MODUL PRAKTIKUM

ALGORITMA DAN PEMROGRAMAN

DENGAN BAHASA PEMROGRAMAN C++

Sekolah Tinggi Manajemen Informatika & Komputer

ATMA LUHUR

Disusun Oleh :

Hamidah

KATA PENGANTAR

Assalamualaikum Wr. Wb

Pertama – tama marilah kita panjatkan Puji dan syukur kehadirat Allah Swt yang senantiasa telah memberikan nikmat yang tak ternilai harganya yaitu nikmat kesehatan kepada kita semuanya dan akhirnya selesai juga penyusunan diktat yang berjudul Algoritma dan Pemrograman ini.

Diktat ini berisi materi untuk matakuliah Algoritma dan Pemrograman. Adapun materi yang dibahas menggunakan antara lain :

1. Pengantar Algoritma, Variabel, Konstanta, Tipe Data
2. Perulangan, Pencabangan, Array
3. Struktur

Dengan adanya diktat ini, diharapkan kepada mahasiswa/i agar aktif baik dalam mengembangkan diri maupun bertanya jika ada materi yang disampaikan belum jelas. Ingatlah selalu peribahasa "*malu bertanya sesat di jalan*". Dengan adanya diktat ini, diharapkan pada kemudian hari lebih memacu saya untuk membuat karya yang lebih baik lagi. Amin..

Akhir kata, Wassallamualaikum Wr. Wb

Pangkalpinang, September 2015

Penyusun

Hamidah, M. Kom

DAFTAR ISI

I. PENDAHULUAN

1.1 Definisi Algoritma

Algoritma adalah urutan aksi-aksi yang dinyatakan dengan jelas dan tidak rancu untuk memecahkan suatu masalah dalam rentang waktu tertentu. Setiap aksi harus dapat dikerjakan dan mempunyai efek tertentu.

Algoritma dapat dituliskan dengan banyak cara, mulai dari menggunakan bahasa alami yang digunakan sehari-hari, simbol grafik bagan alir, sampai menggunakan bahasa pemograman seperti bahasa C atau C++.

1.2 Bahasa Pemrograman C dan C++

Cikal bakal dari Bahasa Pemrograman C++ adalah bahasa C yang dikembangkan mulai awal tahun 1980 oleh Bjarne Stroustrup dari AT&T Bell Laboratories. Bahasa Pemrograman C++ merupakan pengembangan dari bahasa C dan pengembangannya diresmikan pada tahun 1985.

Tahun 1989, dunia pemrograman C mengalami peristiwa penting dengan dikeluarkannya standar bahasa C oleh American National Standars Institute (ANSI), yang kemudian dikenal dengan nama ANSI C.

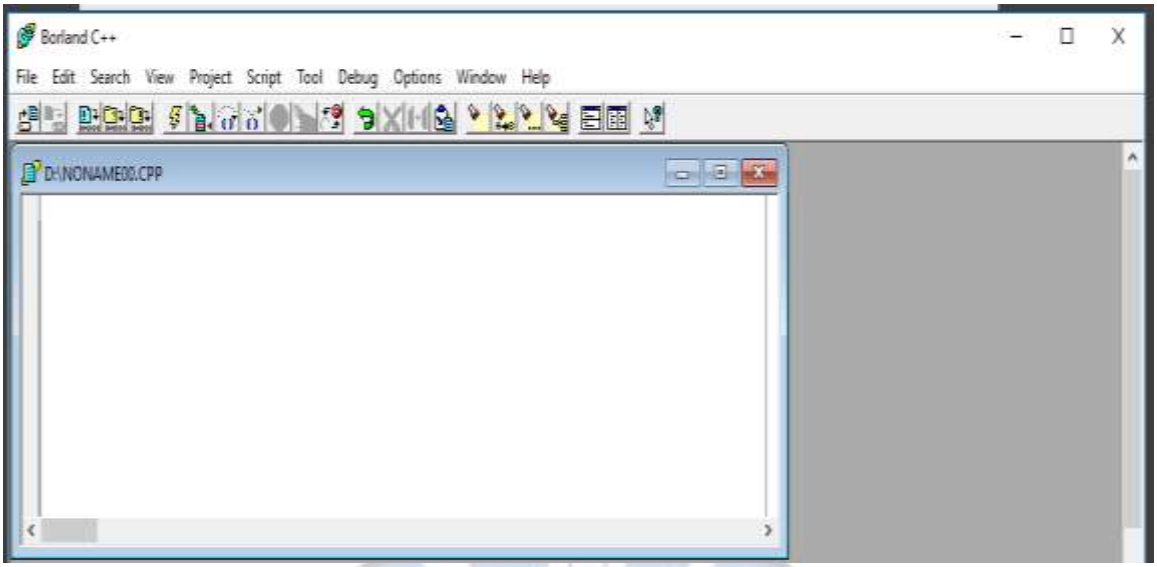
Sebenarnya bahasa C++ mengalami dua tahap evolusi. Bahasa C++ yang pertama, dirilis oleh AT&T Laboratories dan dinamai dengan cfront. Bahasa C++ versi pertama ini hanya berupa kompiler yang menterjemahkan C++ menjadi bahasa C.

Pada evolusi selanjutnya, Borland International Inc. mengembangkan kompiler C++ menjadi sebuah kompiler yang mampu mengubah C++ langsung menjadi bahasa mesin (assembly). Sejak evolusi ini, mulai tahun 1990 C++ menjadi bahasa berorientasi obyek yang digunakan oleh sebagian besar pemrogram professional (*sumber : wikipedia.org*).

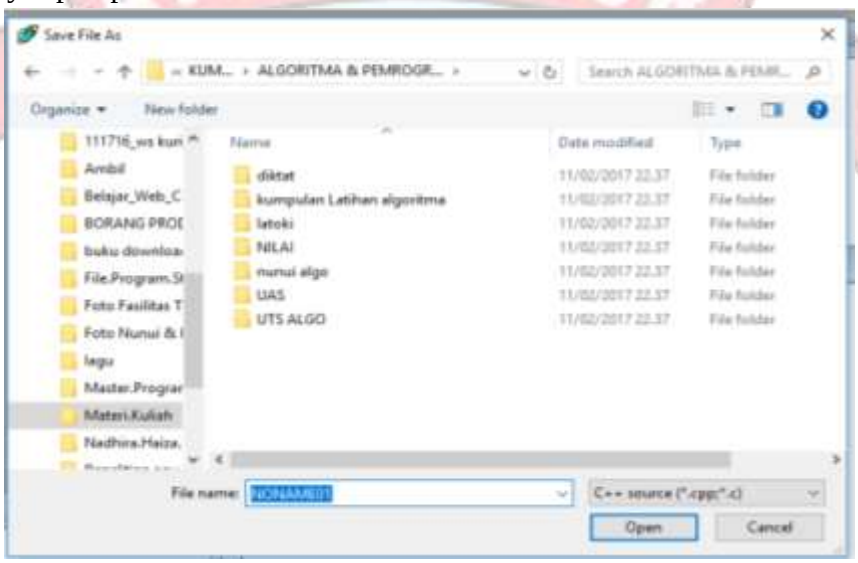
1.3 Langkah-langkah menuliskan program dalam Borland C++

Langkah-langkahnya :

1. Bukalah software Borland C++, akan terlihat tampilan awal Borland C++ sebagai berikut :



2. Tulis source code program bahasa C++. Source code C++ dapat ditulis pada text edit Borland C++.
3. Kompile file dengan (CTRL+ F9 atau pilih submenu Run pada menu Debug) Kompile file dijalankan Untuk mengubah source code menjadi sebuah program, kita gunakan compiler. Setelah source code tercompile, terbentuklah sebuah file objek dengan ekstension " .obj ". File " .obj " ini belum merupakan sebuah program executable.
4. Jalankan Program dengan (CTRL+F9 atau pilih submenu Run pada menu Debug) Setelah kita kompile file yang berisi source code, maka sebagai hasil kompilasi tersebut kita akan mendapatkan suatu file yang bisa dijalankan (executable file). Menjalankan program yang kita buat berarti menjalankan file hasil proses kompilasi tersebut.
5. Untuk menyimpan pilih menu Save As



Jika ada kesalahan, maka program akan langsung menunjuk pada baris manakesalahan tersebut. perbaiki dan jalankan

II. PENGENALAN C++

Setiap program C++ mempunyai bentuk umum seperti di bawah, yaitu :

```
# prepocessor directive
void main() {
// Batang Tubuh Program Utama
 }
```

Penjelasan :

1. **Include**

   Adalah salah satu pengarah prepocessor directive yang tersedia pada C++. Preprocessor selalu dijalankan terlebih dahulu pada saat proses kompilasi terjadi.

   Bentuk umumnya :

   ```
   # include <nama_file>
   ```

   tidak diakhiri dengan tanda semicolon, karena bentuk tersebut bukanlah suatu bentuk pernyataan, tetapi merupakan prepocessor directive. Baris tersebut menginstrusikan kepada kompiler yang menyisipkan file lain dalam hal ini file yang berakhiran .h(file header) yaitu file yang berisi sebagai deklarasi

   contohnya :

   - # include <iostream.h> : diperlukan pada program yang melibatkan objek cout
   - # include <conio.h> : diperlukan bila melibatkan clrscr(), yaitu perintah untuk membersihkan layar.

2. **Fungsi main ()**

   Fungsi ini menjadi awal dan akhir eksekusi program C++. main adalah nama judul fungsi. Melihat bentuk seperti itu dapat kita ambil kesimpulan bahwa batang tubuh program utama berada didalam fungsi main( ). Berarti dalam setiap pembuatan program utama, maka dapat dipastikan seorang pemrogram menggunakan minimal sebuah fungsi. Pembahasan lebih lanjut mengenai fungsi akan diterangkan kemudian. Yang sekarang coba ditekankan adalah kita menuliskan program utama kita didalam sebuah fungsi main().

3. **Komentar**

   Komentar tidak pernah dicompile oleh compiler. Dalam C++ terdapat 2 jenis komentar, yaitu:

   Jenis 1 : /* Komentar anda diletakkan di dalam ini Bisa mengapit lebih dari satu baris */

   Jenis 2 : // Komentar anda diletakkan disini (hanya bisa perbaris)

4. **Tanda Semicolon**

   Tanda semicolon "**;**" digunakan untuk mengakhiri sebuah pernyataan. Setiap pernyataan harus diakhiri dengan sebuah tanda semicolon.

5. **Mengenal cout** (dibaca : C out)

   Pernyataan cout merupakan sebuah objek di dalam C++, yang digunakan untuk mengarahkan data ke dalam standar output (cetak pada layar)

Ketikkan script program berikut :

Nama Program : contoh1.cpp

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

void main()
{
  cout << "Selamat Belajar C++ !!!";
  getch();
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :

```
Selamat Belajar C++!!!
```

Tanda " << " merupakan sebuah operator yang disebut operator "penyisipan/peletakan"

Latihan :

1. Buat program dengan output sebagai berikut :

```
STMIK ATMA LUHUR PANGKALPINANG_
```

2. Buat program dengan output sebagai berikut :

```
STMIK
ATMA LUHUR
PANGKALPINANG
```

III. Tipe Data, Variabel dan Konstanta

3.1 Tipe Data

Tipe data adalah pengelompokkan data berdasarkan isi dan sifatnya. Didalam pemrograman, sifat dari data adalah karakter, bilangan bulat, bilangan pecahan atau boolean. Tipe Data biasanya selalu berpasangan dengan variabel.

Berikut Tabel Tipe Data Dasar (Basic Data Type) yang digunakan dalam Bahasa C++.

3.1.1 Tipe Dasar

Tipe dasar adalah tipe yang dapat langsung dipakai

Tipe Dasar	Ukuran Memori (byte)	Jangkauan Nilai	Jumlah Digit Presisi
Char	1	-128 hingga +127	-
Int	2	-32768 hingga +32767	-
Long	4	-2.147.438.648 hingga 2.147.438.647	-
Float	4	3,4E-38 hingga 3,4E38	6-7
Double	8	1.7E-308 hingga 1.7E308	15-16
long double	10	3.4E-4932 hingga 1.1E4932	19

Catatan :

Untuk mengetahui ukuran memori dari suatu tipe digunakan fungsi sizeof(tipe)

Tipe data dapat diubah (type cast), misalkan :

float x = 3.345;

int p = int(x);

maka nilai p adalah 3 (terjadi truncating).

Tipe data yang berhubungan dengan bilangan bulat adalah char, int, long. Sedangkan lainnya berhubungan dengan bilangan pecahan.

Contoh :

Ketikkan script program berikut :

```
#include <iostream.h>
void main()
{
   int n;
   cout<<n<<endl;    // n sebagai variabel

}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Output :

```
18125
```

Angka 18125 diperoleh ?

Jika variable tidak diinisialisai, namun nilai keluarannya diminta, maka compiler dengan bijak akan menampilkan nilai acak yang nilainya tergantung dari jenis compilernya.

3.1.2 Karakter dan String Literal

String adalah gabungan dari karakter

Contoh :

" Belajar " Literal String

" B " Karakter Panjang String

strlen() nama fungsi untuk menghitung panjang string. Fungsi strlen() dideklarasikan dalam file string.h. Jadi bila anda ingin menggunakan fungsi strlen(), maka prepocessor directive #include<string.h> harus dimasukkan dalam program diatas main().

Contoh :

Ketikkan script program berikut :

```
#include <iostream.h>
#include <conio.h>
#include <string.h>
void main()
{
cout <<strlen("Selamat Datang.\n")<<endl;
cout <<strlen("Selamat Datang.")<<endl;
cout <<strlen("Selamat")<<endl;
cout <<strlen("S")<<endl;
cout <<strlen("")<<endl;
getch();
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :

```
16
15
7
1
0
```

Perhatikan, bahwa disetiap akhir baris pernyataan diakhiri dengan tanda titik – koma (semicolon) “ ; “. Perhatikan, bahwa : • ‘ \n ‘ dihitung satu karakter. \n disebut newline karakter • Endl juga merupakan newline karakter (sama kegunaannya seperti \n). Dalam C++, selain \n terdapat juga beberapa karakter khusus yang biasa disebut escape sequence characters, yaitu :

Karakter	Keterangan
\0	Karakteeer ber-ASCII no1 (karakter null)
\a	Karakter bell
\b	Karakter backspace
\f	Karakter ganti halaman (formfeed)
\n	Karakter baris baru (newline)
\r	Karakter carriage return (ke awal baris)
\t	Karakter tab horizontal
\v	Karakter tab vertika
\\	Karakter \
\'	Karakter ‘
\"	Karakter “
\?	Karakter ?
\ooo	Karakter yang nilai oktalnya adalah ooo (3 digit octal)
\xhh	Karakter yang nilai heksadesimalnya adalah hh (2 digit heksadesimal)

3.1.3 Keyword dan Identifier

Dalam bahasa pemrograman, suatu program dibuat dari elemen-elemen sintaks individual yang disebut token, yang memuat nama variable, konstanta, keyword, operator dan tanda baca.

Ketikkan script program berikut :

```
# include <iostream.h>
void main()
{
  int n=66;
  cout<<n<<endl;    // n sebagai variabel
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :

```
66
```

Program diatas memperlihatkan 15 token, yaitu

main, (,), {, int, n, =, 66, ;, cout, <<, endl, return, 0 dan }

Token n adalah suatu variable

Token 66,0 adalah suatu konstanta

Token int, return dan endl adalah suatu keyword

Token = dan << adalah operator Token(,), {, ;, dan } adalah tanda baca

Baris pertama berisi suatu prepocessor directive yang bukan bagian sebenarnya dari program

3.2 Variabel

Secara umum variabel adalah suatu simbol atau lambang yang mempunyai nilai. Secara teknis dalam pemrograman yang dimaksud dengan variabel adalah area atau tempat didalam memory komputer yang isinya dapat diubah-ubah yang digunakan untuk menampung suatu nilai dan/atau mengambil nilai tersebut. Variabel didefinisikan/dideklarasikan menggunakan kombinasi antara identifier, type dan dapat juga sekaligus diinisialisasi (pemberian nilai awal). Nama dari variabel ditentukan atau dikarang sendiri oleh pembuat program.

```
Variabel = ekspresi ;
```

Ada beberapa aturan dalam pemberian nama variabel yaitu :

- Tidak boleh sama dengan nama atau kata yang sudah disiapkan oleh komputer (reserved word) seperti keyword dan functions.
- Maksimum 32 karakter, bila lebih dari 32 karakter, maka karakter selebihnya tidak diperhatikan oleh komputer.
- Huruf besar (kapital) dan huruf kecil berbeda (case sensitive)
- Karakter pertama harus huruf atau karakter garis bawah (under score), dan karakter berikutnya boleh huruf atau angka, atau karakter garis bawah.
- Tidak boleh mengandung spasi atau blank.

Contoh penamaan variabel yang benar :

```
int angka;
int A ;
int GAJI, TOTAL ;
float  jari_jari;
```

Contoh penamaan variable yang salah :

%nilai_mahasiswa, 80mahasiswa, rata-rata, ada spasi, penting!, dsb

Deklarasi variabel

Deklarasi diperlukan bila kita akan menggunakan pengenal (identifier) dalam program. Identifier dapat berupa variabel, konstanta dan fungsi.

Bentuk umumnya :

```
Tipe Data Nama Variabel ;
```

Contoh :

int x; // Deklarasi x bertipe integer

char y, huruf, nim[10]; // Deklarasi variable bertipe char

float nilai; // Deklarasi variable bertipe float

double beta; // Deklarasi variable bertipe double

int array[5][4]; // Deklarasi array bertipe integer

Adapun contoh programnya sebagai berikut :

Ketikkan script program berikut :

Nama Program : variabel.cpp

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

void main()
{
  int  bulat = 10;
  float pecah = 2.5;

  cout << "Isi Bilangan Bulat    : " << bulat ;
  cout << "\nIsi Bilangan Pecahan : " << pecah ;
  getch();
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :

```
Isi Bilangan Bulat : 10
Isi Bilangan Pecahan : 2.5
```

Penjelasan Program :

- int bulat, float pecah : adalah proses pendeklarasian variabel. Variabel yang dibuat sebanyak 2 buah yaitu variabel dengan nama bulat dan variabel dengan nama pecah. Variabel bulat digunakan untuk menampung bilangan bulat sedangkan variabel pecah digunakan untuk menampung bilangan pecahan.
- bulat = 10, pecah = 2.5 : adalah perintah penugasan (*Assignment Statement*) mengisi nilai/angka kedalam masing-masing variabel.

- \n : adalah perintah ganti baris.

Berikut contoh program menggunakan variabel dengan angka yang akan diisi ke dalam variabel tersebut diinput terlebih dahulu :

Ketikkan script program berikut :

Nama Program : variabelinput.cpp

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

void main()
{
  char kata[5]="SAYA";
  int  bulat = 10;
  float pecah = 2.5;
  cout << "Isi variabel kata : " << kata;
  cout << "\nIsi variabel bulat : " << bulat ;
  cout << "\nIsi variabel pecah : " << pecah ;
  getch();
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

3.3 Konstanta

Pada dasarnya konstanta memiliki kesamaan dengan variabel yaitu sebuah tempat untuk menyimpan sebuah nilai sesuai dengan datanya, hanya saja nilai yang ada dalam konstanta tidak dapat diubah-ubah dan hanya dapat digunakan atau diakses. Konstanta biasanya digunakan untuk menyimpan sebuah nilai yang sering digunakan atau nilai yang sudah pasti. Contohnya jari-jari lingkaran. Dalam bahasa C++ konstanta dapat menggunakan preprocessor directive #define dan const.

a. Menggunakan keyword const

   Contoh : const float PI = 3.14152965;

Berbeda dengan variable, konstanta bernama tidak dapat diubah jika telah diinisialisasi

b. Menggunakan #define

Contoh : #define PI 3.14152965

Keuntungan menggunakan #define apabila dibandingkan dengan const adalah kecepatan kompilasi, karena sebelum kompilasi dilaksanakan, kompiler pertama kali mencari symbol #define (oleh sebab itu mengapa # dikatakan prepocessor directive) dan mengganti semua Phi dengan nilai 3.14152965.

Bentuk Umum Pendeklarasian Konstanta adalah :

```
#define  nama_konstanta  nilai_konstanta
```

Contoh programnya sebagai berikut :

Ketikkan script program berikut :

Nama Program : konstanta.cpp

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

#define phi 3.14

void main()
{
   int jari=10;

   cout << "Keliling Lingkaran dengan jari-jari "<<jari;
   cout << " adalah : "<<(2*phi*jari);
   getch();
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :

```
Keliling Lingkaran dengan jari-jari 10 adalah 62.8
```

Latihan :

1. Buatlah program dengan menggunakan define untuk menghitung volume Tabung (Rumus Volume Tabung : phi x jari-jari x jari-jari x tinggi) dan Luas Tabung (Rumus Luas tabung : 2 x phi x jari-jari x tinggi) dimana jari-jari 7 dan tinggi 24.
2. Buatlah program untuk mencatat data mahasiswa yang terdiri dari field nama, nim dan nilai.

IV. Input dan Operator

4.1 Perintah Input

Setiap bahasa pemrograman tidak akan bisa digunakan secara fleksibel jika tidak memiliki perintah input. Perintah input adalah sebuah perintah dalam bahasa program yang mampu meneruskan nilai dari operator untuk diproses oleh komputer. Perintah input memerlukan perangkat keras input, biasanya adalah keyboard.

Dalam Bahasa C++, perintah cin digunakan untuk menginput suatu nilai dari suatu piranti atau alat masukan (keyboard) untuk selanjutnya di proses oleh program. Sintaknya adalah sebagai berikut :

cin >> variable;

Contoh :

Ketikkan script program berikut :

```
# include <iostream.h>
void main()
{
  char nama[100];    // Deklarasi variable nama
  cout<<"Masukkan nama Anda : ";
  cin>>nama;   // Meminta user untuk menginisialisasi variable nama
  cout<<"Nama anda adalah "<<nama;
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :

```
Masukkan nama Anda : Hamidah
Nama Anda adalah : Hamidah
```

4.2 Operator

Operator adalah symbol yang biasa dilibatkan dalam program untuk melakukan sesuatu operasi atau manipulasi.

Operator yang digunakan didalam pemrograman antara lain :

a. Operator Aritmatika

Beberapa operator aritmatika yang dapat digunakan dalam pemrograman adalah :

1) Operator Perkalian

   Simbol operator perkalian adalah tanda bintang/asterik (*****). Contoh : 5*2

2) Operator Penjumlahan

   Simbol operator penjumlahan adalah tanda tambah/plus (+). Contoh : 5+2

3) Operator Pengurangan

   Simbol operator pengurangan adalah tanda kurang/minus (-). Contoh : 5-2

4) Operator Pembagian

   Simbol operator pembagian adalah tanda garis miring (/). Contoh : 5/2

5) Operator Sisa Hasil Bagi

   Simbol Operator sisa hasil bagi adalah tanda persen (**%**). Contoh : 5%2

Contoh Programnya sebagai berikut :

Ketikkan script program berikut :

Nama Program : aritmatika.cpp

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

void main()
{
  int kali, jumlah, kurang, modulus;
  float bagi;
  kali = 5*2;
  bagi = 5/2;
  jumlah = 5+2;
  kurang = 5-2;
  modulus = 5%2;
  cout <<"\n Angka 1 = 5";
  cout <<"\n Angka 2 = 2";
  cout <<"\nHasil Perkalian Angka 1 dan Angka 2 adalah : "<<kali;
  cout <<"\nHasil Pembagian Angka 1 dan Angka 2 adalah : "<<bagi;
  cout <<"\nHasil Penjumlahan Angka 1 dan Angka 2 adalah : "<<jumlah;
  cout <<"\nHasil Pengurangan Angka 1 dan Angka 2 adalah : "<<kurang;
  cout <<"\nHasil Sisa Pembagian Angka 1 dan Angka 2 adalah : "<<modulus;
  getch();
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :

```
Hasil Perkalian Angka 1 dan Angka 2 adalah : 10
Hasil Pembagian Angka 1 dan Angka 2 adalah : 2.5
Hasil Penjumlahan Angka 1 dan Angka 2 adalah : 7
Hasil Pengurangan Angka 1 dan Angka 2 adalah : 3
Hasil Sisa Pembagian Angka 1 dan Angka 2 adalah : 1
```

b. Operator Relasi

Yaitu operator yang biasa digunakan untuk membandingkan dua buah nilai. Hasil dari operator ini adalah Benar (TRUE) atau Salah (FALSE).

Operator-operator relasi yang dapat digunakan dalam pemrograman sebagai berikut :

- Operator sama dengan

  Simbol operator yang digunakan adalah " == ", operator ini menyatakan bahwa nilai yang dibandingkan antara dua operan adalah sama.

  Contoh : a == b
- Operator tidak sama dengan

  Simbol operator yang digunakan adalah " !=", operator ini menyatakan bahwa nilai yang dibandingkan antara dua operan tidak sama.

  Contoh : a != b
- Operator lebih dari

  Simbol operator yang digunakan adalah " > ", operator ini menyatakan bahwa nilai operan pertama lebih besar dari nilai operan yang kedua.

  Contoh : a > b
- Operator kurang dari

  Simbol operator yang digunakan adalah " < ", operator ini menyatakan bahwa nilai operan pertama lebih kecil dari nilai operan yang kedua.

  Contoh : a < b
- Operator lebih dari sama dengan

Simbol operator yang digunakan adalah " >= ", operator ini menyatakan bahwa nilai operan pertama lebih besar dari atau sama dengan nilai operan yang kedua. Contoh : a >= b

- Operator kurang dari sama dengan

  Simbol operator yang digunakan adalah " <= ", operator ini menyatakan bahwa nilai operan pertama lebih kecil dari atau sama dengan nilai operan yang kedua. Contoh : a <= b

Contoh programnya sebagai berikut :

Ketikkan script program berikut :

Nama Program : relasi.cpp

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

void main()
{

  int a, b;

  cout <<"Input a : ";
  cin >> a;
  cout <<"Input b : ";
  cin >> b ;
  cout << "\nMaka : \n";

  if(a!=b)
  {
    cout << "\nNilai a tidak sama dengan nilai b";
  }
  else
  {
    cout << "\nNilai a sama dengan nilai b";
  }

  if(a<b)
  {
    cout << "\nNilai a kurang dari nilai b";
  }
  else
  {
    cout << "\nNilai a lebih dari nilai b";
  }
  getch();
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :

Tergantung input user

Misalkan :

Input a : 5
Input b : 10
Maka
Nilai a kurang dari nilai b

c. Operator Logika Boolean

Yaitu operator yang biasa digunakan untuk mengaitkan dua buah ungkapan kondisi menjadi sebuah kondisi. Operator-operator logika Boolean yang biasa digunakan dalam pemrograman adalah :

- **Operator dan (AND)**

  Simbol operator yang digunakan adalah " && ".

Ungkapan Kondisi 1	Ungkapan Kondisi 2	Hasil Operator Logika dan
B	B	B
B	S	S
S	B	S
S	S	S

  B = Benar, S = Salah.

  Dari table diatas dapat disimpulkan bahwa hasil operator akan BENAR jika kedua ungkapan kondisi tersebut, kedua-duanya Benar.

- **Operator atau (OR)**

  Simbol operator yang digunakan adalah " || ".

Ungkapan Kondisi 1	Ungkapan Kondisi 2	Hasil Operator Logika dan
B	B	B
B	S	B
S	B	B
S	S	S

  B = Benar, S = Salah

  Dari tabel diatas dapat disimpulkan bahwa hasil operator akan BENAR jika kedua atau salah satu ungkapan kondisi tersebut Benar.

- **Operator bukan / negasi (NOT)**

  Simbol operator yang digunakan adalah "!". Operator ini berfungsi pembalik nilai logika TRUE menjadi FALSE atau nilai logika FALSE menjadi TRUE.

Ungkapan Kondisi 1	Hasil Operator Logika bukan
B	S
S	B

  Misalkan ada dua bilangan, x = 5 dan y = 6. Jika dibentuk menjadi satu kondisi sehingga menjadi (x==y), maka kondisi tersebut bernilai TRUE. Bila dikenai dengan operator bukan, kondisi tersebut menjadi !(x==y), nilai kondisi sebelumnya bernilai TRUE berubah menjadi FALSE.

Contoh programnya sebagai berikut :

Ketikkan script program berikut :

Nama Program : boolean.cpp

```
#include <iostream.h>
#include <conio.h>
void main()
{
    int a, b, c;

    cout << "Input a : "; cin >> a;
    cout << "Input b : "; cin >> b;
    cout << "Input c : "; cin >> c;
    cout << "\nMaka : \n";

    if ((a != b) && (b < c))
    {
      cout << "\nNilai a tidak sama dengan b DAN nilai b lebih kecil dari c";
    }
    else
    {
      cout << "\nNilai a sama dengan b DAN nilai b lebih besar dari c";
    }

    if ((a == b) || (b < c))
    {
      cout << "\nSalah Satu Ungkapan Kondisi bernilai BENAR";
    }
    else
    {
      cout << "\nKedua Ungkapan Kondisi bernilai SALAH";
    }
    getch();
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

- **Operator bukan / negasi (NOT)**

  Operator kondisi digunakan untuk memperoleh nilai dari dua kemungkinan ungkapan1 ? ungkapan2 : ungkapan3 Bila nilai ungkapan1 benar, maka

nilainya sama dengan ungkapan2, bila tidak maka nilainya sama dengan ungkapan3

Contoh :

Ketikkan script program berikut :

```
#include <iostream.h>
void main()
{
   int m = 26, n = 82;
   int min = m < n ? m : n;
   cout<<"Bilangan terkecil adalah "<<min<<endl;
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :

```
Bilangan terkecil adalah : 26
```

- **Operator Naik Dan Turun (Increment dan Decrement)**

Operator increment ++

Operator decrement --

Contoh :

Ketikkan script program berikut :

```
#include <iostream.h>
void main()
{
   int m = 44, n = 66;

   cout<<"m = "<<m<<", n = "<<n<<endl;
   ++m; --n;
   cout<<"m = "<<m<<", n = "<<n<<endl;
   m++; n--;
   cout<<"m = "<<m<<", n = "<<n<<endl;
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :

```
C:\USERS\DELL\DOCUMENTS\NONAME01.exe
m = 44, n= 66
m = 45, n= 65
m = 46, n= 64
```

Latihan

1. Buatlah program untuk menghitung luas bangunan geometri (bujursangkar, lingkaran, segitiga dan trapesium). Data masukkan dibaca dari piranti masukkan dan luas bangun ditampilkan sebagai keluaran.
2. Buatlah program untuk menghitung harga total suatu barang, dimana jumlah barangnya 5, harga perunit 5203.02.
3. Buatlah program untuk penggunaan operasi aritmatika yaitu penjumlahan, pembagian, perkalian, dan pengurangan dengan variabel yang diinputkan.
4. Mencetak sejumlah deret bilangan ganjil antara 1 sampai N, dimana N dimasukkan oleh user.

V. Pencabangan atau Seleksi

Salah satu struktur alur algoritma adalah **pencabangan**. Pencabangan bermanfaat untuk **menentukan satu dari sekian banyak pilihan yang telah disediakan**. Didalam pemrograman, pencabangan akan mengakibatkan ada beberapa baris perintah yang tidak akan dikerjakan/dilewati sama sekali, karena pencabangan membentuk kelompok/blok tersendiri yang akan dikerjakan atau tidak ditentukan berdasarkan **kondisinya**.

Dalam pencabangan, dikenal dengan istilah kondisi (*condition*). Kondisi ini yang menentukan blok/kelompok dari baris perintah dikerjakan atau tidak sama sekali. Kondisi bertugas untuk membandingkan dua atau lebih operan. Operator yang digunakan untuk membandingkan operan tersebut tentu saja menggunakan operator relasi. Hasil akhir dari kondisi adalah nilai BENAR (**TRUE**) atau SALAH (**FALSE**). Contohnya sebagai berikut :

NO	KONDISI	NILAI
1	5 > 2	TRUE
2	5 == 2	FALSE
3	5 <= 7	TRUE
4	5 != 2	TRUE

Pencabangan dibedakan menjadi dua macam, yaitu ***simple conditional branch*** **dan** ***multiway conditional branch***. Berikut adalah penjelasan dari kedua macam bentuk pencabangan tersebut.

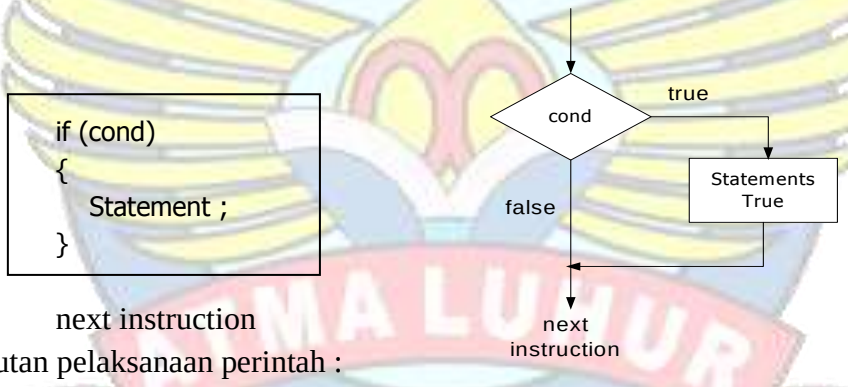
5.1 Simple Conditional Branch

Control statement yang menjadi bagian simple conditional branch adalah control statement if. Control statement if terbagi menjadi 3 macam sesuai dengan keperluannya, yaitu IF-THEN, IF-THEN-ELSE dan NESTED IF.

a) IF - THEN

Ciri dari bentuk pencabangan ini hanya memiliki satu buah grup/kelompok baris perintah yang akan dikerjakan apabila kondisinya BENAR/TRUE. IF-THEN digunakan apabila hanya mempunyai satu pilihan/pilihan tunggal saja.

Bentuk Umum Instruksi IF – THEN dan Flowchart adalah sebagai berikut :



Urutan pelaksanaan perintah :

- Periksa nilai kondisi pada pencabangan (**cond = kondisi**), apakah bernilai **TRUE** atau **FALSE** .
- Bila kondisi bernilai **TRUE** maka **blok perintah (grup/kelompok baris perintah) / statements true** akan dikerjakan, setelah selesai maka langsung mengerjakan **next instruction**.
- Bila kondisi bernilai **FALSE**, maka langsung meloncat mengerjakan **next instruction**.

Berikut contoh program menggunakan IF-THEN :

Ketikkan script program berikut :

Nama Program : ifthen.cpp

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

void main()
{
  int angka;

  cout << "input sebuah angka : ";
  cin >> angka;
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Output Program :

- Kondisi bernilai **BENAR/TRUE**.

```
input sebuah angka : 6
Angka yang diinput > 5
Selesai !!!
```

- Kondisi bernilai **SALAH/FALSE**.

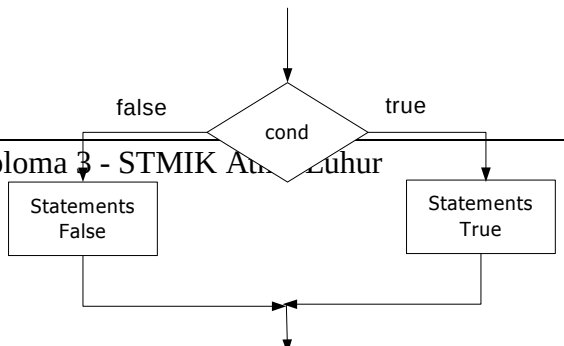
```
input sebuah angka : 4

Selesai !!!
```

b) IF - THEN - ELSE

Ciri dari bentuk pencabangan ini memiliki dua buah grup/kelompok baris perintah. Masing-masing grup mewakili hasil akhir dari kondisinya. Grup/kelompok baris perintah yang pertama akan dikerjakan apabila kondisi bernilai **BENAR/TRUE**, sedangkan grup/kelompok baris perintah yang kedua akan dikerjakan apabila kondisi bernilai **SALAH/FALSE**. IF-THEN-ELSE digunakan apabila mempunyai dua pilihan.

Bentuk Umum Instruksi IF – THEN – ELSE dan Flowchart sebagai berikut :

```
if (cond)
{
   statement true ;
```



```
}
else
{
    statement false ;
}
next instruction
```

Urutan pelaksanaan perintah :

- Periksa nilai kondisi pada pencabangan (**cond = kondisi**), apakah bernilai **TRUE** atau **FALSE** .
- Bila kondisi bernilai **TRUE** maka **blok perintah (grup/kelompok baris perintah) / statements true** akan dikerjakan, setelah selesai maka langsung mengerjakan **next instruction**.
- Bila kondisi bernilai FALSE maka **blok perintah (grup/kelompok baris perintah) / statements fals**e akan dikerjakan, setelah selesai maka langsung mengerjakan **next instruction**.

Berikut contoh program menggunakan IF-THEN-ELSE :

Ketikkan script program berikut :

Nama Program : ifthenelse.cpp

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

void main()
{
  int angka;

  cout << "input sebuah angka : ";
  cin >> angka;
  if (angka>5)
  {
    cout << "Angka yang diinput > 5";
  }
  else
  {
    cout << "Angka yang diinput < 5";
  }
  cout << "\nSelesai !!!";
  getch();
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Output Program :

- Kondisi bernilai **BENAR/TRUE**

```
input sebuah angka : 6
Angka yang diinput > 5
Selesai !!!
```

- Kondisi bernilai **SALAH/FALSE**

```
input sebuah angka : 4
Angka yang diinput < 5
Selesai !!!
```

Contoh 2 :

Ketikkan script program berikut :

```
# include <iostream.h>

void main ()
{
   int usia;

   cout << "Berapa usia Anda ? ";
   cin >> usia;

   if (usia < 17)
      cout << "Anda tidak diperkenankan menonton" << endl;
   else
      cout << "Selamat Menonton" << endl;
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :

```
Berapa usia Anda ? 16
Anda tidak diperkenankan menonton
```

Penjelasan :

Terlihat bahwa kalau usia yang dimasukkan lebih dari 17, program akan memberi pesan Selamat Menonton. Selain dari if … else, juga dikenal bentuk if … else if. Adapun perbedaannya diilustrasikan oleh dua contoh dibawah ini.

NESTED IF

Disebut juga dengan if bersarang/pencabangan bersarang. NESTED IF digunakan apabila pilihan lebih dari dua pilihan. Bentuk pencabangan ini memiliki lebih dari dua grup/kelompok baris perintah dan kondisi lebih dari satu. Tidak ada bentuk yang baku dari NESTED IF, karena NESTED IF dibentuk menggunakan kombinasi antara IF-THEN dan IF-THEN-ELSE.

Berikut adalah contoh dari bentuk-bentuk NESTED IF dan flowchartnya :

Berikut contoh program menggunakan NESTED IF :

Ketikkan script program berikut :

Contoh Program : nestedif.cpp

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

void main()
{
  int angka;

  cout << "input sebuah angka : ";
  cin >> angka;
  if (angka>5)
  {
    cout << "Angka yang diinput > 5";
  }
  else if (angka<5)
  {
    cout << "Angka yang diinput < 5";
  }
  else
  {
    cout << "Angka yang diinput sama dengan 5";
  }
  cout << "\nSelesai !!!";
  getch();
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Output Program :

- Pilihan dengan grup/kelompok baris perintah yang pertama

```
input sebuah angka : 6
Angka yang diinput > 5
Selesai !!!
```

- Pilihan dengan grup/kelompok baris perintah yang kedua

```
input sebuah angka : 4
Angka yang diinput < 5
Selesai !!!_
```

- Pilihan dengan grup/kelompok baris perintah yang ketiga

```
input sebuah angka : 5
Angka yang diinput sama dengan 5
Selesai !!!
```

Contoh 2 :

Ketikkan script program berikut :

```
# include <iostream.h>

void main ()
{
   int m =166;
   if (m > 1000)
      cout << m << " Lebih Besar dari 1000 " << endl;
   if (m > 100)
      cout << m << " Lebih Besar dari 100 " << endl;
   if (m >10)
      cout << m << " lebih besar dari 10 " << endl;
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :

```
166 Lebih Besar dari 100
```

Contoh 3 :

Ketikkan script program berikut :

```
# include <iostream.h>

void main ()
{
    int m = 166;
    if (m > 1000)
       cout << m << " lebih besar dari 1000 " << endl;
    else if (m > 100)
       cout << m << " Lebih besar dari 100 " << endl;
    else if (m >10)
       cout << m << " Lebih besar dari 10 " << endl ;
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :

```
166 Lebih Besar dari 100
```

Contoh 4 :

Ketikkan script program berikut :

```
#include <iostream.h>
void main()
{
   int m = 166;
   if(m > 1000)
      cout<<m<<" lebih besar dari 1000\n";
   else
   {
     if(m > 100)
         cout<<m<<" lebih besar dari 100\n";
     else if(m > 10)
         cout<<m<<" lebih besar dari 10\n";
   }
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :

```
166 Lebih Besar dari 100
```

Latihan :

1. Buatlah program untuk mencari apakah bilangan tersebut ganjil atau genap, dimana bilangan merupakan piranti masukkan

2. Buatlah program untuk menseleksi suatu bilangan dengan ketentuan sebagai berikut : 0<=nilai <30 : Nilai rendah 30<=nilai < 60 : Nilai sedang 60<=nilai<=100 : Nilai tinggi

3. Buatlah program dalam bentuk menu yang mampu menghitung :

   a. Luas dan Keliling Bujur sangkar b. Luas dan Keliling persegi panjang c. Luas dan keliling lingkaran

VI. PENCABANGAN SWITCH CASE

6.1 Multiway Conditional Branch

Control statement yang menjadi bagian multiway conditional branch adalah control statement switch case. Switch case dapat digunakan untuk satu, dua atau lebih dari dua pilihan. Khusus untuk pilihan dari dua, switch case sama dengan nested if, tetapi ada beberapa perbedaan dari kedua control statement tersebut diantaranya :

a. Pernyataan switch berbeda dengan pernyataan if dimana switch hanya dapat menguji kesamaan, sedangkan pernyataan if dapat melakukan evaluasi sembarang tipe ekspresi Boolean. Dengan demikian switch terlihat seperti hanya mencocokkan diantara nilai-nilai ekspresi dan konstanta-konstanta case.
b. Tidak ada konstanta case di blok case yang sama dapat mempunyai nilai yang identik.
c. Pernyataan switch biasanya lebih efisien dibanding if bersarang yang dalam.

Bentuk Umum Control Statement Switch Case :

```
switch(nama_variabel)
{
```

```
        case nilai_variabel_1 :
                statement_1 ;
            break ;
        case nilai_variabel_2 :
                statement_2 ;
            break ;

        ……..
        case nilai_variabel_n :
                statement_n ;
            break ;

        default   :
                statement_default ;
            break ;
    }
```

Perintah break berfungsi untuk keluar dari case. Jika break ditiadakan, maka akan dilanjutkan dengan pelaksanaan dibawahnya sampai selesai. Perintah break merupakan optional, tergantung kebutuhan program. Adakalanya diperlukan untuk menjalankan case sekaligus, hal ini berarti perintah break tidak digunakan.

Contoh program menggunakan switch case sebagai berikut :

Ketikkan script program berikut :

Nama Program : switchcase.cpp

```
#include <iostream.h>
#include <conio.h>
void main()
{
   int kodehari ;

   cout << "Input Kode Hari [ 1 s/d 7 ] : ";
   cin >> kodehari;

   switch(kodehari)
   {
      case 1 :
               cout << "Hari Senin";
               break ;
      case 2 :
               cout << "Hari Selasa";
               break ;
      case 3 :
               cout << "Hari Rabu";
               break ;
      case 4 :
               cout << "Hari Kamis";
               break ;
      case 5 :
               cout << "Hari Jumat";
               break ;
      case 6 :
               cout << "Hari Sabtu";
               break ;
      case 7 :
               cout << "Hari Minggu";
               break ;
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Output Program :

```
Input Kode Hari [ 1 s/d 7 ] : 2
Hari Selasa
```

Contoh 2 :

Ketikkan script program berikut :

```
// Program untuk melihat nilai akhir test
// Nilai A jika nilai diatas 80, B jika 70 <= nilai < 80
// C jika 50 <= nilai < 70 , D jika 30 <= nilai <50
// E jika nilai < 30

# include <iostream.h>
void main ()
{
   int nilai;
   cout << "Masukkan nilai test : ";
   cin >> nilai;
   switch (nilai/10)
   {
     case 10:
      case 9:
      case 8:
         cout << 'A' <<endl;break;
      case 7:
         cout << 'B' <<endl;break;
      case 6:
      case 5:
         cout << 'C' <<endl;break;
      case 4:
      case 3:
         cout << 'D' <<endl;break;
      case 2:
      case 1:
      case 0:
         cout << 'E' <<endl;break;
      default:
         cout << " Salah, nilai diluar jangkauan. " << endl;
   }
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :

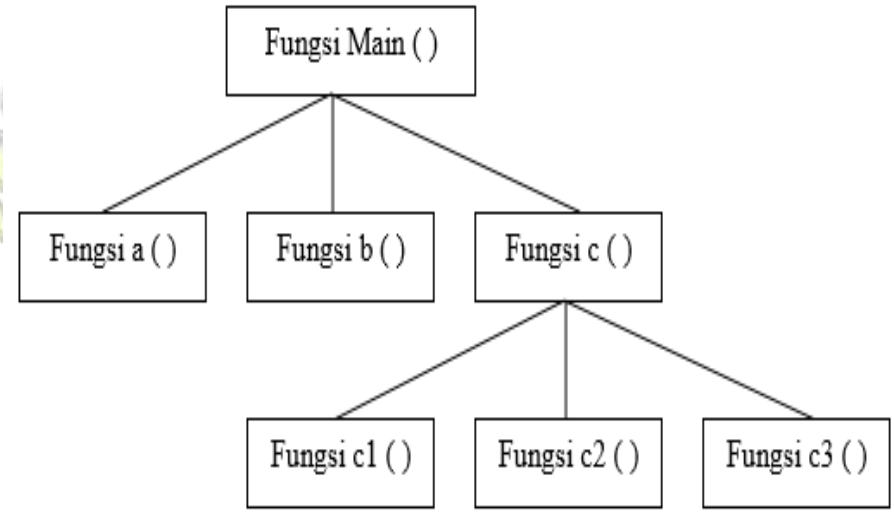
```
Masukkan nilai test : 45
D
Masukkan nilai test : 450
Salah, nilai diluar jangkauan.
```

Latihan :

1. Buatlah program dalam bentuk menu yang mampu menghitung :
   a. Luas dan Keliling Bujur sangkar
   b. Luas dan Keliling persegi panjang
   c. Luas dan keliling lingkaran

7.1 Function/Fungsi

Fungsi adalah sekumpulan perintah operasi program yang dapat menerima argumen input dan dapat memberikan hasil output yang dapat berupa nilai ataupun sebuah hasil operasi. Hasil akhir fungsi akan berupa sebuah nilai balik (return) Nama fungsi yang didefinisikan sendiri oleh pemrogram tidak boleh sama dengan nama build-in function pada compiler C++. Fungsi digunakan agar pemrogram dapat menghindari penulisan bagian program (kode) berulang-ulang, dapat menyusun kode program agar terlihat lebih rapi dan kemudahan dalam debugging program. Parameter adalah nama-nama peubah yang dideklarsikan pada bagian header fungsi. Pemrogram dapat membuat fungsi yang didefinisikan sendiri olehnya.



Bentuk umum :

```
Deklarasi Parameter
{
  Isi fungsi
}
```

7.1.1 Prototipe Fungsi

Sebuah fungsi tidak dapat dipanggil kecuali sudah dideklaraikan, deklarasi fungsi dikenal dengan sebutan prototipe fungsi. Prototipe fungsi berupa :

1. Nama Fungsi
2. Tipe nilai fungsi
3. Jumlah dan tipe argumen

Dan diakhiri dengan titik koma, sebagaimana pada pendeklarasian variabel. Sebagai contoh :

1. long kuadrat (long l) ; Pada contoh pertama, fungsi kuadrat ( ) mempunyai argumen bertipe long dan nilai balik bertipe long.
2. void garis ( ); Pada contoh kedua, fungsi garis ( ) tidakmemiliki argumen dan nilai baliknya tidak ada (void).
3. double maks (double x, double y) Pada contoh ketiga, fungsi maks( ) mempunyai dua buah argumen, dengan masingmasing argumen bertipe double.

Ketikkan script program berikut :

```
# include <iostream.h>

double hasil (int A, int B);

void main()
{
   int x,y;
   double z;

   cout << "Masukkan Nilai x : ";
   cin >> x;

   cout << "Masukkan Nilai y : ";
   cin >> y;

   z = hasil (x,y);

   cout << "Hasil perkaliannya = ";
   cout << x << " x " << y << " = " << z;
}

double hasil (int A, int B)
{
   return (A * B);
}     // Statement Mengembalikan Nilai
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Penjelasan :

// Fungsi Perkalian

// tipe_return nama_fungsi (tipe_argument argumen)

```
double hasil (int A, int B)
{
   return (A * B);
}     // Statement Mengembalikan Nilai
```

Fungsi yang didefinisikan oleh pemrogram terdiri atas dua bagian, yaitu judul (header) dan isi (body). Judul dari sebuah fungsi terdiri dari tipe return (double),

nama fungsi (hasil) dan list parameter (int A, int B). Jadi, judul untuk fungsi hasil adalah double hasil (int A, int B) Isi dari sebuah fungsi adalah blok kode yang mengikuti judulnya. Berisi kode yang menjalankan aksi dari fungsi, termasuk pernyataan return yang memuat nilai fungsi yang akan dikembalikan ke yang memanggilnya, Isi dari fungsi hasil ( ) adalah

{ return A * B; }

Biasanya isi dari fungsi cukup besar. Meskipun demikian, judulnya tetap hanya berada dalam satu baris. Isi dari sebuah fungsi dapat memanggil fungsi itu sendiri (disebut rekursif) atau memanggil fungsi lainnya. Pernyataan return dari sebuah fungsi mempunyai dua manfaat, yaitu akan mengakhiri fungsi dan mengembalikan nilainya ke program pemanggil.

Bentuk umum pernyataan return adalah :

return ekspresi;

Dengan ekspresi adalah sebuah ekspresi yang nilainya dinyatakan untuk sebuah variable yang tipenya sama seperti tipe return. Terdapat juga fungsi yang tidak memberikan nilai balik atau tipe returnnya void.

Ketikkan script program berikut :

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

void tampilkan_judul()  ;
void main()
{
tampilkan_judul();
}
void tampilkan_judul()
{
cout <<"STMIK ATMA LUHUR"<<endl;
cout <<"Manajemen Informatika"<<endl;
cout <<"Jl.Jend. Sudirman Kel.Selindung Kec. Gabek"<<endl;
cout <<"Pangkalpinang"<<endl;

getch();
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

7.1.2 Variabel Lokal dan Variabel Eksternal

Variabel lokal adalah variabel yang didefinisikan dalam suatu fungsi tertentu, sehingga hanya dikenal dalam fungsi tersebut. Dalam hal ini artinya suatu fungsi tidak akan mengenal variabel lokal dan fungsi lain. Suatu fungsi hanya akan mengenal variabel yang didefinisikan dalam fungsi yang bersangkutan. Variabel eksternal adalah variabel yang bersifat global yang dapat dikenali oleh seluruh fungsi yang terdapat dalam program tersebut. Seluruh fungsi akan mengenal variabel yang bersifat eksternal ini. Variabel eksternal dideklarasikan diluar fungsi dan sejajar dengan prototipe fungsi serta pengarah kompiler.

Contoh :

Ketikkan script program berikut :

```
// nama program : clokal_eksternal.cpp
// contoh program variabel lokal dan eksternal

# include <iostream.h>

int data = 100;      // variabel eksternal
void fungsi_satu (); // prototipe fungsi
void fungsi_dua ();
void main ()
{
   int data = 200;  // variabel lokal main
   fungsi_satu();
   fungsi_dua();
   cout << "Nilai data lokal main = " << data << endl;
}

void fungsi_satu()
{
   int data = 300; // variabel lokal fungsi_satu
   cout << "Nilai data lokal satu = " << data << endl;
}

void fungsi_dua()
{
   cout << "Nilai data eksternal = " << data << endl;
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :

```
C:\USERS\DELL\DOCUMENTS\NONAME01.exe
Nilai data lokal satu =300
Nilai data eksternal =100
Nilai data lokal main =200
```

Penjelasan :

Dalam pemograman tersebut terdapat variabel lokal dan variabel eksternal yang namanya sama yaitu data. Dalam fungsi main ( ) dan fungsi_satu ( ) terdapat variabel lokal dengan nama sama tetapi sebetulnya lokasi penyimpanannya dalam memori berbeda, sehingga dua variabel itu berbeda dan tidak saling mengenal.

Fungsi_satu ( ) sebetulnya mengenal variabel eksternal data yang nilainya 100, tetapi karena dalam fungsi tedapat variabel lokal data yang bernilai 300, maka diprioritaskan untuk diproses dalam fungsi tersebut adalah variabel lokalnya. Jika dalam fungsi terdapat variabel lokal dan variabel eksternal yang sama, maka diprioritaskan untuk diproses variabel lokal. Dalam fungsi_dua( ) tidak terdapat variabel lokal sehingga yang diproses pada fungsi tersebut adalah variabel eksternalnya.

7.1.3 Parameter

Parameter adalah sarana komunikasi antar fungsi. Pengertian antar fungsi adalah antara fungsi dengan fungsi lain termasuk antara fungsi dengan fungsi utama. Dalam pemograman yang melibatkan fungsi, diusahakan agar fumgsi bersifat independen artinya tidak tergantung pada fungsi lain. Setiap fungsi hanya mengerjakan satu tugas tertentu. Antar fungsi saling berkomunikasi menggunakan parameter. Terdapat dua macam bentuk parameter dalam hubungannya dengan penggunan fungsi dalam program yaitu : - Parameter Formal : parameter yang diberi nilai. Parameter formal merupakan parameter yang terdapat dalam daftar parameter fungsi. - Parameter Aktual : parameter yang memberi nilai. Parameter fungsi dan digunakan untuk memberi nilai pada parameter formal. Dalam contoh program perkalian di atas parameter formal terdapat pada pendefinisan fungsi :

```
double hasil(int A, int B) //   parameter formal
{
      return (A * B);
}
```

Sedangkan parameter aktual terdapat pada pemanggilan fungsi :

```
void main()
{ ..............
..............
z = hasil(x,y); //   parameter aktual
...........
}
```

a. Cara Melewatkan Parameter

Cara melewatkan suatu parameter dalam Bahasa C++ ada dua cara yaitu :

1) Pemanggilan Secara Nilai (Call by Value)

a) Call by value akan menyalin nilai dari parameter aktual ke parameter formal.

b) Yang dikirimkan ke fungsi adalah nilai dari datanya, bukan alamat memori letak dari datanya.

c) Fungsi yang menerima kiriman nilai akan menyimpannya di alamat terpisah dari nilai aslinya yang digunakan oleh bagian program yang memanggil fungsi.

d) Perubahan nilai di fungsi (parameter formal) tidak akan merubah nilai asli di bagian program yang memanggilnya.

e) Pengiriman parameter secara nilai adalah pengiriman searah, yaitu dari bagian program yang memanggil fungsi ke fungsi yang dipanggil.

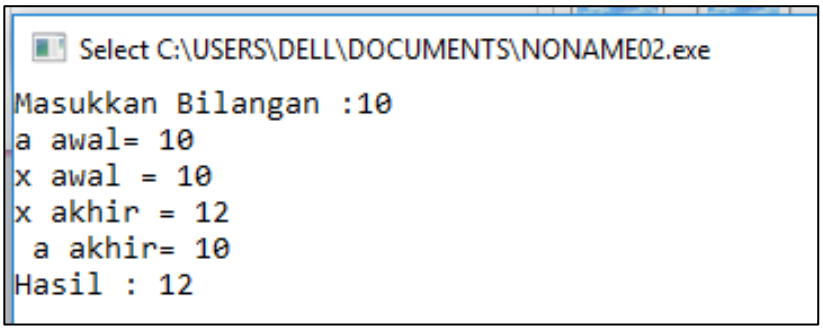
f) Pengiriman suatu nilai dapat dilakukan untuk suatu ungkapan, tidak hanya untuk sebuah variabel, elemen array atau konstanta saja.

Ketikkan script program berikut :

```
#include<iostream.h>
/*COntoh program transfer by value*/

int Tambah(int x);
void main()
{
   int a,hasil;

   cout<<"Masukkan Bilangan: ";
   cin>>a;

   cout<<"a awal= "<<a<<endl;

   hasil = Tambah(a);

   cout<<"a akhir= "<<a<<endl;

   cout<<"Hasil : "<<hasil;
}

int Tambah(int x)
{
   cout<<"x awal = "<<x<<endl;

   x = x + 2;

   cout<<"x akhir = "<<x<<endl;

   return x;
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :



2) Pemanggilan Secara Referensi (Call by Reference)

a) Pemanggilan secara Referensi merupakan upaya untuk melewatkan alamat dari suatu variabel ke dalam fungsi.

b) Yang dikirimkan ke fungsi adalah alamat letak dari nilai datanya, bukan nilai datanya.

c) Fungsi yang menerima kiriman alamat ini maka menggunakan alamat yang sama untuk mendapatkan nilai datanya.

d) Perubahan nilai di fungsi akan merubah nilai asli di bagian program yang memanggil fungsi.

e) Pengiriman parameter secara referensi adalah pengiriman dua arah, yaitu dari fungsi pemanggil ke fungsi yang dipanggil dan juga sebaliknya.

f) Pengiriman secara acuan tidak dapat dilakukan untuk suatu ungkapan.

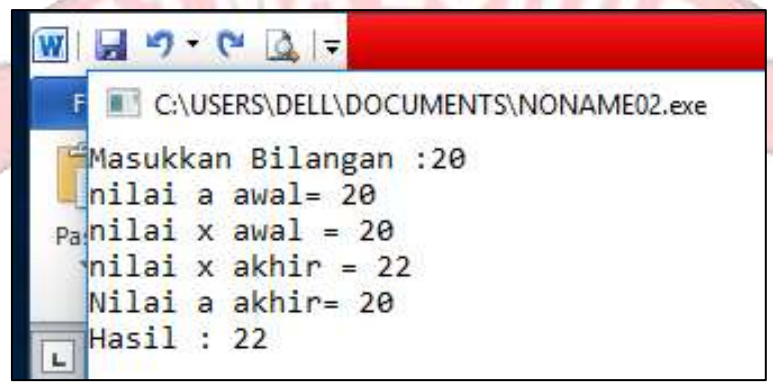
Contoh :

Ketikkan script program berikut :

```
#include<iostream.h>
/*COntoh program transfer by referensi*/

int Tambah(int &x); //x diberi & tuk referensi
                    //sehingga nilai a akan sll mengikuti nilai x
                    //krn var a dan x berada dlm satu alamat memori

void main()
{
   int a,hasil;

   cout<<"Masukkan Bilangan: ";
   cin>>a;

   cout<<"nilai a awal= "<<a<<endl;
   hasil = Tambah(a);
   cout<<"nilai a akhir= "<<a<<endl;
   cout<<"Hasil : "<<hasil;
}

int Tambah(int &x)
{
   cout<<"nilai x awal = "<<x<<endl;
   x = x + 2;
   cout<<"nilai x akhir = "<<x<<endl;
   return x;
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :



7.1.4 Parameter

Salah satu keistimewaan C++ yang sangat bermanfaat dalam pemrograman adalah adanya kemampuan untuk menyetel nilai default Argumen fungsi. Argumen-argumen yang mempunyai nilai bawaan nantinya dapat tidak disertakan di dalam pemanggilan fungsi dan dengan sendirinya C++ akan menggunakan nilai bawaan dari argumen yang tidak disertakan.

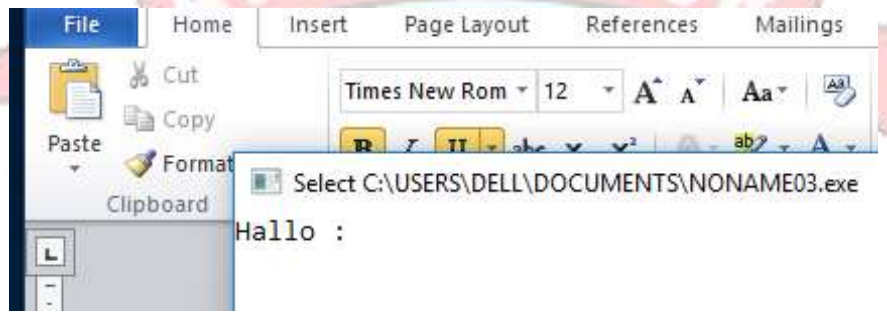
Contoh :

Ketikkan script program berikut :

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

void sayHello(int n);

void main()
{
   sayHello(1);
}

void sayHello(int n)
{

   for(int m=0;m<n;m++)
      cout<<"Halloo :\n";
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :



Penjelasan :

Jika pada program, argumen sayHello tidak diberikan, maka program akan menampilkan : **Halloo :**

Sebanyak satu kali, namun jika argumen pada fungsi sayHello diberikan, misalkan sayHello(4), maka program akan menampilkan 4 kali. Itulah yang disebut dengan nilai default pada fungsi.

7.1.5 Rekursi

Merupakan suatu fungsi dapat memanggil fungsi yang merupakan dirinya sendiri. Penerapan rekursi diantaranya untuk menghitung nilai : xn

Contohnya :

Ketikkan script program berikut :

```
// Operasi pangkat secara rekursi

# include <iostream.h>

long int pangkat (int x, int n);
void main ()
{
   int x, y;

(  cout << " Menghitung x ^ y " << endl;
   cout << " x = ";
   cin >> x;
   cout << " y = ";
   cin >> y;

   cout << x << " ^ " << y << " = " << pangkat(x,y)<< endl;
}

long int pangkat (int x, int n)
{
   if (n==1)
      return (x);
   else
      return ( x* pangkat (x, n-1));
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :

```
C:\USERS\DELL\DOCUMENTS\NONAME03.exe
Menghitung x^y
x =5
y=4
5 ^ 4=625
```

7.1.6 Fungsi-fungsi Bawaan C++

Anda dapat menggunakan fungsi-fungsi bawaan C++, misalkan fungsi-fungsi matematika, pengolah kata dan banyak lagi. Sebenarnya (mungkin tidak terasa bagi anda) main juga adalah fungsi, jadi tanpa anda sadari sebenarnya anda telah menggunakan fungsi. Untuk dapat menggunakan fungsi-fungsi tersebut anda harus meng-include file dimana fungsi tersebut didefinisikan Misalkan :

- Fungsi – fungsi matematika, anda harus meng-include file math.h
- Fungsi – fungsi pengolah string dan karakter, anda harus meng-include file string.h
- Fungsi clrscr(), getch(), getche() dalam file conio.h

7.2 Procedure

Procedure memiliki dua bentuk, yaitu procedure tanpa parameter dan procedure dengan parameter.

a) Procedure tanpa parameter

Bentuk umumnya sebagai berikut :

```
void nama_prosedur(void)
{
  statement ;
}
```

Contoh programnya sebagai berikut :

Ketikkan script program berikut :

Nama program : procedure1.cpp

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

void cetak();

void main()
{
  int nilai;

  cetak();

  cout << "\nSelesai !!!";
  getch();
}

void cetak()
{
  cout << "Haii, saya adalah procedure ";
}
```

Perintah memanggil procedure

Sub Program / PROCEDURE

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :

```
Hai saya adalah prosedur
Selesai
```

b) Procedure dengan parameter

Bentuk umumnya sebagai berikut :

```
void nama_prosedur(tipe_data variabel_input, tipe_data variabel_inputn)
{
    statement ;
}
```

Contoh programnya sebagai berikut :

Ketikkan script program berikut :

Nama Program : procedure2.cpp

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

void cetak(int A);

void main()
{
  int nilai;

  cout << "Input sebuah angka : "; cin >> nilai;

  cetak(nilai);

  cout << "\nSelesai !!!";
  getch();
}

void cetak(int A)
{
  cout << "Angka yang diinput adalah " << A;
}
```

Perintah memanggil procedure

Sub Program / PROCEDURE

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :

```
Input sebuah angka : 100
```

Angka yang diinput adalah : 100
Selesai

Latihan :

1. Buatlah fungsi untuk menghitung luas segitiga?
2. Buatlah program rekursi untuk mencari Nilai n faktorial
3. Buatlah program dengan cara rekursi untuk menampilkan perkalian 3 buah bilangan tersebut nilainya diinputkan.
4. Buat program menggunakan prosedur tanpa parameter untuk menginput biodata mahasiswa :

   Nim : Nim Anda Sendiri
   Nama : Nama Anda Sendiri
   Nilai : 100
   Keterangan : Lulus

IX. Perulangan atau Iterasi (For, While)

Sebuah / kelompok instruksi diulang untuk jumlah pengulangan tertentu. Baik yang terdifinisikan sebelumnya ataupun tidak. Struktur pengulangan terdiri atas dua bagian :

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

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

Perintah atau notasi dalam struktur pengulangan adalah :

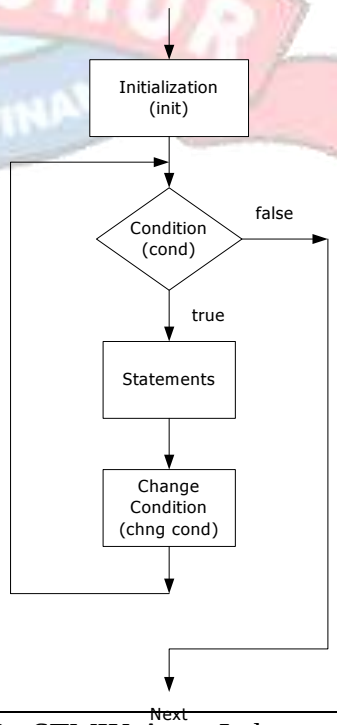
1. Pernyataan for
2. Pernyataan while

9.1 Pernyataan For

Pernyataan for digunakan untuk menghasilkan pengulangan(looping) beberapa kali tanpa penggunaan kondisi apapun. Pada umumnya looping yang dilakukan oleh for telah diketahui batas awal, syarat looping dan perubahannya. Pernyataan for digunakan untuk melakukan looping. Pada umumnya looping yang dilakukan oleh for telah diketahui batas awal, syarat looping dan perubahannya. Selama kondisi terpenuhi, maka pernyataan akan terus dieksekusi.

Bentuk Umumnya :

```
for (inisialisasi ; kondisi ; perubahan)
{
 Statement;  }
```

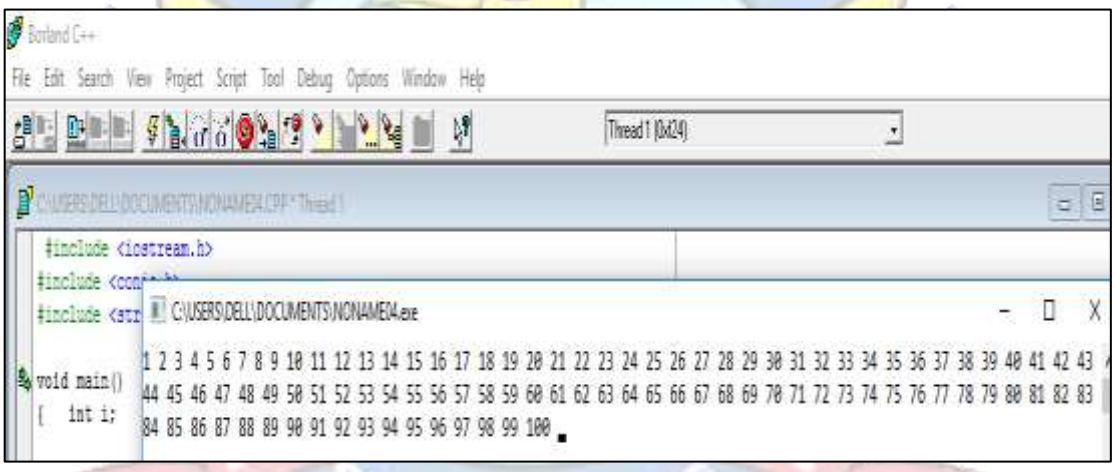


Ketikkan script program berikut :

```
// Program mencetak angka 1-100
# include <iostream.h>

void main ()
{
    int i;

    for (i=1; i<=100; i++)
    cout << i << " " <<endl;
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :



Bagaimana jika program diatas diubah menjadi

Ketikkan script program berikut :

```
# include <iostream.h>

void main()
{
    int i;

    for (i=1; ;i++)
    cout << i << endl;
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :

Maka hasil output akan keluar tidak berhingga

Program diatas akan menampilkan bilangan yang banyaknya tak terhingga sehingga dapat membuat komputer anda berhenti bekerja. Contoh diatas juga merupakan prinsip membuat bom program (contohnya : bom mail)

Pernyataan for dapat berada di dalam pernyataan for lainnya yang biasa disebut nested for.

Ketikkan script program berikut :

```
// Program membentuk segitiga
# include <iostream.h>

void main ()
{
   int tinggi,
       baris,
       kolom;

   cout << "Tinggi segitiga : ";
   cin >> tinggi;

   cout << endl;
   for (baris=1; baris <= tinggi; baris++)
   {
      for (kolom = 1; kolom <= baris; kolom++)
         cout << '*';
      cout << endl;
   }
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :

```
C:\USERS\DELL\DOCUMENTS\NONAME04.exe
Tinggi segitiga :6

*
**
***
****
*****
******
```

9.2 Pernyataan while

Pernyataan while merupakan salah satu pernyataan yang berguna untuk memproses suatu pernyataan atau beberapa pernyataan beberapa kali. Pernyataan while memungkinkan statemen-statemen yang ada didalamnya tidak diakukan sama sekali.

Bentuk Umumnya :

```
while (kondisi)
{  Pernyataan ;
 }
```

Contoh :

Ketikkan script program berikut :

```
# include <iostream.h>

void main ()
{
   int i;

   i=0;
   while (i<10)
   {
      cout << "C++" << endl;
      i++;
   }
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :

```
C:\USERS\DELL\DOCUMENTS\NONAME04.exe
C++
C++
C++
C++
C++
C++
C++
C++
C++
C++
```

Penjelasan : Program diatas digunakan untuk mengulangan tulisan sebanyak 10 kali

9.3 Pernyataan Do.. while

Pernyataan do…while mirip seperti pernyataan while, hanya saja pada do…while pernyataan yang terdapat didalamnya minimal akan sekali dieksekusi.

Bentuk Umumnya :

```
do {
pernyataan ;
} while(kondisi);
```

Contoh :

Ketikkan script program berikut :

```
# include <iostream.h>

void main ()
{
   int i;
   i=0;

   do
   {
      cout << "C++" << endl;
      i++;
   }
   while (i<10);
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :

```
C:\USERS\DELL\DOCUMENTS\NONAME04.exe
C++
C++
C++
C++
C++
C++
C++
C++
C++
C++
```

Latihan :

1. Buatlah program untuk mencetak deret 10 9 8 7 6 5 4 3 2 1
2. Buatlah program untuk mencetak deret 2 4 6 8 10 12 14 16 18 20
3. Buatlah program dengan ouput sebagai berikut :

```
Input Jumlah Ulang : 4
Pangkalpinang
Pangkalpinang
Pangkalpinang
Pangkalpinang
```

4. Buatlah program untuk mencetak (gunakan perulangan while atau for)

```
* * * *
* * *
* *
*
```

5. Buatlah program yang menampilkan 5 buah bilangan, yaitu mulai dari bilanagn ke 5 sampai bilangan ke 1 dengan nilai awal bilangan 8. Tampilan bilangan tersebut adalah menurun dan contohnya adalah : bilangan ke 5, i=3 (diperoleh dari 8-5)dan seterusnya sampai bilangan 1, i=7 (diperoleh dari 8-1=7)

X. Perulangan atau Iterasi (do..while, continue break)

Sebuah / kelompok instruksi diulang untuk jumlah pengulangan tertentu. Baik yang terdifinisikan sebelumnya ataupun tidak. Struktur pengulangan terdiri atas dua bagian :

1. Kondisi pengulangan yaitu ekspresi boolean yang harus dipenuhi untuk melaksanakan pengulangan
2. Isi atau badan pengulangan yaitu satu atau lebih pernyataan (aksi) yang akan diulang.

Perintah atau notasi dalam struktur pengulangan adalah :

1. Pernyataan do..while
2. Pernyataan continue dan break

10.1 Pernyataan Do.. while

Pernyataan do…while mirip seperti pernyataan while, hanya saja pada do…while pernyataan yang terdapat didalamnya minimal akan sekali dieksekusi.

Bentuk Umumnya :

```
do {
pernyataan ;
} while(kondisi);
```

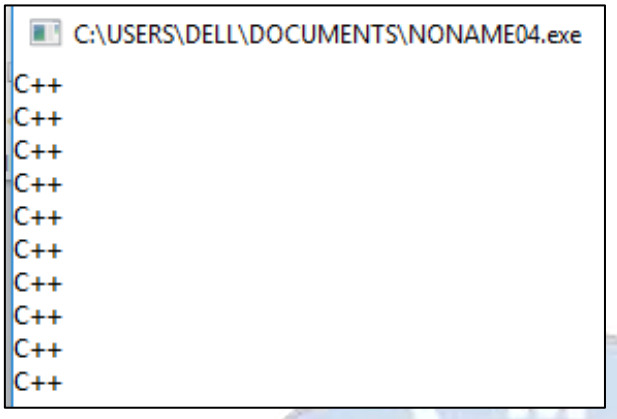
Contoh :

Ketikkan script program berikut :

```
# include <iostream.h>

void main ()
{
   int i;
   i=0;

   do
   {
      cout << "C++" << endl;
      i++;
   }
   while (i<10);
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :



10.2 Pernyataan continue dan break

Pernyataan break akan selalu terlihat digunakan bila menggunakan pernyataan switch. Pernyataan ini juga digunakan dalam loop. Bila pernyataan ini dieksekusi, maka akan mengakhiri loop dan akan menghentikan itrasi pada saat tersebut.

Pernyataan continue digunakan untuk pergi ke bagian awal dari blok loop untuk memulai iterasi berikutnya.

Contoh :

```
// Program Continue dan break

# include <iostream.h>
void main ()
{
   int i;
   for (i=0; i<10; i++)
   {
      if (i==4) continue;
      cout << " Bilangan " << i <<endl;
      if (i==6) break;
   }
}
```

Outputnya :

```
Bilangan 0
Bilangan 1
Bilangan 2
Bilangan 3
Bilangan 5
Bilangan 6
```

Latihan :

1. Buatlah program untuk mencetak deret 10 9 8 7 6 5 4 3 2 1
2. Buatlah program untuk mencetak deret 2 4 6 8 10 12 14 16 18 20
3. Buatlah program dengan ouput sebagai berikut :

```
Input Jumlah Ulang : 4
Pangkalpinang
Pangkalpinang
Pangkalpinang
Pangkalpinang
```

4. Buatlah program untuk mencetak (gunakan perulangan while atau for)

```
* * * *
* * *
* *
*
```

6. Buatlah program yang menampilkan 5 buah bilangan, yaitu mulai dari bilanagn ke 5 sampai bilangan ke 1 dengan nilai awal bilangan 8. Tampilan bilangan tersebut adalah menurun dan contohnya adalah : bilangan ke 5, i=3 (diperoleh dari 8-5)dan seterusnya sampai bilangan 1, i=7 (diperoleh dari 8-1=7)

XI. ARRAY

11.1 ARRAY

Larik merupakan sekumpulan data yang mempunyai nama dan tipe yang sama. Larik sering disebut juga variabel berindeks. Nilai suatu data dalam larik ditentukan oleh nama dan indeks. Larik banyak digunakan pada operasi yang melibatkan indeks seperti pada statistik dan matriks. Tipe data larik dapat berupa larik satu demensi, dua demensi, tiga demensi atau banyak dimensi.

Bentuk Umum Larik Satu Dimensi :

```
tipe_larik nama_larik [ukuran]
```

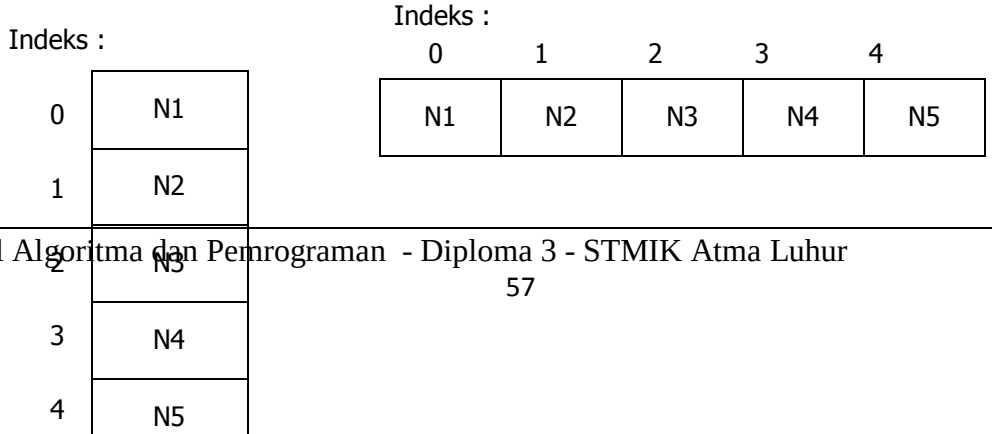
Bentuk Umum Larik Dua Dimensi :

```
tipe_larik nama_larik [ukuran1][ukuran2]
```

Perhatikan :

- Tanda kurung [ ] digunakan untuk menunjukkan elemen larik
- Perhitungan elemen larik dimulai dari 0, bukan 1

C++ tidak mengecek larik. Bila anda menyatakan int x[10], ini artinya 10 elemen yang dimulai dari 0. Karena itu elemen terakhir larik adalah x[9]. Bila anda salah mereferensikannya dengan x[10], anda akan mendapatkan harga yang tidak terpakai. Akan lebih buruk lagi jika anda memberikan harga ke x[10], yang tidak dapat diterima.



11.2 Representasi Larik

Misalkan kita memiliki sekumpulan data ujian seorang siswa, ujian pertama bernilai 90, kemudian 95,78,85. Sekarang kita ingin menyusunnya sebagai suatu data kumpulan ujian seorang siswa. Dalam array kita menyusunnya sebagai berikut

```
ujian[0] = 90;
ujian[1] = 95;
ujian[2] = 78;
ujian[3] = 85;
```

Empat pernyataan diatas memberikan nilai kepada array ujian. Tetapi sebelum kita memberikan nilai kepada array, kita harus mendeklarasikannya terlebih dahulu, yaitu :

```
int ujian[4];
```

Perhatikan bahwa nilai 4 yang berada didalam tanda kurung menujukkan jumlah elemen larik, bukan menunjukkan elemen larik yang ke-4. Jadi elemen larik ujian dimulai dari angka 0 sampai 3. Pemrogram juga dapat menginisialisasi larik sekaligus mendeklarasikannya, sebagai contoh :

```
int ujian[4] = {90,95,78,85};
```

Elemen terakhir dari larik diisi dengan karakter '\0'. Karakter ini memberitahu kompiler bahwa akhir dari elemen larik telah dicapai. Walaupun pemrogram tidak dapat melihat karakter ini secara eksplisit, namun kompiler mengetahui dan membutuhkannya. Sekarang kita akan membuat daftar beberapa nama dosen di STMIK Atma Luhur

```
char dosen[3][15] ;
char dosen[0][15] = "Okkita";
```

char dosen[1][15] = “Budi”;
char dosen[2][15] = “Delpiah”;

Larik diatas terlihat berbeda denga contoh larik pertama kita. Perhatikan bahwa pada larik dosen memilih dua buah tanda kurung [ ][ ]. Larik seperti itu disebut larik dua.

dimensi. Tanda kurung pertama menyatakan total elemen yang dapt dimiliki oleh larik dosen dan tanda kurung kedua menyatakan total elemen yang dapat dimiliki setiap elemen larik dosen. Dalam contoh diatas, tanda kurung kedua menyatakan karakter yang menyatakan nama dosen.

11.3 Menghitung Jumlah Elemen Array

Karena fungsi sizeof() mengembalikan jumlah byte yang sesuai dengan argumennya, maka operator tersebut dapat digunakan untuk menemukan jumlah elemen array, misalnya

```
int array[ ] = {26,7,82,166};
cout<<sizeof(array)/sizeof(int);
```

akan mengembalikan nilai 4, yaitu sama dengan jumlah elemen yang dimiliki larik.

11.4 Melewatkan Array Sebagai Argumen Fungsi

Larik dapat dikirim dan dikembalikan oleh fungsi. Pada saat larik dikirim ke dalam fungsi, nilai aktualnya dapat dimanipulasi.

Contoh :

Ketikkan script program berikut :

```
#include <iostream.h>
void ubah(int x[]);
void main()
{
   int ujian[] = {90,95,78,85};
   ubah(ujian);
   cout<<" Elemen kedua dari array ujian adalah "<<ujian[1]<<endl;
}

void ubah(int x[])
{
   x[1] = 100;
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :

```
Elemen kedua dari array ujian adalah : 95
```

Latihan :

Buatlah program yang menghitung jumlah elemen dalam suatu array (larik) dengan array (larik) 1 dimensi { 1,3,5,4,7,2,99,16,45,67,89,45}

XII. STRUKTUR

Struktur digunakan untuk mengelompokkan sejumlah data yang memiliki tipe dan ukuran yang berbeda. Struktur mirip dengan array, hanya saja array memiliki data yang memiliki tipe dan ukuran yang sama. Variabel-variabel yang membentuk sebuah struktur dinamakan elemen struktur.

12.1 Deklarasi Struktur

Struktur dapat dideklarasikan dengan sintaks berikut.

```
struct nama_struktur {
        elemen_struktur_1;
        elemen_struktur_2;
        elemen_struktur_3;
        …
};
```

atau

```
struct {
        elemen_struktur_1;
        elemen_struktur_2;
        elemen_struktur_3;
        …
} nama_struktur;
```

Contoh deklarasi :

```
1  struct mahasiswa {
2          char nim[10];
3          char nm_mhs[20];
4          float ipk;
5  };
```

```
1  struct {
2          char nim[10];
3          char nm_mhs[20];
4          float ipk;
5  } mahasiswa;
```

Struktur juga dapat diinisialisasi pada suatu pendeklarasian seperti dapat dilihat pada contoh berikut.

```
1  struct {
2          char nim[10];
3          char nm_mhs[20];
4          float nilai;
5  } mahasiswa = {“1011500096”, “Eza Budi Perkasa”, 3.67};
```

Elemen pada struktur dapat diakses dengan sintaks **nama_struktur.elemen_struktur**. Sebagai contoh, untuk mengakses elemen **nilai** pada struktur **mahasiswa**, maka penulisannya adalah **mahasiswa.nilai**.

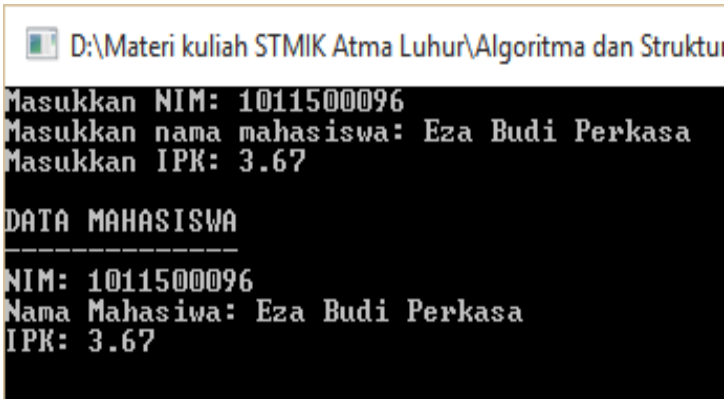
Contoh program:

Ketikkan script program berikut :

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

void main() {
   struct {
      char nim[11];
      char nm_mhs[21];
      float ipk;
   } mahasiswa;
   cout << "Masukkan NIM: "; cin >> mahasiswa.nim;
   cout << "Masukkan nama mahasiswa: "; gets(mahasiswa.nm_mhs);
   cout << "Masukkan IPK: "; cin >> mahasiswa.ipk;
   cout << endl;
   cout << "DATA MAHASISWA" << endl;
   cout << "---------------" << endl;
   cout << "NIM: " << mahasiswa.nim << "\0" << endl;
   cout << "Nama Mahasiwa: " << mahasiswa.nm_mhs << "\0" << endl;
   cout << "IPK: " << mahasiswa.ipk << endl;
   getch();
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :



12.2 Struktur Dalam Struktur

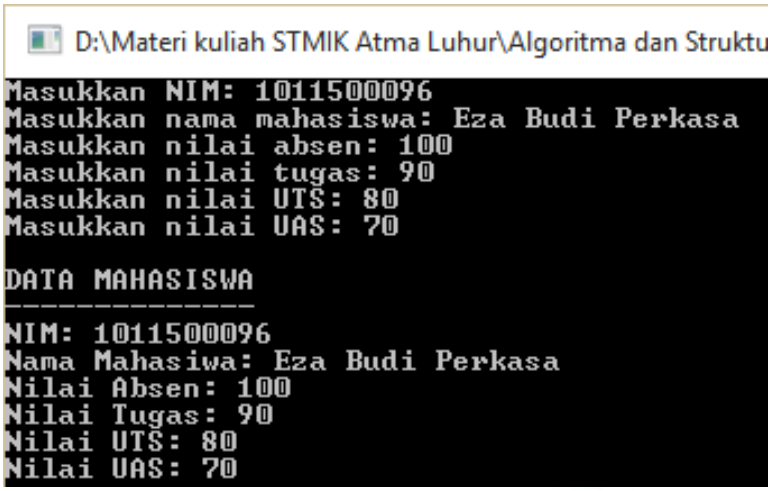
Sebuah struktur dapat berada di dalam struktur lainnya. Contoh penerapan struktur dalam struktur tersebut dapat dilihat pada program berikut.

Ketikkan script program berikut :

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

void main() {
   struct mahasiswa {
      char nim[11];
      char nm_mhs[21];
   };
   struct nilai {
      float absen;
      float tugas;
      float uts;
      float uas;
   };
   struct {
      struct mahasiswa mhs;
      struct nilai nil;
   } data_mahasiswa;
   cout << "Masukkan NIM: "; cin >> data_mahasiswa.mhs.nim;
   cout << "Masukkan nama mahasiswa: "; gets(data_mahasiswa.mhs.nm_mhs);
   cout << "Masukkan nilai absen: "; cin >> data_mahasiswa.nil.absen;
   cout << "Masukkan nilai tugas: "; cin >> data_mahasiswa.nil.tugas;
   cout << "Masukkan nilai UTS: "; cin >> data_mahasiswa.nil.uts;
   cout << "Masukkan nilai UAS: "; cin >> data_mahasiswa.nil.uas;
   cout << endl;
   cout << "DATA MAHASISWA" << endl;
   cout << "--------------" << endl;
   cout << "NIM: " << data_mahasiswa.mhs.nim << "\0" << endl;
   cout << "Nama Mahasiwa: " << data_mahasiswa.mhs.nm_mhs << "\0" << endl;
   cout << "Nilai Absen: " << data_mahasiswa.nil.absen << endl;
   cout << "Nilai Tugas: " << data_mahasiswa.nil.tugas << endl;
   cout << "Nilai UTS: " << data_mahasiswa.nil.uts << endl;
   cout << "Nilai UAS: " << data_mahasiswa.nil.uas << endl;
   getch();
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :



12.3 Array Pada Struktur

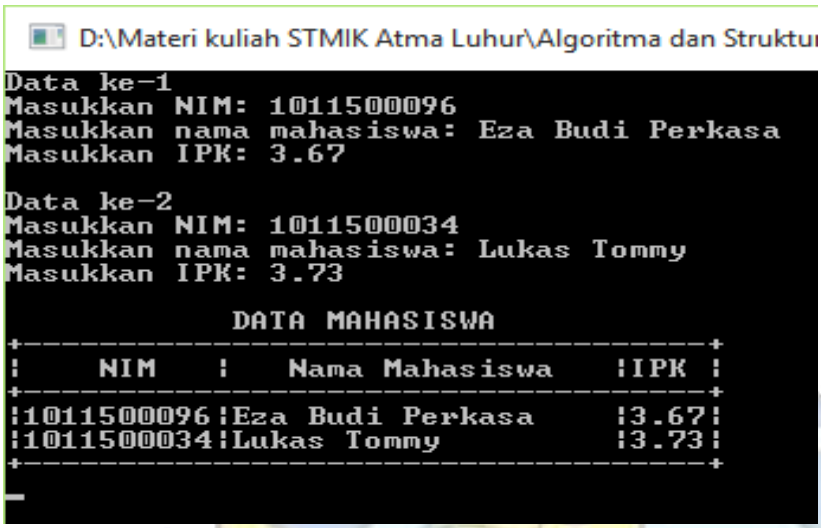
Suatu array dapat digunakan pada struktur. Contoh penggunaannya dapat dilihat pada program berikut.

Ketikkan script program berikut :

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

void main() {
   int i, sp;
   struct {
      char nim[11];
      char nama[21];
      float ipk;
   } mahasiswa[2];
   for(i=0;i<2;i++) {
      cout << "Data ke-" << (i + 1) << endl;
      cout << "Masukkan NIM: "; cin >> mahasiswa[i].nim;
      cout << "Masukkan nama mahasiswa: "; gets(mahasiswa[i].nama);
      cout << "Masukkan IPK: "; cin >> mahasiswa[i].ipk;
      cout << endl;
   }
   cout << "            DATA MAHASISWA            " << endl;
   cout << "+--------------------------------------+" << endl;
   cout << "|    NIM    |   Nama Mahasiswa   |IPK |" << endl;
   cout << "+--------------------------------------+" << endl;
   for(i=0;i<2;i++) {
      cout << "|" << mahasiswa[i].nim;
      cout << "|" << mahasiswa[i].nama;
      for(sp=1;sp<20-strlen(mahasiswa[i].nama)+1;sp++) { //Menambah spasi
         cout << " ";
      }
      cout << "|" << mahasiswa[i].ipk << "|" << endl;
   }
   cout << "+--------------------------------------+" << endl;
   getch();
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :



XIII. PREPROCESSOR DIRECTIVE

13.1 Preprocessor Directive

Preprocessor directive adalah suatu perintah yang termasuk dalam program tetapi bukan instruksi dari program itu sendiri, melainkan untuk preprosesor. Preprosesor dijalankan secara otomatis oleh kompiler ketika proses kompilasi program berlangsung, dibuat nilai pembuktian pertama, dan menerjemahkan kode program di dalam kode objek. Dalam penggunaannya, preprocessor directive selalu dimulai dengan tanda #. Beberapa preprocessor directive yang dapat digunakan adalah sebagai berikut.

a. #define
b. #include
c. #if - #endif
d. #if - #else - #endif
e. #elif
f. #undef
g. #ifdef dan #ifndef

A. #define

Preprocessor directive #define digunakan untuk mendefinisikan suatu nilai tertentu kepada suatu nama konstanta.

Sintaks: **#define nama_konstanta nilai_konstanta**

Contoh: #define A 6 (berarti nilai 6 didefinisikan sebagai A)

Dalam pendeklarasian preprocessor directive #define, nama konstanta sebaiknya ditulis dengan huruf kapital untuk membedakannya dengan nama variabel. Adapun nilai konstanta merupakan nilai yang diberikan pada nama konstanta. Nilai konstanta dapat berupa:

- Numerik → #define PI 3.14
- Karakter → #define HURUF ‘B’
- String → #define JABATAN “Kaprodi”
- Pernyataan → #define CETAK (“Borland C++”)
- Fungsi → #define LUAS (n*n)

Setelah #define ditentukan di dalam program, maka cukup dituliskan nama konstantanya saja. #define akan mengganti semua nama konstanta dengan nilainya sebelum proses kompilasi dimulai.

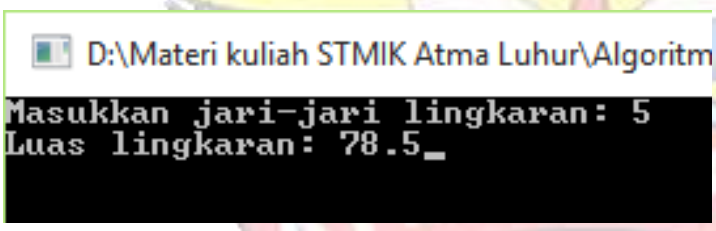
Contoh program :

Ketikkan script program berikut :

```
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
#define PI 3.14
#define LUAS_LINGKARAN(r) PI*r*r

void main() {
    int jari;
    cout << "Masukkan jari-jari lingkaran: ";
    cin >> jari;
    cout << "Luas lingkaran: " << LUAS_LINGKARAN(jari);
    getch();
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :



B. #include

Preprosesor #include berfungsi untuk memasukkan atau menyertakan file header ke dalam program yang dibuat. #include memiliki dua bentuk penulisan:

- #include “nama_file_header”
- #include <nama_file_header>

Untuk penulisan pertama, kompiler akan mencari file header yang disebutkan pada direktori yang sedang aktif dan apabila tidak ditemukan, maka kompiler akan mencari direktori tempat file header tersebut berada. Untuk penulisan kedua, pertama kali kompiler akan mencari file header yang disebutkan pada direktori yang ada file headernya, kecuali pada direktori yang sedang aktif.

C. #if - #endif

Preprosesor #if - #endif digunakan untuk mengompilasi jika pernyataan kondisi pada #if bernilai benar dan jika tidak akan diabaikan. Pernyataan kondisi berupa ekspresi konstanta yang dideklarasikan dengan #define.

Sintaks :

```
#if ekspresi
        pernyataan;
#endif
```

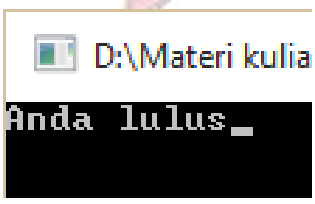
Contoh program :

Ketikkan script program berikut :

```
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
#define nilai 85

void main() {
    #if nilai > 56
        cout << "Anda lulus";
    #endif
    getch();
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :



D. #if - #else - #endif

Preprosesor #if - #else - #endif digunakan untuk mengompilasi jika pernyataan kondisi pada #if bernilai benar atau #else jika salah. Pernyataan kondisi berupa ekspresi konstanta yang dideklarasikan dengan #define.

Sintaks :

```
#if ekspresi
        pernyataan-1;
#else
        pernyataan-2;
#endif
```

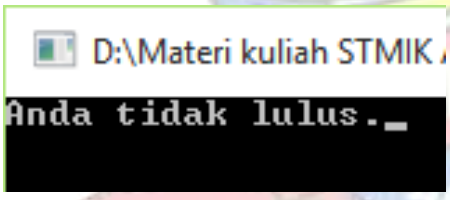
Contoh program:

Ketikkan script program berikut :

```
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
#define nilai 50

void main() {
    #if nilai > 56
        cout << "Anda lulus";
    #else
        cout << "Anda tidak lulus.";
    #endif
    getch();
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :



E. #elif

Preprosesor #elif digunakan untuk mengompilasi pernyataan bertingkat. Dalam hal ini, #elif merupakan kombinasi dari #if dan #else. Perintah dijalankan sesuai kondisi yang telah ditentukan. Hasilnya hanya dapat dijalankan sesuai dengan ketentuan yang benar.

Sintaks:

```
#if ekspresi-1
        pernyataan-1;
#elif ekspresi-2
        pernyataan-2;
…
#elif ekspresi-n
        pernyataan-n;
```

#endif

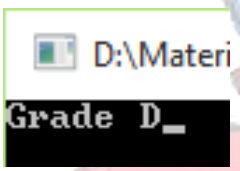
Contoh program:

Ketikkan script program berikut :

```
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
#define nilai 50

void main() {
    #if nilai >= 0 && nilai < 45
        cout << "Grade E";
    #elif nilai >= 45 && nilai < 57
        cout << "Grade D";
    #elif nilai >= 57 && nilai < 68
        cout << "Grade C";
    #elif nilai >= 68 && nilai < 78
        cout << "Grade B";
    #elif nilai >= 78 && nilai < 101
        cout << "Grade A";
    #else
        cout << "Nilai ngawur!";
    #endif
    getch();
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :



F. #undef

Preprosesor #undef digunakan untuk menghilangkan nilai konstanta yang telah didefinisikan oleh #define.

Sintaks :

```
#undef nama_konstanta
```

Contoh program :

Ketikkan script program berikut :

```
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
#define NILAI_MAKS 150
#if NILAI_MAKS > 100
    #undef NILAI_MAKS
    #define NILAI_MAKS 100
#elif NILAI_MAKS < 0
    #undef NILAI_MAKS
    #define NILAI_MAKS 0
#else
    #undef NILAI_MAKS
    #define NILAI_MAKS 999
#endif

void main() {
    cout << NILAI_MAKS;
    getch();
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :

G. #ifdef dan #ifndef

Preprosesor #ifdef dan #ifndef memeriksa apakah konstanta yang dideklarasikan oleh #define terdefinisi. Apabila konstanta terdefinisi dan kita menggunakan #ifdef, maka pernyataan yang berada pada blok preprosesor tersebut akan dieksekusi. Apabila konstanta tidak terdefinisi dan kita menggunakan #ifndef, maka pernyataan pada blok preprosesor tersebut akan dieksekusi.

Sintaks:

#ifdef nama_konstanta	**#ifndef nama_konstanta**

pernyataan; **#endif**	**pernyataan;** **#endif**

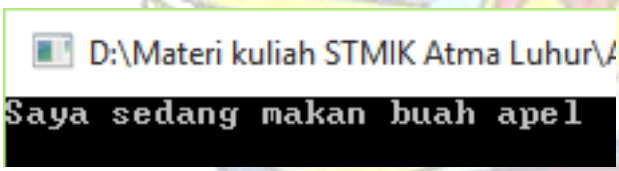
Contoh program (#ifdef):

Ketikkan script program berikut :

```
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
#define BUAH "apel"

void main() {
    #ifdef BUAH
        cout << "Saya sedang makan buah " << BUAH;
    #endif
    getch();
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :



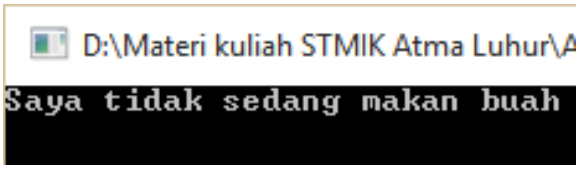
Contoh program (#ifndef):

Ketikkan script program berikut :

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

void main() {
    #ifndef BUAH
        cout << "Saya tidak sedang makan buah";
    #endif
    getch();
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

Outputnya :



XIV. FILE HEADER

14.1 File Header

File header adalah suatu file dengan ekstensi .h. File ini berisi deklarasi fungsi dan definisi konstanta. Selain file-file header standar yang disediakan oleh C++, kita juga dapat membuat file header sendiri dengan cara yang sama dengan membuat program utama. Perlu diperhatikan bahwa pada saat menyimpan file header yang telah dibuat, kita harus menggunakan ekstensi .h.

Contoh header (simpan dengan nama "hitung.h") :

```
#define PI 3.14
#define LUAS_PERSEGI(s) s*s
#define LUAS_LINGKARAN(r) PI*r*r
#define KELILING_PERSEGI(s) 6*s
#define KELILING_LINGKARAN(r) 2*PI*r
#define VOLUME_KUBUS(s) s*s*s
#define VOLUME_BOLA(r) 1.33*PI*r*r*r
```

Contoh program :

Ketikkan script program berikut :

```
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
#include "hitung.h"

void main() {
   float bil;
   cout << "Masukkan sembarang bilangan: "; cin >> bil;
   cout << endl;
   cout << "Luas persegi dengan sisi " << bil << " adalah " <<
           LUAS_PERSEGI(bil) << endl;
   cout << "Luas lingkaran dengan jari-jari " << bil << " adalah " <<
           LUAS_LINGKARAN(bil) << endl;
   cout << "Keliling persegi dengan sisi " << bil << " adalah " <<
           KELILING_PERSEGI(bil) << endl;
   cout << "Keliling lingkaran dengan jari-jari " << bil << " adalah " <<
           KELILING_LINGKARAN(bil) << endl;
   cout << "Volume kubus dengan rusuk " << bil << " adalah " <<
           VOLUME_KUBUS(bil) << endl;
   cout << "Volume bola dengan jari-jari " << bil << " adalah " <<
           VOLUME_BOLA(bil) << endl;
   getch();
}
```

Simpan hasil pekerjaan Anda di folder masing-masing, dan jalankan (dengan memilih menu **Debug** -> **Run** atau **Ctrl+F9**)

DAFTAR PUSTAKA

Algoritma & Struktur Data dengan C, C++ dan Java, Moh. Syukani, Mitra Wacana Media, Jakarta, 2004

Asyiknya Belajar Struktur Data di Planet C++, Dwi Sanjaya, Elex Media Komputindo, 2005

Belajar Pemrograman Dengan Bahasa C++ dan Java dari Nol Menjadi Andal, M. Shalahuddin - Rosa A. S., Informatika Bandung, 2007

Dasar-Dasar Algoritma & Pemrograman, Fathul Wahid, Penerbit ANDI, 2004

Panduan Pemrograman C++, Frieyadie, Penerbit ANDI, Yogyakarta, 2006.

Struktur Data (Algoritma & Struktur Data 2) Dengan C, C++, Moh. Syukani, Mitra Wacana Media, Jakarta, 2007.