TELKOM SCHOOLS

BIDANG KEAHLIAN: TEKNOLOGI INFORMASI & KOMUNIKASI
PROGRAM KEAHLIAN: TEKNIK KOMPUTER & INFORMATIKA
PAKET KEAHLIAN: REKAYASA PERANGKAT LUNAK
MATA PELAJARAN:PEMROGRAMAN DASAR X
SEMESTER: GENAP

Nomor Dokumen : RPL C1-11-2015
Versi : 2.0
Tanggal : 28 Februari 2017

Diterbitkan oleh:
TELKOM SCHOOLS

Directorate of Primary & Secondary Education (PSE)
Telkom Education Foundation
Gedung Sate -- West Wing @ 2nd Floor -- Room #7
Jl. Cisanggarung No.2 Bandung 40115, West Java Indonesia
t: ++62 22 7272 078, 22 7106 043 ext 134, f: ++62 22 7102 4444

TELKOM SCHOOLS

BIDANG KEAHLIAN: TEKNOLOGI INFORMASI & KOMUNIKASI
PROGRAM KEAHLIAN: TEKNIK KOMPUTER & INFORMATIKA
PAKET KEAHLIAN: REKAYASA PERANGKAT LUNAK
MATA PELAJARAN:PEMROGRAMAN DASAR X
SEMESTER: GENAP

Nomor Dokumen : RPL C1-11-2015
Versi : 2.0
Tanggal : 28 Februari 2017

Dengan diberlakukannya Buku Ajar ini, maka Buku Ajar versi sebelumnya dinyatakan tidak berlaku lagi. **(*)**

Ditetapkan di : Bandung
Pada tanggal : 31 Desember 2015

Director of PSE
Telkom Education Foundation

_____
NIK.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

KEGIATAN BELAJAR 5

TIPE DATA, VARIABEL, OPERATOR DAN EKSPRESI

A. Tujuan Pembelajaran

Setelah melakukan kegiatan pembelajaran siswa dapat :
1. Menjelaskan variabel tunggal dan array
2. Menjelaskan tipe data yang digunakan pada pemrograman komputer
3. Menjelaskan operator yang digunakan pada pemrograman komputer
4. Mendeklarasikan variabel tunggal dan array
5. Menentukan tipe data yang digunakan pada pemrograman komputer
6. Menggunakan operator yang digunakan pada pemrograman komputer

B. Deskripsi Materi

Sebelum mempelajari materi pembelajaran ini, bacalah informasi di bawah ini dan jawab pertanyaan di bawahnya.

Perhatikan potongan kode berikut.

```
void 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;

    system("pause");
}
```

```
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 : 2
Press any key to continue . . .
```

Dari kode di atas, coba kalian cari informasi mengenai apa yang dimaksud dengan char, int, short int, long int, float, dan double.

1. Tipe Data

Dalam pemrograman C++, tipe data merupakan penentu jenis nilai yang terdapat dalam sebuah program. Secara umum, terdapat empat jenis tipe data dasar dalam C++, yaitu tipe bilangan bulat, bilangan riil, logika, dan karakter.

1) Tipe bilangan bulat

Jenis bilangan yang tidak mengandung angka lebih dari 0 di belakang koma. Berikut ini adalah macam-macam tipe data bilangan bulat.

Tabel 1.1 Tipe data bilangan bulat

Tipe	Ukuran	Range
int	2 atau 4 byte	Signed: -32.768 to 32.767 atau -2.147.483,648 s.d. 2.147.483.647 Unsigned: 0 s.d. 4.294.967.295
short int	2 byte	Signed: -32.768 s.d. 32.767 Unsigned: 0 to 65.535
long int	4 byte	Signed: -2.147.483.648 s.d. 2.147.483.648 Unsigned: 0 s.d. 4.294.967.295

2) Tipe bilangan riil

Jenis bilangan yang mengandung angka di belakang koma. Berikut ini adalah macam-macam tipe data bilangan riil.

Tabel 1.2 Tipe data bilangan riil

Tipe	Ukuran	Range
float	4 byte	1,2E s.d. 3,4E+38
double	8 byte	2,3E-308 s.d. 1,7E+308
long double	10 byte	3,4E-4932 s.d. 1,1E+4932

3) Tipe logika

Jenis bilangan yang hanya mengandung dua buah nilai yaitu 1 untuk nilai benar (true) dan 0 untuk nilai salah (false). Keyword yang digunakan adalah bool dan memiliki ukuran 1 byte.

4) Tipe karakter

Tipe data karakter terdiri atas bermacam-macam karakter, yaitu angka, huruf, atau tanda baca tunggal. Dalam C++, tipe data karakter dideklarasikan dengan keyword char, di mana char hanya dapat menampung 1 karakter saja.

Tipe data karakter yang dapat menampung lebih dari 1 karakter disebut dengan tipe data string. String merupakan array dari tipe data char.

Untuk lebih jelasnya perhatikan listing kode berikut.

```
#include "stdafx.h"
#include "iostream"
using namespace std;


void main()
{
    char huruf = 'a';
    char kalimat[] ="Belajar C++ ";

    cout << huruf << endl;
    cout << kalimat << endl;

    system("pause");
}
```

Gambar 5.1 Tipe Data Karakter

Jika kode program tersebut dieksekusi, maka hasil yang ditampilkan adalah:

```
a
Belajar C++
Press any key to continue . . .
```

Gambar 5.2 Hasil Eksekusi Tipe Data Karakter

5) Tipe array

Array adalah sekelompok data yang bertipe sama dan menempati satu lokasi memori secara berurutan. Berikut ini adalah sintaks untuk mendeklarasikan tipe data array.

tipedata namaVariabel[jumlah_elemen];

Contoh: `char huruf[5];`

Maksudnya, variabel dengan nama 'huruf' bertipe data array dan dapat menampung sebanyak 5 karakter.
Penjelasan lebih lanjut mengenai variabel akan diuraikan pada subbab berikutnya.

Latihan Soal

1. Sebutkan macam-macam tipe data bilangan bulat!
2. Jelaskan perbedaan tipe data float dan double!
3. Jelaskan yang dimaksud dengan tipe data logika!
4. Jelaskan cara mendeklarasikan tipe data string menggunakan char!
5. Perhatikan potongan kode berikut!

```
#include "stdafx.h"
#include "iostream"


using namespace std;

void main()
{
double p, result;
float param = 6.5;

result = log(param);

cout << "ln(" << param << ") = " << result;
system("pause");
}
```

Sebutkan tipe data yang digunakan pada kode di atas!

2. Variabel

Suatu wadah yang berfungsi menyimpan nilai atau informasi. Variabel dapat menampung nilai yang tetap maupun berubah-ubah. Sebelum dapat digunakan, variabel harus dideklarasikan terlebih dahulu agar dapat dikenali oleh program.

Cara mendeklarasi variabel adalah menambahkan tipe data dan nama variabel yang digunakan. Tipe data ditambahkan untuk menentukan jenis nilai atau informasi yang akan ditampung dalam variabel.

Selain itu, penamaan variabel harus mengikuti aturan-aturan tertentu, di antaranya:

1) Boleh terdiri atas gabungan angka dan huruf di mana karakter pertama harus berupa huruf
2) Bersifat case-sensitive, di mana nama dengan huruf besar dan huruf kecil dianggap berbeda
3) Tidak boleh menggunakan spasi
4) Tidak boleh menggunakan simbol-simbol khusus kecuali underscore (_)
5) Tidak boleh menggunakan nama-nama yang menjadi keyword dalam c++ seperti switch, for, dll.
6) Panjang maksimal nama variabel adalah 31 karakter

Contoh pendeklarasian variabel pada sebuah listing kode C++.

```
int bilangan;
bilangan = 5;
```

Maksudnya, sebuah variabel dengan nama ‘bilangan’ dapat menampung data dengan tipe integer. Baris kode selanjutnya menyatakan bahwa variabel bilangan telah menampung data bertipe integer, yaitu 5. Pendeklarasian variabel tersebut sebenarnya dapat dipersingkat menjadi:

```
int bilangan = 5;
```

Latihan Soal

1. Apa yang dimaksud dengan variabel?
2. Jelaskan cara inisialisasi sebuah variabel dan contohnya!
3. Sebutkan aturan-aturan penamaan sebuah variabel!
4. Suatu variabel diinisialisasi sebagai berikut

```
int 4ngka;
bool switch = true;
float bilangan genap;
```

Pendeklarasian ketiga variabel tersebut menimbulkan pesan error pada compiler. Jelaskan alasannya!

3. Operator

Operator adalah simbol-simbol khusus yang digunakan untuk memanipulasi data. Dalam C++, terdapat komponen lain yang berhubungan erat dengan operator, yaitu operand. Operand adalah angka atau karakter yang akan dioperasikan menggunakan operator. Berdasarkan jumlah operand yang digunakan, jenis operator dikelompokkan menjadi tiga macam, yaitu operator unary, operator binary, dan operator ternary. Operator unary bekerja dengan satu operand, operator binary bekerja dengan dua operator, sedangkan operator ternary bekerja dengan tiga operand.

Berdasarkan fungsinya, terdapat empat macam operator yang umumnya digunakan dalam C++, di antaranya:

1) Operator relasional

Operator yang digunakan untuk menentukan hubungan atau relasi dua buah nilai.

Tabel 1.4 Operator Relasional

Simbol	Definisi	Contoh	Keterangan
==	Sama dengan	X == Y	X=2; Y=2 maka X== Y bernilai benar
>	Lebih besar dari	X > Y	X=5; Y=4 maka X > Y bernilai benar
<	Kurang dari	X < Y	X=4; Y=5 maka X < Y bernilai benar
>=	Lebih dari atau sama	X >= Y	X=4; Y=3 maka X => Y bernilai benar X=3; Y=3 maka X => Y bernilai benar
<=	Sama atau kurang dari	X <= Y	X=6; Y=8 maka X <= Y bernilai benar X=8; Y=8 maka X <= Y bernilai benar
!=	Tidak sama dengan	X != Y	X=1; Y=2 maka X != Y bernilai benar

Untuk lebih memahami operator relasional, perhatikan potongan kode berikut.

```
#include "stdafx.h"
#include "iostream"

using namespace std;

void main()
{
    int a,b;
    cout << "Masukkan nilai a: "; cin >> a;
    cout << "Masukkan nilai b: "; cin >> b;

    if (a == b)
        cout << "Nilai a sama dengan b\n";
    if (a < b)
        cout << "Nilai a lebih kecil dari b\n";
    if (a > b)
        cout << "Nilai a lebih besar dari b\n";
    if (a != b)
        cout << "Nilai a tidak sama dengan b\n";
    if (a <= b)
        cout << "Nilai a lebih kecil atau sama dengan b\n";
    if (a >= b)
        cout << "Nilai a lebih besar atau sama dengan b\n";


system("pause");
}
```

Gambar 5.3 Operator Relasional

Jika dieksekusi maka hasilnya adalah sebagai berikut.

```
Masukkan nilai a: 10
Masukkan nilai b: 20
Nilai a lebih kecil dari b
Nilai a tidak sama dengan b
Nilai a lebih kecil atau sama dengan b
Press any key to continue . . .
```

Gambar 5.4 Hasil Operator Relasional

2) Operator aritmatika

Operator yang digunakan untuk melakukan operasi aritmatika. Berikut ini adalah macam-macam operator aritmatika dalam c++.

Tabel 1.5 Operator Aritmetika

Simbol	Nama	Operasi	Contoh
+	Plus	Penjumlahan	1 + 1 maka hasilnya 2
-	Minus	Pengurangan	2 – 1 maka hasilnya 1
*	Asteriks	Perkalian	3 * 2 maka hasilnya 6
/	Slash	Pembagian	8 / 2 maka hasilnya 4
%	Percent	Sisa hasil bagi	7 % 4 maka hasilnya 3

Perhatikan kode berikut.

```
#include "stdafx.h"
#include "iostream"

using namespace std;

void main()
{
    int a = 20;
    int b = 8;
    int c ;

   c = a + b;
   cout << "\na = " << a << endl;
   cout << "b = " << b << endl << endl;

   c = a - b;
   cout << "a - b = " << c << endl ;

   c = a + b;
   cout << "a + b = " << c << endl ;

   c = a * b;
   cout << "a x b = " << c << endl ;

   c = a / b;
   cout << "a / b = " << c << endl ;

   c = a % b;
   cout << "a % b = " << c << endl ;

system("pause");
}
```

Gambar 5.5 Operator Aritmetika

Jika potongan kode di atas dieksekusi, hasil yang ditampilkan pada program adalah sebagai berikut.

```
a = 20
b = 8

a - b = 12
a + b = 28
a x b = 160
a / b = 2
a % b = 4
Press any key to continue . . .
```

Gambar 5.6 Hasil Operator Aritmetika

3) Operator logika

Operator yang digunakan untuk melakukan operasi yang selalu menghasilkan nilai 1 (benar) atau 0 (salah). Macam-macam operator logika yang umum digunakan dalam C++ ditunjukkan oleh Tabel 1.6.

Tabel 1.6 Operator Logika

Simbol	Operasi	Contoh
&&	AND	A B A&&B 0 0 0 0 1 0 1 0 0 1 1 1
||	OR	A B A||B 0 0 0 0 1 1 1 0 1 1 1 1
!	NOT	A !A 0 1 1 0

4) Operator *increment* dan *decrement*
Operator *increment* (++) digunakan untuk menambah nilai suatu variabel atau operand sebanyak 1. Berikut ini adalah cara penggunaan operator increment pada sebuah operand.

```
x++;
      //sama dengan
x = x + 1;
```

Sebaliknya, operator decrement digunakan untuk mengurangi nilai suatu variabel atau operand sebanyak 1. Berikut ini adalah cara penggunaan operator increment pada sebuah operand.

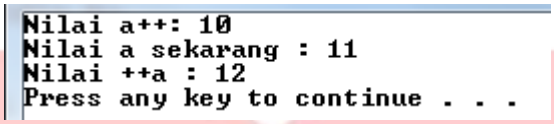
```
x--;
      //sama dengan
x = x - 1;
```

Baik operator increment maupun decrement dapat ditambahkan sesudah (postfix) atau sebelum (prefix) operand yang akan dioperasikan. Namun, hasil yang diperoleh dapat berbeda. Perhatikan potongan kode berikut.

```
#include "stdafx.h"
#include "iostream"
using namespace std;


void main()
{
    int a=10;

    cout << "Nilai a++: " << a++ << endl;

    cout << "Nilai a sekarang : " << a << endl;

    cout << "Nilai ++a : " << ++a << endl;
    system("pause");
}
```

Gambar 5.7 Operator Increment dan Decrement

Jika listing kode tersebut dieksekusi, maka hasil yang ditampilkan adalah sebagai berikut.



Gambar 5.8 Hasil Operator Increment dan Decrement

Dari hasil di atas, disimpulkan bahwa dengan menggunakan postfix increment, nilai awal a (10) akan dikembalikan terlebih dahulu sebelum ditambah 1. Ketika nilai a dipanggil kembali pada baris berikutnya, nilai a sudah berubah menjadi 11. Sedangkan, dengan prefix increment, nilai a ditambah 1 terlebih dahulu untuk kemudian dikembalikan.

4. Ekspresi

Ekspresi adalah ungkapan yang menghasilkan sebuah nilai dan terdiri atas operand dan operator. Beberapa macam ekspresi dalam C++, di antaranya:

1) Ekspresi aritmatika
   Ekspresi yang menggunakan operator aritmatika serta memiliki operand dan hasil yang bertipe numerik.
   Misal: 3 + 4
   - Operator: +
   - Operand: 3 dan 4
2) Ekspresi relasional
   Ekspresi yang digunakan untuk membandingkan suatu nilai bertipe numerik dan menghasilkan nilai bertipe boolean. Ekspresi ini menggunakan operator relasional.
3) Ekspresi logika
   Ekspresi yang digunakan untuk membandingkan suatu nilai bertipe boolean dan menghasilkan nilai yang juga bertipe boolean. Operator yang digunakan adalah operator logika.

C. Uji Kemampuan

1. Sebutkan macam-macam tipe data bilangan bulat!
2. Jelaskan perbedaan tipe data float dan double!
3. Perhatikan potongan kode berikut!

```
char kar1 ='*';
char kar2[] = "********";
```

   Jelaskan perbedaan jenis tipe data dari kedua variabel yang dideklarasikan pada potongan kode di atas!
4. Sebutkan aturan pendeklarasian suatu variabel dalam C++!
5. Tentukan pendeklarasian variabel yang benar (B) dan yang salah (S)
   a. `int bilangan = 3;`
   b. `char huruf = “c++”;`
   c. `int bil genap = 2;`
   d. `char kar=’A’;`
   e. `float 1angka = 4.50;`
   f. `bool switch = true;`
6. Jelaskan perbedaan fungsi *prefix increment* dan *postfix increment*!
7. Sebutkan contoh ekspresi aritmetika, relasional, dan logika dalam C++!

KEGIATAN BELAJAR 6

STRUKTUR KONTROL PERCABANGAN

A. Tujuan Pembelajaran

Setelah melakukan kegiatan pembelajaran siswa dapat:
1. Menjelaskan percabangan 1 kondisi
2. Menjelaskan percabangan 2 kondisi
3. Menjelaskan percabangan lebih dari 2 kondisi
4. Menjelaskan percabangan bersarang
5. Menerapkan percabangan 1 kondisi
6. Menerapkan percabangan 2 kondisi
7. Menerapkan percabangan lebih dari 2 kondisi
8. Menerapkan percabangan bersarang

B. Deskripsi Materi

Sebelum mempelajari materi pembelajaran ini, bacalah informasi di bawah ini dan jawab pertanyaan di bawahnya.

Tim pengelola taman hiburan akan menambah wahana baru, yaitu wahana roller coaster, untuk menarik minat pengunjung pada liburan akhir tahun. Tim tersebut sedang mempertimbangkan syarat-syarat bagi pengunjung yang diperbolehkan menikmati wahana baru tersebut.

Jika kamu adalah salah satu dari anggota tim pengelola tersebut, apa solusi yang dapat kamu berikan?
Coba buat algoritma yang menjelaskan solusi yang kamu buat untuk permasalahan di atas.

Struktur percabangan adalah pemilihan statement atau pernyataaan yang akan dieksekusi berdasarkan kondisi atau syarat tertentu. Dalam C++, terdapat dua buah jenis struktur yang digunakan dalam percabangan, yaitu if dan switch.

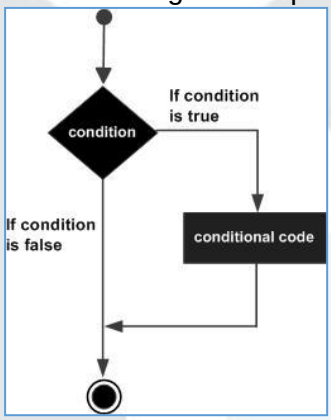
1. If statement

Pernyataan yang terdapat dalam sebuah struktur percabangan akan dieksekusi jika kondisi bernilai benar. Jika kondisi bernilai salah, maka pernyataan dalam blok tersebut tidak akan dieksekusi dan melanjutkan ke baris kode berikutnya. Sintaks dari if statement dalam C++ adalah sebagai berikut.

```
If (kondisi) {
        //Dieksekusi jika kondisi bernilai benar
}
```

Gambar 6.1 Sintaks if statement

Untuk lebih jelasnya, perhatikan diagram alir pada Gambar 6.2.



Gambar 6.2 Diagram alir if statement

Kerjakan Latihan 1 untuk lebih memahami if statement.

Latihan 1

```
#include "stdafx.h"
#include "iostream"
using namespace std;


void main()
{
    int jawab;

    cout << "\n13 * 13= ";
    cin >> jawab;

    if (jawab == 169)
        cout << "Anda benar\n";
    cout << "Terima kasih\n";
    system("pause");
}
```

Gambar 6.3 Latihan 1

Hasil:

```
13 * 13= 169
Anda benar
Terima kasih
Press any key to continue . . .
```

Gambar 6.4 Hasil Latihan 1

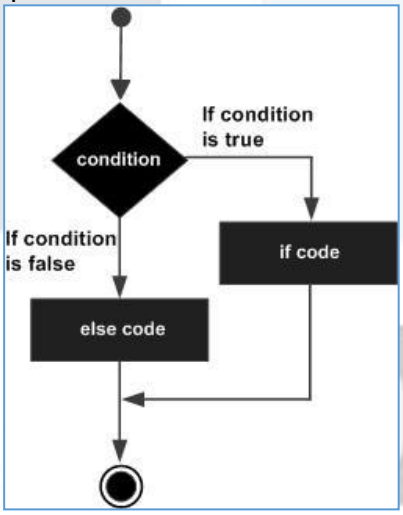
2. If else statement

Struktur percabangan yang memiliki opsi untuk kondisi benar dan kondisi salah. Jika kondisi bernilai benar, maka pernyataan di dalam blok if akan dieksekusi. Sebaliknya, jika kondisi bernilai salah, pernyataan di dalam blok else akan dieksekusi. Perhatikan sintaks berikut.

```
If (kondisi)
{
    //dieksekusi jika kondisi bernilai benar
}
else
{
    //dieksekusi jika kondisi bernilai salah
}
```

Gambar 6.5 Sintaks if else statement

Untuk lebih jelasnya perhatikan Gambar 6.6.



Gambar 6.6 Diagram alir if statement

Kerjakan Latihan 2 untuk lebih memahami if else statement.

Latihan 2

```
#include "stdafx.h"
#include "iostream"
using namespace std;


void main()
{
    int nilai;

    cout << "\nMasukkan nilai Anda: ";
    cin >> nilai;

    if (nilai > 79)
    {
        cout << "Bagus sekali. Tingkatkan ya..\n";
    }
    else
    {
        cout << "Sayang sekali. Tetap semangat..\n";
    }
    system("pause");
}
```

Gambar 6.7 Latihan 2

Hasil:

```
Masukkan nilai Anda: 60
Sayang sekali. Tetap semangat..
Press any key to continue . . .
```

Gambar 6.8 Hasil Latihan 2

3. Nested if statements

Struktur percabangan yang berada di dalam struktur percabangan lain. Berikut ini adalah sintaks nested if statements dalam C++. Berikut ini adalah sintaks nested if statements dalam C++

```
If (kondisi1)
{
    //Mengeksekusi if statement berikut jika kondisi1 bernilai benar

    If(kondisi2)
    {
        //Dieksekusi jika kondisi2 bernilai benar
    }
}
```

Gambar 6.9 Sintaks Nested if statements

Kerjakan Latihan 3 untuk lebih memahami nested if statement.

Latihan 3

```
#include "stdafx.h"
#include "iostream"
using namespace std;


void main()
{
    int angka;

    cout << "\nMasukkan angka lebih dari 2 yang habis dibagi 3: ";
    cin >> angka;

    if (angka > 2)
        if (angka%3==0)
        cout << "Angka " << angka << " lebih dari 2 dan habis dibagi 3\n";

system("pause");
}
```

Gambar 6.10 Latihan 3

Hasil:

```
Masukkan angka lebih dari 2 yang habis dibagi 3: 12
Angka 12 lebih dari 2 dan habis dibagi 3
Press any key to continue . . .
```

Gambar 6.11 Hasil Latihan 4

Tugas

Modifikasi listing kode Latihan 3 agar dapat menampilkan hasil seperti gambar berikut.

```
Masukkan angka lebih dari 2 yang habis dibagi 3: 5
Angka 5 lebih dari 2 tetapi tidak habis dibagi 3
Press any key to continue . . .
```

```
Masukkan angka lebih dari 2 yang habis dibagi 3: 1
Angka 1 lebih kecil dari 2
Press any key to continue . . .
```

4. **If-else if-else statement**

Struktur percabangan if else yang diikuti oleh pernyataan if else lain dan dapat digunakan untuk menguji beberapa kondisi berbeda. Berikut ini adalah sintaks if-else if-else statement dalam C++.

```
If (kondisi1)
{
   //Dieksekusi jika kondisi1 bernilai benar
}
else if (kondisi2)
{
   //Dieksekusi jika kondisi2 bernilai benar
}
...
...
else if(kondisi-n)
{
   //Dieksekusi jika kondisi-n bernilai benar
}
else
{
   //Dieksekusi jika semua kondisi bernilai salah
}
```

Gambar 6.12 Sintaks if-else if-else statement

Kerjakan Latihan 4 untuk lebih memahami if else if statement.

Latihan 4

```
#include "stdafx.h"
#include "iostream"
using namespace std;


void main()
{
    int bil1, bil2, bil3;

    cout << "\nMasukkan 3 bilangan secara acak \n";
    cout << "Bilangan I  : "; cin >> bil1;
    cout << "Bilangan II : "; cin >> bil2;
    cout << "Bilangan III: "; cin >> bil3;

    if (bil1>bil2 && bil1>bil3)
    {
        cout << bil1 << " adalah bilangan terbesar\n";
    }
    else if (bil2>bil1 && bil2>bil3)
    {
        cout << bil2 << " adalah bilangan terbesar\n";
    }
    else
    {
        cout << bil3 << " adalah bilangan terbesar\n";
    }



system("pause");
}
```

Gambar 6.13 Latihan 4

Hasil:

```
Masukkan 3 bilangan secara acak
Bilangan I   : 1
Bilangan II  : 2
Bilangan III: 3
3 adalah bilangan terbesar
Press any key to continue . . . _
```

Gambar 6.14 Hasil Latihan 4

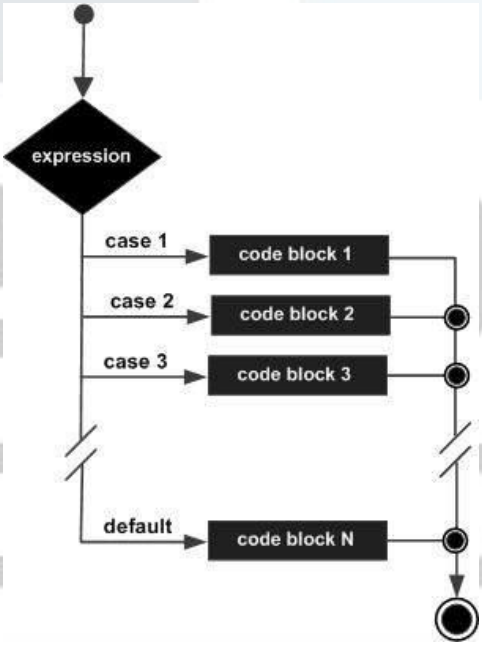
5. Switch statement

Struktur percabangan yang digunakan untuk menguji sebuah variabel dan mencocokkannya dengan beberapa nilai yang disebut *case*. Sintaks dari switch statement dalam C++ ditunjukkan oleh Gambar 6.15.

```
Switch(variabel)
{
case 'nilai1':
      //Dieksekusi jika memenuhi nilai1
      break;
case 'nilai2':
      //Dieksekusi jika memenuhi nilai2
      break;
case 'nilai-n':
      //Dieksekusi jika memenuhi nilai-n
      break;
default:
//dieksekusi jika variabel tidak memenuhi semua case yang ada
}
```

Gambar 6.15. Sintaks switch statement

Untuk lebih jelasnya, perhatikan gambar berikut.



Gambar 6.16 Diagram alir switch statement

Kerjakan Latihan 5 untuk memahami switch statement

Latihan 5

```
#include "stdafx.h"
#include "iostream"
using namespace std;

void main()
{
    char menu;

    cout << "\n== KANTIN SEHAT ==\n";
    cout << "Menu hari ini:\n\n";
    cout << "A. Nasi putih + Sayur bayam + Bakwan jagung\n";
    cout << "B. Nasi goreng + Telur ceplok + Sosis\n";
    cout << "C. Nasi putih + Ayam goreng kremes + Sambal\n";

    cout << "\nPilih menu yang ingin dipesan (A/B/C):";
    cin >> menu;

    switch (menu)
    {
    case 'A':
        cout << "\nPesanan Anda (Menu A) sudah dicatat..\nMohon tunggu sebentar\n";
        break;
    case 'B':
        cout << "\nPesanan Anda (Menu B) sudah dicatat..\nMohon tunggu sebentar\n";
        break;
    case 'C':
        cout << "\nPesanan Anda (Menu C) sudah dicatat..\nMohon tunggu sebentar\n";
        break;
    default:
        cout << "\nMohon tekan A/B/C untuk memilih menu\n";
        break;
    }
system("pause");
```

Gambar 6.17 Latihan 5

Hasil

```
== KANTIN SEHAT ==
Menu hari ini:

A. Nasi putih + Sayur bayam + Bakwan jagung
B. Nasi goreng + Telur ceplok + Sosis
C. Nasi putih + Ayam goreng kremes + Sambal

Pilih menu yang ingin dipesan (A/B/C):B

Pesanan Anda (Menu B) sudah dicatat..
Mohon tunggu sebentar
Press any key to continue . . .
```

Gambar 6.18 Hasil Latihan 5

Tugas

1. Berdasarkan kode program pada Latihan 5, apa yang terjadi jika karakter a/b/c yang diinputkan?
2. Apa fungsi dari *statement* break yang ditambahkan di setiap akhir *case*?

C. Uji Kemampuan

1. Dalam suatu struktur percabangan, perlu ditambahkan operator relasional untuk menentukan kondisi. Jelaskan definisi dari operator-operator berikut:
   a. ==
   b. !=
   c. >=
   d. <=
2. Tuliskan sintaks dasar dari struktur-struktur percabangan berikut:
   a. if statement
   b. if…else statement
   c. nested if statement
   d. switch statement
3. Jelaskan tujuan penambahan keyword *break* di akhir setiap *case* pada *switch statement*!
4. Buatlah program kalkulator sederhana dengan ketentuan sebagai berikut:
   - Bilangan diinputkan diinputkan oleh user (2 bilangan)
   - User dapat memilih macam-macam operasi yang ingin dilakukan
   - Terdapat min.3 macam operasi aritmetika
   - Bebas memilih jenis struktur percabangan yang akan digunakan
5. Sebuah toko dengan nama ”HALAL MAKMUR JAYA”, dalam melayani pembeli, mempunyai ketentuan dalam memberikan potongan harga sebagai berikut:
   - Tidak ada potongan jika total pembelian kurang dari Rp.50.000
   - Jika total pembelian lebih dari atau sama dengan Rp.50.000 potongan yang diterima sebesar 20% dari total pembelian.

   Buat program yang menerapkan ketentuan-ketentuan tersebut dan juga menampilkan nama petugas serta pembeli!

KEGIATAN BELAJAR 7

STRUKTUR KONTROL PERULANGAN

A. Tujuan Pembelajaran

Setelah melakukan kegiatan pembelajaran siswa dapat :

1. Menjelaskan perulangan dengan kondisi di awal
2. Menjelaskan perulangan dengan kondisi di akhir
3. Menjelaskan perulangan dengan kondisi akhir diinputkan
4. Menjelaskan perulangan dengan pernyataan *continue*
5. Menjelaskan perulangan dengan pernyataan *break*
6. Menyajikan perulangan dengan kondisi di awal
7. Menyajikan perulangan dengan kondisi di akhir
8. Menyajikan perulangan dengan kondisi akhir diinputkan
9. Menyajikan perulangan dengan pernyataan *continue*
10. Menyajikan perulangan dengan pernyataan *break*

B. Deskripsi Materi

Sebelum mempelajari materi pembelajaran ini, bacalah uraian di bawah ini dan jawab pertanyaan di bawahnya.

```
void main() {
    cout << "Belajar C++\n";
    cout << "Belajar C++\n";
    cout << "Belajar C++\n";
    cout << "Belajar C++\n";
    cout << "Belajar C++\n";
    cout << "Belajar C++\n";
    cout << "Belajar C++\n";
    cout << "Belajar C++\n";
    cout << "Belajar C++\n";
    cout << "Belajar C++\n";
}
```

Budi sedang mengerjakan tugas membuat program yang mencetak kalimat “Belajar C++” sebanyak 10 kali. Untuk menyelesaikannya, Budi memutuskan untuk menambahkan perintah cout sebanyak 10 kali, sesuai ketentuan tugas tersebut.

Bagaimana pendapatmu mengenai metode yang digunakan Budi? Dapatkah kamu memberi solusi lain yang lebih efektif

Struktur kontrol digunakan untuk mengeksekusi suatu blok kode secara berulang hingga batas yang ditentukan. Jenis perulangan dapat dibagi menjadi dua kategori utama, yaitu perulangan yang sudah diketahui batas jumlah perulangannya dna perulangan yang belum diketahui batas jumlah perulangannya. Terdapat beberapa macam keyword yang digunakan untuk membuat sebuah struktur kontrol perulangan, di antaranya:

1. For loop

For loop digunakan untuk mengeksekusi suatu blok kode yang sudah diketahui jumlah perulangannya. Berikut ini adalah sintaks for loop dalam C++.

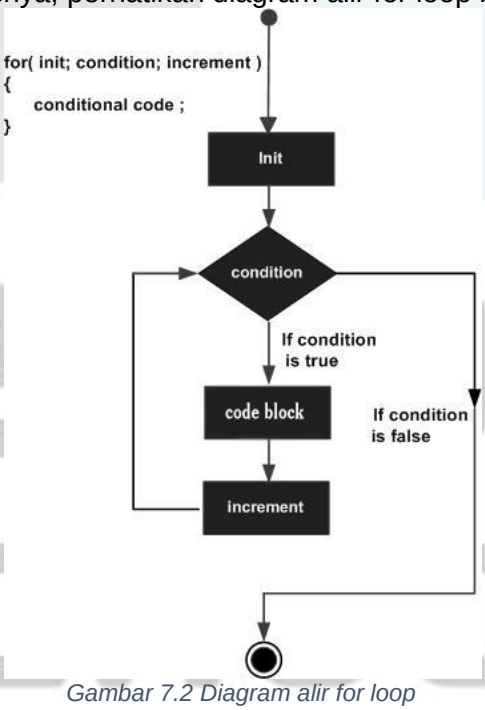
```
for (init;kondisi;increment)
{
   //Dieksekusi hingga batas yang ditentukan
}
```

Gambar 7.1 Sintaks for loop

Penjelasan mengenai sintaks for loop diuraikan sebagai berikut:

a) **init** adalah variabel yang menjadi nilai awal sekaligus counter atau penghitung dalam for loop
b) **kondisi** adalah syarat yang berfungsi mengendalikan jalannya perulangan. Selama kondisi terpenuhi, maka blok kode akan terus diulang.
c) **increment** merupakan ekspresi yang melaksanakan penambahan nilai init setiap dilakukan perulangan.

Untuk lebih jelasnya, perhatikan diagram alir for loop berikut.



Gambar 7.2 Diagram alir for loop

Kerjakan Latihan 1 untuk lebih memahami for loop.

Latihan 1

```
#include "stdafx.h"
#include "iostream"
using namespace std;

void main()
{
    cout << "\nMencetak angka 1-10\n\n";
    for (int X = 1; X<=10; X++)
    {cout << "\n " << X;}

    cout << "\n\n";
system("pause");
}
```

Gambar 7.3 Latihan 1

Hasil

```
Mencetak angka 1-10


 1
 2
 3
 4
 5
 6
 7
 8
 9
 10

Press any key to continue . . .
```

Gambar 7.4 Hasil Latihan 1

Tugas

Buat listing kode untuk menampilkan hasil sebagai berikut:

```
Mencetak angka 10-1


 10
 9
 8
 7
 6
 5
 4
 3
 2
 1

Press any key to continue . . .
```

```
Mencetak angka kelipatan 10 antara 1-100
 10
 20
 30
 40
 50
 60
 70
 80
 90
 100

Press any key to continue . . .
```

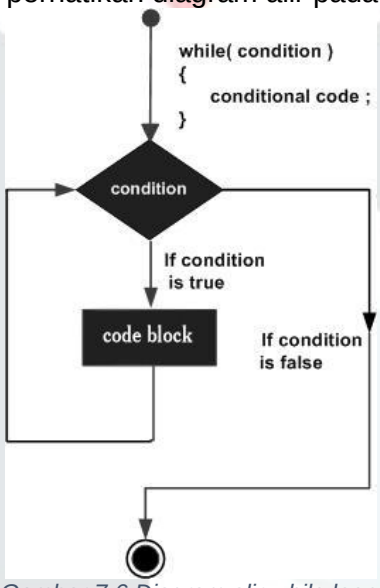
2. While loop

Pada while loop, pengecekan kondisi dilakukan sebelum dilakukan perulangan. Oleh karena itu, while loop dapat digunakan untuk mengeksekusi suatu blok kode yang belum diketahui jumlah perulangannya. Perulangan akan terus dilakukan selama kondisi yang ditentukan masih terpenuhi. Berikut ini adalah sintaks while loop dalam C++.

```
while(kondisi)
{
   //Dieksekusi selama kondisi terpenuhi
   //blok kode increment
}
```

Gambar 7.5 Sintaks while loop

Untuk lebih jelasnya, perhatikan diagram alir pada Gambar 7.6.



Gambar 7.6 Diagram alir while loop

Kerjakan Latihan 2 untuk lebih memahami while loop.

Latihan 2

```
#include "stdafx.h"
#include "iostream"
using namespace std;

void main()
{
    int counter=50;

    while (counter>=5)
    {
        cout << counter <<"\n";

        counter-=5;
    }
    system("pause");
}
```

Gambar 7.7 Latihan 2

Hasil:

```
50
45
40
35
30
25
20
15
10
5
Press any key to continue . . .
```

Gambar 7.8 Hasil Latihan 2

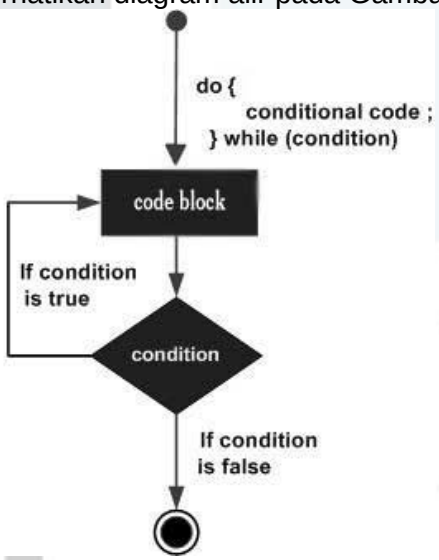
3. Do…while loop

Berbeda dengan perulangan lainnya, do…while loop melakukan pengecekan kondisi di akhir. Berikut ini adalah sintaks do…while loop dalam C++.

```
Do
{
   //blok kode yang akan dieksekusi
   //blok kode inkremen
}
while(kondisi);
```

Gambar 7.9 Sintaks do…while loop

Berdasarkan sintaks di atas, dapat diketahui bahwa blok kode di dalam do akan selalu dieksekusi setidaknya satu kali meskipun kondisi tidak terpenuhi. Untuk lebih jelasnya, perhatikan diagram alir pada Gambar 7.10



Gambar 7.10 Diagram alir do…while loop

Kerjakan Latihan 3 untuk lebih memahami do…while loop.

Latihan 3

```
#include "stdafx.h"
#include "iostream"
using namespace std;

void main()
{
    int i=1;

    do
    {

        cout << "Belajar Pemrograman Dasar\n";
        i++;
    } while (i<=5);
    system("pause");
}
```

Gambar 7.11 Latihan 3

Hasil:

```
Belajar Pemrograman Dasar
Belajar Pemrograman Dasar
Belajar Pemrograman Dasar
Belajar Pemrograman Dasar
Belajar Pemrograman Dasar
Press any key to continue . . .
```

Gambar 7.12 Hasil Latihan 3

Tugas

1. Apa yang terjadi jika kondisi diubah i<2? Bagaimana hasil yang ditampilkan pada program?

2. Letakkan baris *increment* (i++) sebelum perintah cetak “Belajar Pemrograman Dasar”. Apakah terdapat perbedaan dari hasil yang ditampilkan?

4. Nested loops

Nested loops adalah struktur kontrol perulangan (for, while, do…while) yang berada di dalam struktur kontrol perulangan lain. C++ memungkinkan sebuah nested loop memiliki sebanyak 256 tingkat perulangan. Sintaks nested loops menggunakan for loop ditunjukkan oleh Gambar 7.13.

```
for(init1;kondisi1;increment1)
{
    for (init2;kondisi2;increment2)
    {
        //blok kode perulangan 2
    }
    //blok kode perulangan 2
}
```

Gambar 7.13 Nested loops dengan for

Sintaks nested loops menggunakan while loop ditunjukkan oleh Gambar 7.14.

```
while(kondisi1)
{
    while(kondisi2)
    {
        //blok kode perulangan 2
    }

    //blok kode perulangan 1
}
```

Gambar 7.14 Nested loops dengan while

Sedangkan, sintaks nested loops dengan do…while ditunjukkan oleh Gambar 7.15.

```
do
{
    //blok kode perulangan 1
    do
    {
        //blok kode perulangan 2
    }
    while(kondisi2)
}
while(kondisi1)
```

Gambar 7.15 Sintaks nested loops dengan do…while

Kerjakan Latihan 4, Latihan 5 untuk lebih memahami nested loops.

Latihan 4

Di latihan ini, kondisi akhir perulangan diinputkan oleh user.

```
#include "stdafx.h"
#include "iostream"
using namespace std;

void main()
{
    int sisi;

    cout << "\nPERSEGI ANGKA\n";
    cout << "\nMasukkan sisi persegi: "; cin >> sisi;
    for (int x = 1; x<=sisi; x++)
    {
        cout << "\n";
        for (int y = 1; y<=sisi; y++)
            {
               cout << " " << y;
            }
    }
   cout << "\n\n "; system("pause");
}
```

Gambar 7.16 Latihan 4

Hasil

```
PERSEGI ANGKA

Masukkan sisi persegi: 5

 1 2 3 4 5
 1 2 3 4 5
 1 2 3 4 5
 1 2 3 4 5
 1 2 3 4 5

 Press any key to continue . . .
```

Gambar 7.17 Hasil Latihan 4

Tugas

1. Apa fungsi dari baris kode perulangan yang pertama?
   `for (int x = 1; x<=sisi; x++)`

2. Apa fungsi dari baris kode perulangan yang kedua?
   `for (int y = 1; y<=sisi; y++)`

Latihan 5

```
#include "stdafx.h"
#include "iostream"
using namespace std;


void main()
{
    for (int I = 1; I <=5; I++)
    {
        for (int J = 1; J <= I ; J++)
            { cout << " " << J; }
        cout << "\n";
    }

    system("pause");
}
```

Gambar 7.18 Latihan 5

Hasil:

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
Press any key to continue . . .
```

Gambar 7.19 Hasil Latihan 5

Tugas

Buat listing kode untuk menampilkan hasil seperti gambar berikut:

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1


Press any key to continue . . .
```

5. Break dan Continue

Pernyataan *break* dan *continue* ditambahkan di untuk mengubah alur dari sebuah struktur kontrol perulangan. Berikut ini adalah penjelasan lebih rinci mengenai pernyataan *break* dan *continue*.

a) *Break*

Break ditambahkan di dalam perulangan untuk menghentikan proses perulangan meskipun kondisi masih bernilai *true*.

Untuk lebih memahami penggunaan break, kerjakan Latihan 6.

Latihan 6

```
#include "stdafx.h"
#include "iostream"
using namespace std;


void main()
{
    int counter=0;
    do {
        counter++;
        if(counter == 7)
            break;

        cout << " Counter : " << counter << "\n";
    }
    while (counter<10);
    system("pause");
}
```

Gambar 7.20 Latihan 6

Hasil

```
Counter : 1
Counter : 2
Counter : 3
Counter : 4
Counter : 5
Counter : 6
Press any key to continue . . .
```

Gambar 7.21 Hasil Latihan 7

b) *Continue*

Continue ditambahkan di dalam perulangan untuk meloncati proses perulangan pada kondisi tertentu kemudian dilanjutkan pada proses perulangan berikutnya.

Untuk lebih memahami penggunaan *continue*, kerjakan Latihan 7.

Latihan 7

```
#include "stdafx.h"
#include "iostream"
using namespace std;


void main()
{
    int hitung=0;
    do {
        hitung++;
        if(hitung == 5)
            continue;

        cout << " Hitung : " << hitung << "\n";
    }
    while (hitung<10);
    system("pause");
}
```

Gambar 7.22 Latihan 7

Hasil:

```
Hitung : 1
Hitung : 2
Hitung : 3
Hitung : 4
Hitung : 6
Hitung : 7
Hitung : 8
Hitung : 9
Hitung : 10
Press any key to continue . . .
```

Gambar 7.23 Hasil Latihan 7

C. Uji Kemampuan

1. Perulangan dapat digolongkan menjadi dua kategori utama. Jelaskan dan sebutkan pernyataan perulangan yang sesuai untuk masing-masing kategori!
2. Bagaimana perbedaan mekanisme perulangan dengan pernyataan while dan do…while loop?
3. Perhatikan dua potongan kode berikut!

```
for (int i = 1; i <= 10; i++)
{
    cout << i << " ";
}
```

```
for (int i = 10; i >= 1; i--)
{
    cout << i << " ";
}
```

   Bagaimana hasil dari program ketika kedua potongan tersebut dieksekusi?

4. Perhatikan potongan kode berikut!

```
for (int i = 1; i > 0; i++)
{
    cout << i;
}
```

   Apa yang akan terjadi jika program tersebut dieksekusi?
5. Jelaskan fungsi dari:
   a. *break*
   b. *continue*

KEGIATAN BELAJAR 8

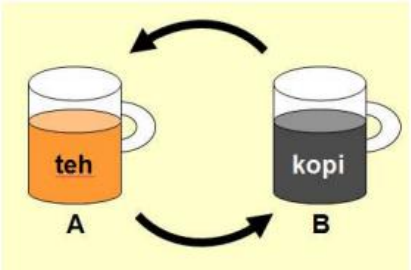
PENGEMBANGAN ALGORITMA APLIKASI

A. Tujuan Pembelajaran

Setelah melakukan kegiatan pembelajaran siswa dapat :
1. Menjelaskan langkah-langkah merancang program komputer untuk permasalahan yang kompleks
2. Menjelaskan debugging dan error handling
3. Membuat desain dan analisa algoritma program untuk penyelesaian permasalahan kompleks
4. Mengembangkan program untuk penyelesaian permasalahan kompleks
5. Menerapkan debugging dan error handling

B. Deskripsi Materi

Sebelum mempelajari materi pembelajaran ini, bacalah uraian di bawah ini dan jawab pertanyaan di bawahnya.



Terdapat dua gelas A dan B, gelas A berisi air teh dan gelas B berisi air kopi. Isi gelas A dan B akan ditukar sehingga gelas A yang semula berisi air teh menjadi berisi air kopi, dan begitu pula sebaliknya pada gelas B.

1. Apa permasalahan yang diungkapkan pada uraian di atas?
2. Jelaskan solusi sederhana untuk menyelesaikan permasalahan di atas.
3. Apa saja kebutuhan yang harus tersedia untuk menyelesaikan masalah di atas?
4. Berdasarkan solusi dan kebutuhan yang telah ditentukan, buat kumpulan instruksi-instruksi untuk melaksanakan pemecahan masalah di atas.

1. Langkah-Langkah Pengembangan Program

Dalam mengembangkan sebuah aplikasi, terdapat beberapa tahap yang harus dilalui, di antaranya:

1) Menentukan masalah
   Sebuah aplikasi dibuat untuk memberikan solusi dari suatu permasalahan yang kompleks. Permasalahan tersebut perlu dijabarkan terlebih dahulu agar diketahui solusi-solusi yang tepat untuk diwujudkan dalam aplikasi yang akan dikembangkan.
2) Merencanakan dan merancang aplikasi
   Berdasarkan analisis permasalahan, dibuat rancangan yang diwujudkan dalam suatu algoritma pemrograman. Algoritma himpunan instruksi-instruksi yang memperinci langkah-langkah proses pelaksanaan dalam pemecahan suatu masalah. Algoritma dapat dibuat dengan berbagai macam cara, di antaranya dengan kalimat yang digunakan sehari-hari, simbol grafik bagan alir (flowchart), dan bahasa pseudocode, yaitu bahasa yang menyerupai bahasa pemrograman.
3) Mengimplementasi
   Pada tahap ini, aplikasi dibuat menggunakan bahasa pemrograman dengan mengacu pada algoritma yang telah dibuat pada tahap sebelumnya.
4) Mendokumentasi program
   Pada tahap ini, dibuat dokumentasi aplikasi yang berisi komentar-komentar cuplikan program. Fungsinya adalah untuk menjelaskan variabel-variabel, parameter, prosedur, dan fungsi. Dokumentasi akan mempermudah pengembang atau pihak lain yang akan akan melakukan perbaikan pada program.
5) Menguji program
   Program yang telah dibuat perlu diuji untuk mengetahui apakah program tersebut layak untuk digunakan.
6) Merawat program
   Program yang telah dibuat dan melalui pengujian perlu dirawat untuk mendeteksi bug yang tidak diketahui sebelumnya dan memperbaharui fitur-fitur yang mempermudah pengguna program

Tugas

1) Sebutkah tahap-tahap pengembangan program!
2) Jelaskan apa yang dimaksud dengan tahap merencanakan dan merancang dalam pengembangan aplikasi!
3) Mendokumentasi merupakan salah satu tahap dalam pengembangan aplikasi. Jelaskan tujuan dari kegiatan tersebut!
4) Apa tujuan dari tahap perawatan aplikasi?

2. Debugging dan Error Handling

Dalam mengembangkan sebuah program, tidak dapat dipungkiri bahwa akan ditemui error atau bug pada jalannya program karena kesalahan yang dibuat oleh pengembang. Oleh karena itu, tahap implementasi bukan merupakan tahap akhir dalam pengembangan sebuah program. Seorang

pengembang harus memastikan bahwa program yang dikembangkan memiliki resiko sekecil mungkin untuk menemui error atau bug. Hal ini dapat dicapai dengan *debugging* dan *error handling*.

1) *Debugging*

Setelah program selesai dibuat, pengembang harus mendeteksi apakah terdapat kesalahan pada program. Jika ditemui kesalahan, maka pengembang akan mencari solusi untuk memperbaikinya.. Kegiatan ini disebut *debugging* dan memiliki tujuan mengurangi sebanyak mungkin resiko kesalahan pada jalannya program.

Kesalahan atau error yang mungkin muncul selama jalannya program dapat dikategorikan menjadi dua jenis, yaitu *syntax errors* dan *semantic errors*. Syntax errors atau kesalahan sintaks timbul ketika pengembang salah dalam menuliskan kode yang sesuai dengan aturan bahasa pemrograman yang dipakai. Misal, perhatikan sintaks pemrograman C++ berikut.

```
int 123angka;
```

Seperti yang kalian ketahui, C++ telah mengatur bahwa sebuah variabel tidak boleh memiliki nama yang diawali oleh angka dan untuk memperbaikinya cukup dengan mengganti nama sesuai aturan yang telah ditetapkan. Kesalahan semacam ini digolongkan sebagai *syntax errors* dan biasanya akan langsung dideteksi oleh *compiler* ketika program dieksekusi. Oleh karena itu, syntax error biasanya tidak terlalu kompleks sehingga cukup mudah dalam perbaikannya.

Jenis *error* yang kedua adalah *semantic error*, yaitu kesalahan yang terjadi ketika sebuah program tidak berjalan semestinya meskipun dari segi penulisan kode sudah valid. Misal, potongan kode berikut digunakan untuk mencetak “C++” sebanyak 10 kali.

```
for (int i = 10; i >= 10; i++)
{
    cout << "C++ ";
}
```

Gambar 8.1 Contoh Semantic Error

Ketika listing kode di atas dieksekusi, maka tentu akan terjadi infinity loop, atau proses perulangan secara terus menerus tanpa henti, dan hanya dapat dihentikan ketika jendela program ditutup. Hal ini karena terdapat kesalahan pada inisialisasi dan kondisi perulangan. Untuk memperbaikinya, maka nilai i diubah menjadi `1` dan kondisi diubah menjadi `i <= 10`. Semantic error tidak diketahui oleh compiler tetapi akan terlihat ketika program dijalankan. Pada kasus-kasus lain, tingkat kerumitan *semantic error* akan lebih tinggi, tergantung pada skala program yang dikembangkan.

2) *Error handling*

Error handling adalah kegiatan mengantisipasi kesalahan-kesalahan tidak terduga yang mungkin muncul ketika program berjalan. Selama pengembangan program, pengembang dapat menambahkan listing kode yang mampu 'menangkap' kesalahan-kesalahan tidak terduga tersebut, dan kemudian mengatasinya. Misal, program berikut akan membagi dua bilangan yang diinput oleh *user*.

```
float a,b,hasil;

cout << "\nMasukkan bilangan I : "; cin >> a;
cout << "Masukkan bilangan II: "; cin >> b;

hasil = a/b;

cout << a << " : " << b << " = " << hasil << endl;
```

Gambar 8.2 Listing kode untuk membagi dua bilangan

Dengan kode pemrograman di atas, program dapat berjalan sebagaimana mestinya. Namun, masalah akan muncul jika user menginputkan nilai 0 untuk bilangan kedua atau variabel b. Jika hal ini terjadi, akan muncul error message dan akhirnya program akan berhenti. Dengan error handling, seorang pengembang dapat menambahkan kode yang akan mengatasi kesalahan-kesalahan tidak terduga seperti ini.

Untuk listing kode seperti ditunjukkan Gambar 8.2, kita dapat menambahkan baris kode yang akan mengecek nilai dari variabel b sebelum dilakukan operasi pembagian. Perhatikan Gambar 8.3.

```
float a,b,hasil;

cout << "\nMasukkan bilangan I : "; cin >> a;
cout << "Masukkan bilangan II: "; cin >> b;

if (b == 0)
{
    cout << "Bilangan ke-2 tidak boleh bernilai 0";
    cout << "\n\nKeluar program...";

}
else
{
    hasil = a/b;
    cout << a << " : " << b << " = " << hasil << endl;
}
```

Gambar 8.3 Contoh error handling

Dengan listing kode seperti ditunjukkan Gambar 8.3, program tidak harus dihentikan secara paksa ketika nilai variabel b yang diinputkan adalah 0. Sebaliknya, program akan menampilkan pesan bahwa nilai bilangan kedua tidak boleh 0, sebelum akhirnya user keluar dari program.

Tugas

1) Jelaskan yang dimaksud dengan *debugging*!
2) Jelaskan yang dimaksud dengan *syntax error* dan *semantic error*, serta berikan contoh!
3) Apa yang dimaksud dengan *error handling*?

3. Pengembangan Aplikasi untuk Penyelesaian Masalah Kompleks

Judul aplikasi : Program Kalkulator Luas dan Volume Bangun Ruang
Spesifikasi : Program dengan struktur percabangan switch bersarang untuk menghitung luas serta volume bangun ruang (kubus, balok, dan tabung)
Bahasa pemrograman : C++

a) Analisis permasalahan

Terdapat bermacam-macam bangun ruang dengan rumus penghitungan luas dan volume yang berbeda-beda. Kalkulator pada umumnya hanya dimanfaatkan pada operasi aritmetika saja. Oleh karena itu, dibutuhkan sebuah aplikasi kalkulator yang secara spesifik dapat menentukan luas dan volume berbagai macam bangun ruang dengan tepat dan akurat.

Untuk mengembangkan aplikasi ini dengan C++, metode yang tepat adalah menggunakan struktur percabangan *nested switch* atau switch bersarang. Dengan nested switch, user diberi kebebasan untuk memilih menu sesuai kebutuhan.

Selain itu, dibutuhkan pula variabel-variabel untuk menyimpan data yang diinput user. Data tersebut berupa nilai menu yang dipilih user serta nilai yang akan dioperasikan untuk menghitung luas dan volume bangun ruang. Tipe data untuk setiap variabel yang digunakan dalam penghitungan luas dan volume adalah double agar nilai yang dihasilkan akurat. Sedangkan, variabel yang menyimpan data yang diinput user ketika memilih menu adalah integer dan char.

b) Algoritma

1. Menginput 1 untuk menu luas bangun ruang atau 2 untuk menu volume bangun ruang
2. Jika 1, menginput a untuk luas kubus, b untuk luas balok, dan c untuk luas tabung
3. Untuk luas kubus, input nilai ke variabel sisi dan hitung luas_kubus = 6*sisi*sisi
4. Cetak luas_kubus
5. Untuk luas balok, input nilai ke variabel p, l, dan t_balok hitung luas_balok = 2*p*l+2*p* t_balok +2*l* t_balok
6. Cetak luas_balok
7. Untuk luas tabung, input nilai ke variabel r dan t_tabung dan hitung luas_tabung = 2*3.14*jari*jari+2*3.14*jari*t_tabung
8. Cetak luas_tabung
9. Jika 2, menginput ‘a’ untuk volume kubus, ‘b’ untuk volume balok, dan ‘c’ untuk volume tabung

10. Untuk volume kubus, input nilai ke variabel sisi dan hitung vol_kubus = sisi*sisi*sisi
11. Cetak vol_kubus
12. Untuk volume balok, input nilai ke variabel p, l, dan t_balok dan hitung vol_balok = p*l*t_balok
    Cetak vol_balok
13. Untuk volume tabung, input nilai ke variabel r dan t_tabung dan hitung vol_tabung = 3.14*jari*jari*t_tabung
14. Cetak vol_tabung

c) Kode pemrograman
Kode pemrograman dapat dilihat pada Lampiran.

d) Error handling
Untuk *error handling*, pada keyword default ditambahkan baris kode untuk menampilkan pesan ketika *user* salah menginputkan menu. Selain itu, ditambahkan dua *value case* untuk setiap menu luas atau volume bangun ruang yang dipilih, seperti berikut:

```
case 'a':
case 'A':
  cout << "Anda memilih Menghitung Luas Permukaan Kubus\n";
  cout << "Masukkan sisi kubus : "; cin >> sisi; cout << "cm\n";
  luas_kubus = 6*sisi*sisi;
  cout << "Maka Luas Permukaan kubus :" << luas_kubus << "cm2\n";
  break;
```

C. Uji Kemampuan

Buat program C++ untuk penyelesaian masalah kompleks dalam kehidupan sehari-hari. Pada laporan, sertakan analisis permasalah, algoritma, rincian error handling, serta dokumentasi aplikasi (penjelasan variabel, debugging, dll.).

DAFTAR PUSTAKA

Kadir, Abdul. (2004). *Panduan Pemrograman Visual C++.* Yogyakarta: ANDI.

Liberty, Jesse, dan Rogers Cadenhead. (2011). *Sams Teach Yourself C++ in 24 Hours.*Indianapolis: Pearson Education Inc.

Prata, Stephen. (2012). *C++ Primer Plus Sixth Edition*. Indianapolis: Pearson Education, Inc.

Suprapto. (2008). Bahasa Pemrograman. Jakarta:Departemen Pendidikan Nasional.

Sutedjo, Budi. (2009). *Algoritma dan Teknik Pemrograman.* Yogyakarta: ANDI.

Tutorials Point. *Learn C++ Programming Language* [pdf], (http://www.tutorialspoint.com/cplusplus/cpp_tutorial.pdf, diakses tanggal 23 Desember 2015)

Munir, Rinaldi. 2011. Algoritma dan Pemrograman dalam Bahasa Pascal. Bandung: Informatika.

LAMPIRAN

Kalkulator Luas dan Volume Bangun Ruang

Source code:

```
#include "iostream"
#include "stdafx.h"
using namespace std;

void main()
{
int pilih1;
char pilih2,pilih3;
double sisi,luas_kubus, vol_kubus;
double panjang,lebar,tinggi_balok,luas_balok, vol_balok;
double jari, tinggi_tabung, luas_tabung, vol_tabung;

cout << "Main Menu Bangun Ruang (Kubus, Balok, Tabung) : " << endl;
cout << "1. Menghitung Luas Permukaan" << endl;
cout << "2. Menghitung Volume" << endl;

cout << "Anda Memilih : "; cin >> pilih1;

switch(pilih1)
{
  case 1:
         cout << "Menu Menghitung Luas Permukaan :" << endl;
         cout << "a. Luas Kubus" << endl;
         cout << "b. Luas Balok" << endl;
         cout << "c. Luas Tabung" << endl;
         cout << "Anda memilih : "; cin >> pilih2;

           switch(pilih2)
           {
         case 'a':
                 case 'A':
                 cout << "Anda memilih Menghitung Luas Permukaan Kubus" << endl;
                 cout << "Masukkan sisi kubus : "; cin >> sisi; cout << "cm" <<       endl;
                         luas_kubus = 6*sisi*sisi;
                         cout << "Maka Luas Permukaan kubus :" << luas_kubus << "cm2" << endl;
                         break;
                 case 'b':
                 case 'B':
                         cout << "Anda Memilih Menghitung Luas Permukaan Balok" << endl;
                         cout << "Masukkan Panjang Balok  : "; cin >> panjang; cout << "cm" << endl;
                         cout << "Masukkan Lebar Balok     : "; cin >> lebar; cout << "cm" << endl;
                         cout << "Masukkan Tinggi Balok    : ";
                          cin >> tinggi_balok; cout << "cm" << endl;
                         luas_balok = 2*panjang*lebar+2*panjang*tinggi_balok+2*lebar*tinggi_balok;
                         cout << "Maka Luas Permukaan Balok :" << luas_balok << "cm2" << endl;
                         break;
                 case 'c':
                 case 'C':
                         cout << "Anda Memilih Menghitung Luas Permukaan Tabung" << endl;
                         cout << "Masukkan Jari-Jari Tabung : ";
                          cin >> jari; cout << "cm" << endl;
                         cout << "Masukkan Tinggi Tabung : ";
                          cin >> tinggi_tabung; cout << "cm" << endl;
                         luas_tabung = 2*3.14*jari*jari+2*3.14*jari*tinggi_tabung;
                         cout << "Maka Luas Permukaan Tabung : " << luas_tabung << "cm2" << endl;
                         break;
                 }
                 break;

         case 2:
                 cout << "Menu Menghitung Volume :" << endl;
                 cout << "a. Volume Kubus" << endl;
                 cout << "b. Volume Balok" << endl;
                 cout << "c. Volume Tabung" << endl;
                 cout << "Anda memilih : "; cin >> pilih3;
```

```
                switch(pilih3)
                {
                case 'a':
                case 'A':
                        cout << "Anda memilih Menghitung Volume Kubus" << endl;
                        cout << "Masukkan sisi kubus : "; cin >> sisi; cout << "cm" << endl;
                        vol_kubus = sisi*sisi*sisi;
                        cout << "Maka Volume kubus :" << vol_kubus << "cm3" << endl;
                        break;
                case 'b':
                case 'B':
                        cout << "Anda Memilih Menghitung Volume Balok" << endl;
                        cout << "Masukkan Panjang Balok : ";
                        cin >> panjang; cout << "cm" << endl;
                        cout << "Masukkan Lebar Balok     : ";
                        cin >> lebar; cout << "cm" << endl;
                        cout << "Masukkan Tinggi Balok    : ";
                        cin >> tinggi_balok; cout << "cm" << endl;
                        vol_balok = panjang*lebar*tinggi_balok;
                        cout << "Maka Volume Balok :" << vol_balok << "cm3" << endl;
                        break;
                case 'c':
                case 'C':
                        cout << "Anda Memilih Menghitung Volume Tabung" << endl;
                        cout << "Masukkan Jari-Jari Tabung : ";
                        cin >> jari; cout << "cm" << endl;
                        cout << "Masukkan Tinggi Tabung : ";
                        cin >> tinggi_tabung; cout << "cm" << endl;
                        vol_tabung = 3.14*jari*jari*tinggi_tabung;
                        cout << "Maka Volume Tabung : " << vol_tabung << "cm3" << endl;
                        break;
                }
                break;

                default:
                        cout << "Maaf, Tidak Ada Dalam Menu!" << endl;
        }
}
```