MODUL ALGORITMA DAN PEMROGRAMAN DASAR

Oleh :

Sevi Nurafni

Catatan

1. Modul ini dirancang untuk dapat menjadi pegangan mata kulaih algoritma dan pemrograman dasar
2. Anda dapat membuka modul ini saat latitan praktikum
3. Anda sangat disarankan untuk mencoba menjalankan semua program modul di komputer anda, supaya anda dapat mengtahui keluaran dari program yang ada
4. Anda sangat disarankan untuk berekperimen dari program-program yang ada di modul ini supaya anda mendapat gambaran lebih jelas menganai apa yang program anda lakukan
5. Anda sangat disarankan membaca tutoial dari tempat lain dan mengeksplor sendiri bahasa yang anda gunakan

MODUL 1

1. 1 Pendahuluan

Pada modul ini, kita menggunakan C++ versi 23 dan menggunakan text editor visual studio code

Beberapa karakteristik dari bahasa ini :

- C++ ini bersifat *case sensitive* yang artinya perbedaan huruf besar dan hurus kecil menyebabkan perbedaan makna
- C++ adalah bahasa yang menggunakan static typing, yang artinya semua variabel harus memiliki tipe data yang dideklerasikan saat kompilasi seperti (int, float, string, doubel, boolean, dll)

1. 2 Input dan output

Dalam program c++ untuk menulis “Hello World!” ke layar adalah sebagai berikut :

Contoh 1

```cpp
#include <iostream>

using namespace std ;

int main() {

    cout << "Hello World!" << endl;

    return 0;

}
```

Contoh 2

```cpp
#include <iostream>

using namespace std;

int main() {
    cout << " Ini
program";                                   //outputnya: Ini
program
    cout << " pertama saya" << endl;              // outputnya:
pertama saya
    cout << " Dengan" << " Koma" << endl;  // outputnya: Dengan Koma
```

```
    cout << string("Dengan") + string("Plus") << endl; //outputnya:
DenganPlus
    //Perhatikan tanda ‘+’ pada cout melakukan kontatinasi pada
string saja

    string variabel = "sebuah nilai"; cout << "Dengan format " <<
variabel << endl;
    // outputnya: Dengan format sebuah nilai

   // semua yang diberi tanda (//) akan diabaikan oleh compiler
    return 0;
}
```

Untuk melakukan input, kita membutuhkan penampung untuk menyimpan data yang diinputkan. Sebagai contoh, di bawah ini adalah program yang menerima input dan menuliskan ulang yang dimasukkan

Contoh 3

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

using namespace std;

int main() {
    string S;
    int N;

    // Meminta pengguna memasukkan kalimat
    cout << " Masukkan kalimat: ";
    getline(cin, S);  // Menggunakan getline untuk input string
dengan spasi

    // Menampilkan kalimat yang dimasukkan pengguna
    cout << " Anda memasukkan kalimat " << S << endl;

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

    // Menampilkan hasil penjumlahan angka dengan 5
    cout << " Jika angka Anda ditambah 5, hasilnya " << N + 5 <<
endl;

    return 0;
}
```

1

2

3

4

Penjelasan

Pada bagian nomor (1) dan (3), program membaca input dari pengguna dan menyimpannya ke dalam variabel S dan N. Untuk menerima input berupa bilangan (sehingga dapat dijumlahkan, dikalikan, atau dioperasikan lainnya), kita menggunakan std::cin yang secara otomatis menyimpan nilai ke variabel dengan tipe data yang sesuai. Dalam hal ini, untuk menerima bilangan bulat, kita gunakan tipe int untuk variabel N. Selain int untuk bilangan bulat, C++ juga mendukung tipe data float atau double untuk bilangan real, serta banyak tipe data lain yang bisa digunakan sesuai kebutuhan.

1. 3 Variabel

Variabel adalah penanda identitas yang digunakan untuk menampung suatu nilai. Nilai yang ditampung oleh variabel dapat diubah sepanjang kode program. Secara teknis, variabel merujuk kepada suatu alamat di memory komputer

Dalam matematika, konsep variabel biasanya menggunakan karakter x atau y.

A. Cara Pendeklarasian Variabel

Deklarasi adalah proses untuk memberitahu compiler C++ bahwa kita ingin membuat sebuah variabel. Bahasa C++ termasuk bahasa pemrograman yang menggunakan konsep strongly typed programming language, yang artinya untuk setiap variabel harus ditulis akan berisi tipe data apa. Apakah itu angka bulat (integer), angka pecahan (float / double), huruf (char), atau yang lain.

Deklarasi variabel ditulis dengan format sebagai berikut:

tipe_data nama_variabel;

keterangan :

- tipe_data = untuk merepresentasikan tipe data
- nama_variabel = untuk merepresentasikan nama variabel

Berikut contoh mendeklarasikan variabel di C++:

```
// operasi aritmatika
#include <iostream>

using namespace std;

int main() {
    // Deklarasi variabel integer
    int x, y, hasil;


// Input nilai x dan y
```

```
    cout << "Masukkan nilai x: ";
    cin >> x;
    cout << "Masukkan nilai y: ";
    cin >> y;

    // Operasi aritmatika
    hasil = x + y;

    // Output hasil
    cout << "Hasil penjumlahan " << x << " + " << y << " = " << hasil
<< endl;

    return 0;
}
```

1. 4 Tipe Data

Ada beberapa macam tipe data, namun dalam pengenalan komputasi kita hanya akan banyak menggunakan tipe data berikut:

Tipe data	Penulisan syntax dalam c++	Deskripsi
Boolean	bool	True atau false
Integer	int	Untuk seluruh angka yang mempresentasikan bilangan bulat
Float	float	Untuk seluruh angka yang mempresentasikan bilangan real
String	string	Kumpulan dari beberapa karakter seperti (huruf, simbol)

Contoh penggunaanya

```
#include <iostream>
#include <string> // Library untuk tipe data string

using namespace std;

int main() {
    // Deklarasi variabel
    bool isActive = true;         // Tipe bool (true/false)
    int age = 25;                 // Tipe int (bilangan bulat)
```

```
    float height = 1.75;          // Tipe float (bilangan real / desimal)
    string name = "Hamzah";  // Tipe string (kumpulan karakter)

    // Menampilkan nilai variabel
    cout << "Apakah aktif? " << boolalpha << isActive << endl; // boolalpha untuk menampilkan true/false
    cout << "Umur: " << age << " tahun" << endl;
    cout << "Tinggi badan: " << height << " meter" << endl;
    cout << "Nama: " << name << endl;

    // Menggunakan variabel untuk perhitungan dan logika
    if (isActive) {
        cout << name << " berusia " << age << " tahun dan memiliki tinggi " << height << " meter." << endl;
    }

    return 0;
}
```

Dalam C++, Anda tidak dapat menggunakan petik satu (') untuk menyimpan string. Petik satu digunakan untuk karakter tunggal (tipe char), sedangkan petik dua (") digunakan untuk string (kumpulan karakter, tipe string)

1. 5 Operator

A. Operator aritmatika

Operator aritmatika dalam C++ adalah simbol yang digunakan untuk melakukan operasi aritmatika umum, seperti penjumlahan, pengurangan, perkalian, pembagian, dan modulus

operator	Deskripsi	Contoh
+	Penjumlahan	2+3 bernilai 5
-	Pengurangan	1-8 bernilai -7
*	Perkalian	5 * 6 bernilai 30
/	Pembagian	13/5 bernilai 2.6
//	Pembagian (dibulatkan ke bawah)	13//5 bernilai 2
%	Sisa bagi (modulo)	13%5 bernilai 3

Berikut contoh penggunaan operasi aritmatika :

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

int main() {
int a, b, c, d, e;

a = 2 + 3;
b = 1 - 8;
c = 5 * 6;
d = 13 / 5;
e = 13 // 5
f = 13 % 5;

cout << a << endl; // 5
cout << b << endl; // -7
cout << c << endl; // 30
cout << d << endl; // 2.6
cout << e << endl; // 2
cout << f << endl; // 3

return 0;
}
```

B. Operator Assignment

Operator penugasan adalah operator untuk memasukkan suatu nilai ke dalam variabel. Operator ini sebenarnya sudah sering kita pakai sejak awal buku. Dalam bahasa C++, operator assignment menggunakan tanda sama dengan (=). Dalam bahasa inggris, operator ini disebut sebagai operator assignment.

Operator	Deskripsi	Contoh
=	Assginment	N=5
+=	Penjumlahan	N += 5, N akan di tambah 5
|-=	Pengurangan	N -= 5, N akan dikurang 5.
*=	Perkalian	N *= 5, N akan dikali 5.
/=	Pembagian	N /= 5, N akan dibagi 5.

//=	Pembagian (dibulatkan ke bawah	N //= 5, N akan dibagi 5 (dibulatkan ke bawah).
%=	Sisa bagi / modulo	N %= 5, N akan dimodulo 5

Berikut contoh penggunaan operator assignment

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

int main() {
int a = 10, b = 10, c = 10, d = 10, e = 10, f = 10;

cout << "Operator assignment gabungan bahasa C++" << endl;
cout << "=======================================" << endl;
cout << "Variabel a, b, c, d, e, f = 10" << endl;
cout << endl;

a += 5;
b -= 3;
c *= 3;
d /= 3;
e %= 3;
f <<= 2;

cout << "Hasil operasi a += 5: " << a << endl;
cout << "Hasil operasi b -= 3: " << b << endl;
cout << "Hasil operasi c *= 3: " << c << endl;
cout << "Hasil operasi d /= 3: " << d << endl;
cout << "Hasil operasi e %= 3: " << e << endl;
cout << "Hasil operasi f <<= 2: " << f << endl;

return 0;
}
```

Dalam kode program kita membuat 6 operator assignment gabungan. Variabel a, b, c, d, e, dan f semuanya diisi nilai awal 10, kemudian di proses dengan berbagai operator.

C. Operator Relasional

Operator perbandingan dipakai untuk membandingkan 2 buah operand. Hasil dari operator ini berupa nilai boolean true atau false. Operator perbandingan sering dipakai dalam proses pengambilan keputusan atau percabangan kode program dengan struktur if else

Operator	Deskripsi	Contoh true	Contoh false
= =	Sama dengan	2 = = 2	2 == 3
!=	Tidak sama dengan	3 != 2	3 != 3
<	Kurang dari	2 < 3	2 < 2
>	Lebih dari	3 > 2	2 > 3
<=	Kurang dari sama dengan	2 <= 2	3 <= 1
>=	Lebih dari sama dengan	6 >= 5	2 >= 4

Berikut contoh penggunaan operator relasional

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

int main() {
bool a;

 a = 2 == 2;
 cout << std::boolalpha << a << endl; // true

 a = 3 != 2;
 cout << std::boolalpha << a << endl; // false

 a = 2 < 3;
 cout << std::boolalpha << a << endl; // true

 a = 3 > 2;
 cout << std::boolalpha << a << endl; // false

 a = 3 <= 1;
```

```
        cout << std::boolalpha << a << endl; // true

        a = 2 >= 4;
        cout << std::boolalpha << a << endl; // true

        return 0;
}
```

D. Operator logika

Operator logika adalah operator yang dipakai untuk membuat kesimpulan logis dari 2 kondisi boolean: true atau false. Nilai akhir dari operator logika juga bertipe boolean, sedangkan nilai awalnya bisa berbentuk boolean, atau bisa juga berupa operasi yang menghasilkan nilai boolean seperti operasi perbandingan.

Sifat-sifar operator perbandingan :

- Operator logika AND hanya menghasilkan nilai true jika kedua operand juga bernilai true
- Operator logika OR menghasilkan nilai true jika salah satu operand ada yang bernilai true. Hasil akhir hanya bisa false jika kedua operand bernilai false.
- Dan operator ! dipakai untuk mengubah nilai operand saat ini. Jika true akan menjadi false, dan jika false akan menjadi true:

Operator	Deskripsi	Contoh true	Contoh false
and	Dan: True jika kedua operand True	(1 < 2)and (3 == 3)	(1 == 2)and (3 == 3)
or	Atau: True jika salah satu operand True	(1 < 2)or (4 == 3)	(3 < 2)or (2 == 3)
not	Atau: True jika salah satu operand True	not (3 < 2)	not (1 < 2)

Berikut contoh operator logika :AND,OR

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

int main() {
```

```
    bool a;

    //operator and &&
    a = true && true;
    cout << std::boolalpha << a << endl; // true

    a = true && false;
    cout << std::boolalpha << a << endl; // false

    //operator || or
    a = false || true;
    cout << std::boolalpha << a << endl; // true

    a = false || false;
    cout << std::boolalpha << a << endl; // false

    cout << endl;
    return 0;
}
```

Berikut contoh operator logika :Not

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

int main() {
bool a, b;

a = !true;
cout << std::boolalpha << a << endl; // false

b = !(1 > 2);
cout << std::boolalpha << !a << endl; // true

cout << endl;
```

```
        return 0;
}
```

MODUL 2

1. 6 Percabangan

Dalam pemrograman, terdapat percabangan. Dengan demikian, program kita dapat berprilaku tergantung input user. Misal kita buat program yang memeriksa apakah sebuah bilangan positif:

```cpp
#include <iostream>
#include <string>  // Untuk menggunakan fungsi to_string

using namespace std;

int main() {
    int N;

    cout << "Masukkan nilai N: ";
    cin >> N;

    if (N > 0) {
        cout << to_string(N) + " bilangan positif" << endl;
    }

    return 0;
}
```

Laku, jika kita ingin menuliskan kebalikannya:

```cpp
#include <iostream>
#include <string>  // Untuk menggunakan fungsi to_string

using namespace std;

int main() {
    int N;

    cout << "Masukkan nilai N: ";
    cin >> N;

    if (N > 0) {
        cout << to_string(N) + " bilangan positif" << endl;
    } else {
        cout << to_string(N) + " bilangan bukan positif" << endl;
    }
```

```
    return 0;
}
```

Namun, kita tahu kadang bilangan bisa nol atau negatif, jadi perlu kita tambahkan

```
#include <iostream>
#include <string>  // Untuk menggunakan fungsi to_string

using namespace std;

int main() {
    int N;

    cout << "Masukkan nilai N: ";
    cin >> N;

    if (N > 0) {
        cout << to_string(N) + " bilangan positif" << endl;
    } else if (N < 0) {
        cout << to_string(N) + " bilangan negatif" << endl;
    } else {
        cout << to_string(N) + " bilangan nol" << endl;
    }

    return 0;
}
```

Kita bisa modifiksi program di atas dengan switch

```
#include <iostream>
#include <string>  // Untuk menggunakan fungsi to_string

using namespace std;

int main() {
    int N;

    cout << "Masukkan nilai N: ";
    cin >> N;

    // Normalisasi nilai N untuk switch
    int kategori = 0;  // Default ke nol
    if (N > 0) {
        kategori = 1;  // Positif
    } else if (N < 0) {
```

```
        kategori = -1;  // Negatif
    }

    switch (kategori) {
        case 1:
            cout << to_string(N) + " bilangan positif" << endl;
            break;
        case -1:
            cout << to_string(N) + " bilangan negatif" << endl;
            break;
        case 0:
            cout << to_string(N) + " bilangan nol" << endl;
            break;
        default:
            cout << "Input tidak valid." << endl;
            break;
    }

    return 0;
}
```

Perhatikan juga kalau kita bisa membuat else ini berulang sampai yang kita mau, selain itu, kita juga bisa meletakkan if dalam if

```
#include <iostream>
#include <string>  // Untuk menggunakan fungsi to_string

using namespace std;

int main() {
    int N;

    cout << "Masukkan nilai N: ";
    cin >> N;

    if (N >= 0) {  // Cek apakah N positif atau nol
        if (N > 0) {
            cout << to_string(N) + " bilangan positif" << endl;
        } else {  // Jika N == 0
            cout << to_string(N) + " bilangan nol" << endl;
        }
    } else {  // Jika N < 0
        cout << to_string(N) + " bilangan negatif" << endl;
    }

    return 0;
```

```
}
```

1. 7 Latihan Modul 1

A. Kesalahan penulisan syntax

Cari dan perbaiki kesalahan syntax yang mencegah program berjalan dengan benar

a.

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

int main() {
    cout << "Masukkan angka pertama: ";
    int angka1;
    cin >> angka1

    cout << "Masukkan angka kedua: ";
    int angka2;
    cin >> angka2;

    int hasil = angka1 + angka2;
    cout << "Hasil penjumlahan: " << hasil << endl;

    return 0;
```

b.

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

int main() {
    int nilai;
    cout << "Masukkan nilai Anda: ";
    cin >> nilai;

    if (nilai >= 60);
        cout << "Selamat! Anda lulus." << endl;
    else
        cout << "Maaf, Anda belum lulus." << endl;

    return 0;
}
```

B. Program Ganjil-Genap

Buatlah sebuah program untuk menampilkan hasil operasi dasar logika AND, OR, NOT.

Contoh 1 :

```
Masukan nilai a: true
Masukan nilai b: false
a AND b = false
```

Contoh 2 :

```
Masukan nilai a: true
Masukan nilai b: false
a OR b = true
```

Contoh 3 :

```
Masukan nilai a: true
Masukan nilai b: false
not a = false
not b = true
```

C. Kalkulator Sederhana

Buatlah sebuah program kalkulator sederhana yang menerima 2 buah angka dan sebuah karakter operasi, dan menuliskan hasil perhitungannya. Operator yang diterima adalah + (tambah), - (kurang), * (kali), / (bagi, dibulatkan ke bawah), % (sisa bagi).

Contoh 1 :

```
    Masukkan angka
pertama: 2
Masukkan angka
kedua: 6 Masukkan
operator: +
   2 + 6 = 8
```

Contoh 2 :

```
    Masukkan angka
pertama: 13
Masukkan angka
kedua: 5 Masukkan
operator: %
    13 % 5 = 3
```