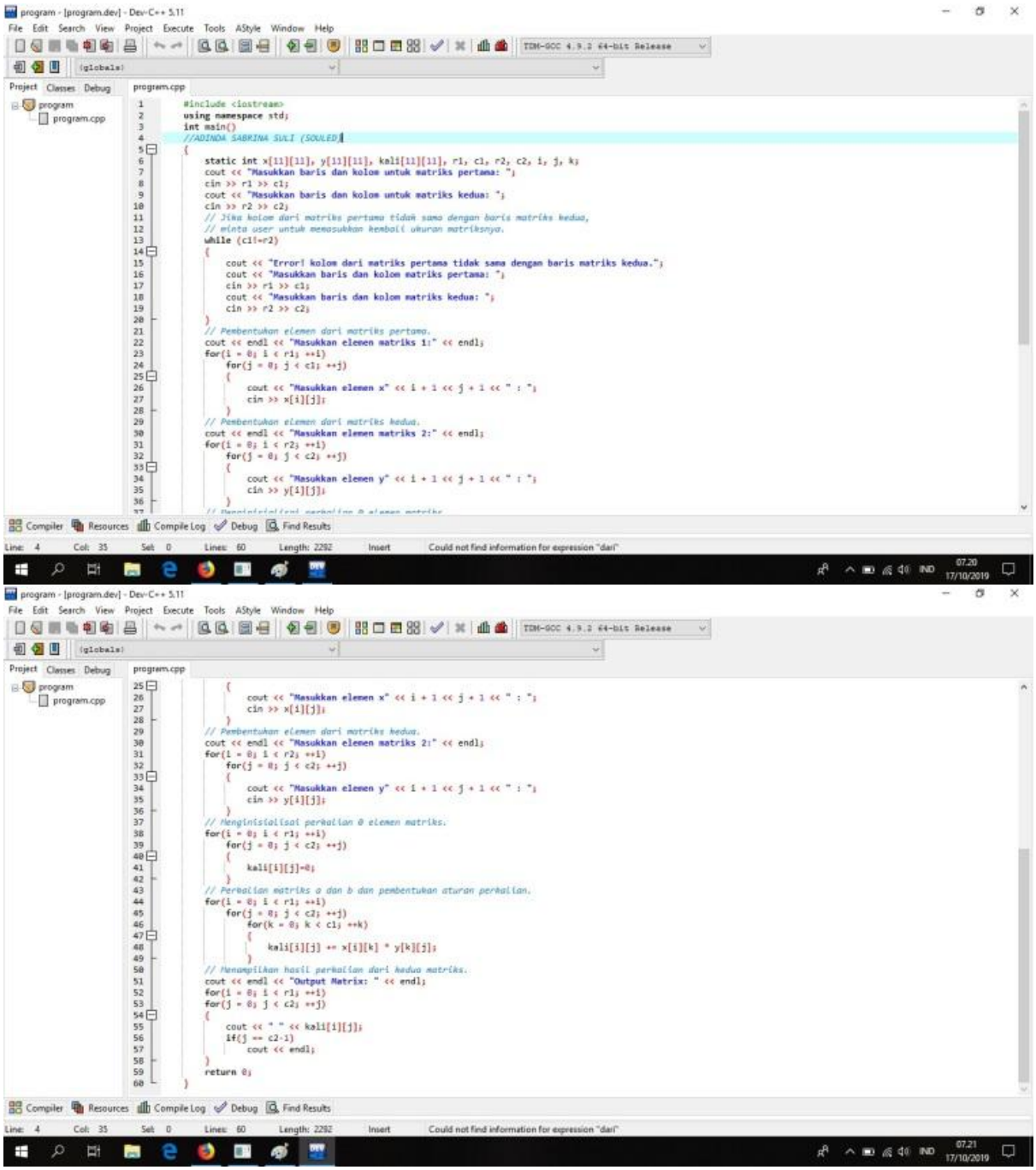
contoh program berdasarkan kasus

Buatlah program untuk mengalikan dua buah matriks!

```
#include <iostream>
using namespace std;
int main()
//ADINDA SABRINA SULI (SOULED)
{
static int x[11][11], y[11][11], kali[11][11], r1, c1, r2, c2, i, j, k;
cout << “Masukkan baris dan kolom untuk matriks pertama: “;
cin >> r1 >> c1;
cout << “Masukkan baris dan kolom untuk matriks kedua: “;
cin >> r2 >> c2;
// Jika kolom dari matriks pertama tidak sama dengan baris matriks kedua,
// minta user untuk memasukkan kembali ukuran matriksnya.
while (c1!=r2)
{
cout << “Error! kolom dari matriks pertama tidak sama dengan baris matriks kedua.”;
cout << “Masukkan baris dan kolom matriks pertama: “;
cin >> r1 >> c1;
cout << “Masukkan baris dan kolom matriks kedua: “;
cin >> r2 >> c2;
}
// Pembentukan elemen dari matriks pertama.
cout << endl << “Masukkan elemen matriks 1:” << endl;
for(i = 0; i < r1; ++i)
for(j = 0; j < c1; ++j)
{
cout << “Masukkan elemen x” << i + 1 << j + 1 << ” : “;
cin >> x[i][j];
}
// Pembentukan elemen dari matriks kedua.
```

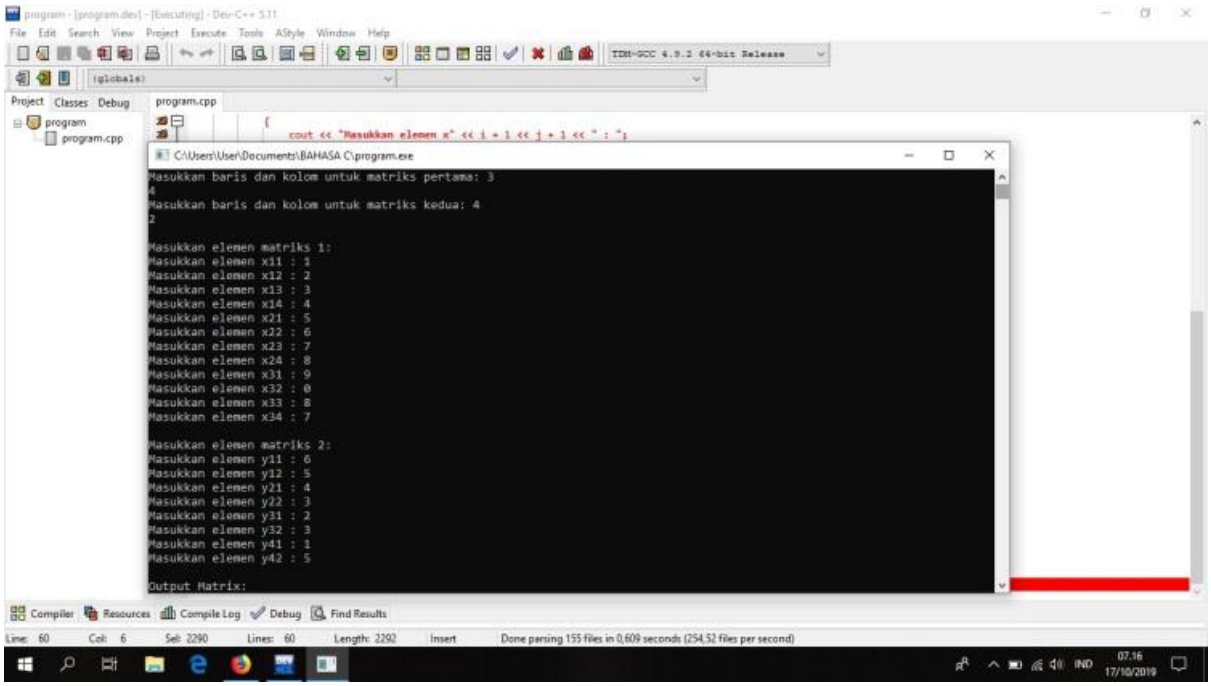
```
cout << endl << “Masukkan elemen matriks 2:” << endl;
for(i = 0; i < r2; ++i)
for(j = 0; j < c2; ++j)
{
cout << “Masukkan elemen y” << i + 1 << j + 1 << ” : “;
cin >> y[i][j];
}
// Menginisialisai perkalian 0 elemen matriks.
for(i = 0; i < r1; ++i)
for(j = 0; j < c2; ++j)
{
kali[i][j]=0;
}
// Perkalian matriks a dan b dan pembentukan aturan perkalian.
for(i = 0; i < r1; ++i)
for(j = 0; j < c2; ++j)
for(k = 0; k < c1; ++k)
{
kali[i][j] += x[i][k] * y[k][j];
}
// Menampilkan hasil perkalian dari kedua matriks.
cout << endl << “Output Matrix: ” << endl;
for(i = 0; i < r1; ++i)
for(j = 0; j < c2; ++j)
{
cout << ” ” << kali[i][j];
if(j == c2-1)
cout << endl;
}
return 0;
}
```

Berikut ini adalah tampilan pemrograman dengan dev c++ dari program diatas



```
#include <iostream>
using namespace std;
int main()
//ADINDA SABRINA SULI (SOULED)
{
    static int x[11][11], y[11][11], kali[11][11], r1, c1, r2, c2, i, j, k;
    cout << "Masukkan baris dan kolom untuk matriks pertama: ";
    cin >> r1 >> c1;
    cout << "Masukkan baris dan kolom untuk matriks kedua: ";
    cin >> r2 >> c2;
    // Jika kolom dari matriks pertama tidak sama dengan baris matriks kedua,
    // minta user untuk memasukkan kembali ukuran matriksnya.
    while (c1!=r2)
    {
        cout << "Error! kolom dari matriks pertama tidak sama dengan baris matriks kedua.";
        cout << "Masukkan baris dan kolom matriks pertama: ";
        cin >> r1 >> c1;
        cout << "Masukkan baris dan kolom matriks kedua: ";
        cin >> r2 >> c2;
    }
    // Pembentukan elemen dari matriks pertama.
    cout << endl << "Masukkan elemen matriks 1:" << endl;
    for(i = 0; i < r1; ++i)
        for(j = 0; j < c1; ++j)
        {
            cout << "Masukkan elemen x" << i + 1 << j + 1 << " : ";
            cin >> x[i][j];
        }
    // Pembentukan elemen dari matriks kedua.
    cout << endl << "Masukkan elemen matriks 2:" << endl;
    for(i = 0; i < r2; ++i)
        for(j = 0; j < c2; ++j)
        {
            cout << "Masukkan elemen y" << i + 1 << j + 1 << " : ";
            cin >> y[i][j];
        }
    // Menginisialisasi perkalian 0 elemen matriks.
    for(i = 0; i < r1; ++i)
        for(j = 0; j < c2; ++j)
        {
            kali[i][j]=0;
        }
    // Perkalian matriks a dan b dan pembentukan aturan perkalian.
    for(i = 0; i < r1; ++i)
        for(j = 0; j < c2; ++j)
            for(k = 0; k < c1; ++k)
            {
                kali[i][j] += x[i][k] * y[k][j];
            }
    // Menampilkan hasil perkalian dari kedua matriks.
    cout << endl << "Output Matrix: " << endl;
    for(i = 0; i < r1; ++i)
    for(j = 0; j < c2; ++j)
    {
        cout << " " << kali[i][j];
        if(j == c2-1)
            cout << endl;
    }
    return 0;
}
```

berikut ini adalah tampilan program yang dihasilkan :


program - [program.dev] - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
program
program.cpp
program.cpp
cout << "Masukkan elemen x" << i + 1 << j + 1 << " : ";
C:\Users\User\Documents\BAHASA C\program.exe
Masukkan baris dan kolom untuk matriks pertama: 3
4
Masukkan baris dan kolom untuk matriks kedua: 4
2
Masukkan elemen matriks 1:
Masukkan elemen x11 : 1
Masukkan elemen x12 : 2
Masukkan elemen x13 : 3
Masukkan elemen x14 : 4
Masukkan elemen x21 : 5
Masukkan elemen x22 : 6
Masukkan elemen x23 : 7
Masukkan elemen x24 : 8
Masukkan elemen x31 : 9
Masukkan elemen x32 : 0
Masukkan elemen x33 : 8
Masukkan elemen x34 : 7
Masukkan elemen matriks 2:
Masukkan elemen y11 : 6
Masukkan elemen y12 : 5
Masukkan elemen y21 : 4
Masukkan elemen y22 : 3
Masukkan elemen y31 : 2
Masukkan elemen y32 : 3
Masukkan elemen y41 : 1
Masukkan elemen y42 : 5
Output Matrix:
Compiler Resources Compile Log Debug Find Results
Line: 60 Col: 6 Sel: 2290 Lines: 60 Length: 2292 Insert Done parsing 155 files in 0,609 seconds (254,52 files per second)
IND 07.16 17/10/2019