Rina Firliana, M. Kom,
Patmi Kasih, M. Kom

adjie media nusantara

Algoritma & Pemrograman C++

Rina Firliana, M. Kom
Patmi Kasih, M. Kom

Algoritma & Pemrograman C++

Penerbit Adjie Media Nusantara, Nganjuk

Algoritma & Pemrograman C++

Ditulis oleh:
Rina Firliana, M. Kom, & Patmi Kasih, M. Kom

ISBN: 978-602-5605-25-3

viii, 82 hal., 17 x 25 Cm

Desain Sampul : Adjie Media Nusantara
Layout : Adjie Media Nusantara

Diterbitkan oleh Penerbit **Adjie Media Nusantara**

CV. Adjie Media Nusantara
Jl. Demang Palang No.9 Watudandang Prambon Nganjuk
Kode Pos: 64484
Telp: (0358) 792436 / 082244863077
E-mail: penerbit@adjiemedianusantara.co.id
Website: adjiemedianusantara.co.id

Cetakan Pertama, Januari 2018
Dicetak oleh CV. Adjie Media Nusantara

Kata Pengantar

Dengan mengucapkan Alhamdulillah segala puji syukur selalu terpanjatkan kepada Allah SWT., atas limpahan rahmat Nya sehingga dapat terselesaikannya penyusunan Buku Algoritma dan Pemrograman C++ ini. Penulis berharap Buku Algoritma dan Pemrograman C++ ini dapat mempermudah pemahaman pada materi Algoritma dan Pemrograman C++ dapat memperlancar kegiatan Algoritma dan Pemrograman C++.

Ucapan terima kasih kami sampaikan kepada semua pihak yang terkait dalam penyelesaian penyusunan Buku Algoritma dan Pemrograman C++ ini. Penulis menyadari bahwa Buku Algoritma dan Pemrograman C++ ini masih jauh dari sempurna, oleh karena itu penulis mengharapkan kritik dan saran yang dapat dijadikan masukan dalam perbaikan dimasa yang akan datang. Semoga penyusunan Buku Algoritma dan Pemrograman C++ ini dapat memberi manfaat yang sebesar-besarnya.

Penulis

Daftar Isi

Bab I

ALGORITMA DAN PEMROGRAMAN

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 merupakan logika, metode, dan tahapan (urutan) sistematis yang digunakan untuk memecahkan suatu permasalahan. Algoritma dapat dituliskan dengan banyak cara, mulai dari menggunakan bahasa alami yang digunakan sehari–hari, simbol grafik bagan alir (flowchart), sampai menggunakan bahasa pemrograman seperti bahasa C atau C++.

Program adalah kumpulan instruksi komputer, sedangkan metode dan tahapan sistematis dalam program adalah algoritma. Program ini ditulis dengan menggunakan bahasa pemrograman. Jadi bisa kita sebut bahwa program adalah suatu implementasi dari bahasa pemrograman.
Beberapa pakar memberi formula bahwa:

program = struktur data + algoritma

Bagaimanapun juga struktur data dan algoritma berhubungan sangat erat pada sebuah program. Algoritma yang baik tanpa pemilihan struktur data yang tepat akan membuat program menjadi kurang baik, demikian juga sebaliknya. Struktur data disini bisa berupa list, tree, graph, dsb.

A. PENGENALAN BAHASA C++

Sekilas C++

Bahasa pemrograman computer terdiri atas dua bagian, yaitu bahasa pemrograman tingkat tinggi (*high level language*) dan bahasa pemrograman tingkat rendah (*low level language*). Penggolongan ini

menggunakan C dan/atau C++. C tersedia untuk hampir semua komputer.

Pada akhir dekade 1970-an, C telah berkembang dengan menjadi sesuatu yang sekarang disebut "C tradisional", "C klasik", atau "C Kernighan dan Ritchie".

C++ adalah penambahan dari C, dikembangkan oleh Bjarne Stroustup pada awal dekade 1980 an di Bell Laboratories. C++ memberikan tambahan fitur yang meningkatkan kekuatan bahasa C, dan yang lebih penting lagi, kemampuan untuk pemograman berbasis object (Object Oriented Programming).

B. Kelebihan dan Kekurangan

Kelebihan Bahasa C /C++ :

- Bahasa C++ tersedia hampir di semua jenis komputer.
- Kode bahasa C/C++ sifatnya adalah portable dan fleksibel untuk semua jenis komputer.
- Proses executable program bahasa C/C++ lebih cepat.
- Dukungan pustaka yang banyak.
- C adalah bahasa yang terstruktur.
- C++ sudah mendukung OOP (Object Oriented Programming).

Kekurangan Bahasa C/C++ :

- Banyaknya Operator serta fleksibilitas penulisan program kadang–kadang membingungkan pemakai.
- Bagi pemula pada umumnya akan kesulitan menggunakan pointer dan penerapan konsep OOP.

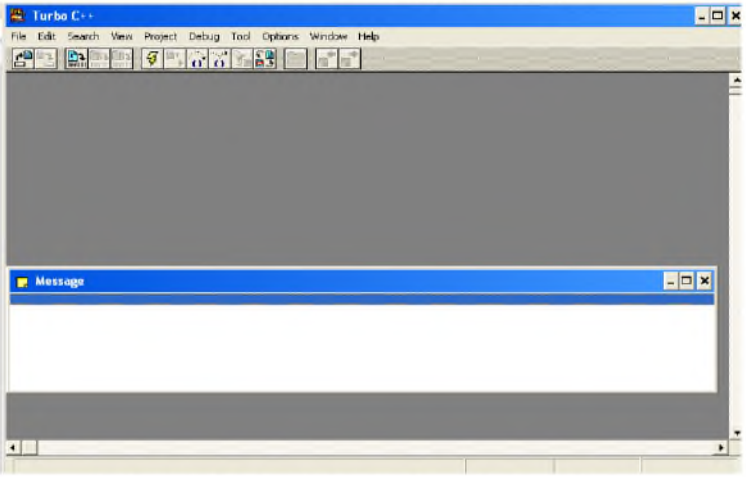
Editor Bahasa C/C++

Untuk memulai membuat program, tersedia berbagai editor yang dapat digunakan , diantaranya : Turbo C++, Borland C++, C++ builder, Microsoft Visual C++, dlsb.

C. Langkah–langkah menuliskan program dalam Borland C++

1. Bukalah editor Borland C++ melalui START menu, atau double click icon turbo C++

2. Maka akan muncul tampilan aplikasi sebagai berikut,



Gambar: User Interface Borland C++ 5.02

3. Source Code Program C/C++ dapat ditulis di text editor Borland C++. File → New → Text Edit, maka akan muncul layar putih tempat untuk menulis program C++.
4. Program dapat disimpan dengan Pilih Menu File, Save As atau Save, Save All.
5. Untuk mengeksekusi dan mengcompile (menjalankan) program C++ yang telah kita buat kita mengklik tombol  petir berikut:

   Kompile file dapat juga dilakukan dengan (ALT + F9 atau pilih submenu Compile)

   Compiler dijalankan untuk mengubah source code menjadi sebuah bahasa program.

   Setelah kita kompile file yang berisi source code, maka sebagai hasil kompilasi, kita akan mendapatkan suatu file yang bisa dijalankan (executable file). Menjalankan program yang kita buat berarti menjalankan file hasil proses kompilasi tersebut.

D. Struktur program C++

Struktur dasar program bahasa C++ terdiri dari tiga bagian, sebagai berikut:

- ***Preprocessor Directives***

  Preprocessor directives merupakan perintah dalam bahasa C++ yang akan membuat dan mengkompilasi file perantara dengan program utam

- **Deklarasi Global**

  Deklarasi global merupakan bagian program C++ tempat pendeklarasian konstanta, variabel, fungsi atau header fungsi, tipe data baru, atau struktur data yang memiliki sifat global.

- **Fungsi Utama**

  Fungsi utama adalah fungsi pertama yang akan dieksekusi oleh kompiler C++. Dan kompiler C++ hanya akan mengeksekusi fungsi-fungsi lain yang dipanggil pada fungsi utama atau yang memiliki hubungan dengan fungsi utama secara tidak langsung.

  Bagian ini digunakan untuk meletakkan seluruh instruksi program. Instruksi-instruksi yang akan diberikan untuk dikerjakan ditulis di di dalam main() diantara "{" dan "}".Yang perlu diperhatikan bahwa setiap instruksi atau baris perintah diakhiri dengan tanda baca titik koma (;) atau ***semicolon***.

  Definisi fungsi merupakan tempat pendefinisian header fungsi yang telah dideklarasikan pada bagian deklarasi global.

```
#include<nama_file_header.h> → preprocessor
directives

tipe_data nama_variabel;     → deklarasi variable
global

tipe_data main()     → fungsi utama
{

      <blok pernyataan>
}
```

Blok

Contoh:

```
1   //preprocessor directive
2       #include <stdio.h> //file header
3   //deklarasi global
4       int TES = 1000;
5   //fungsi utama
6       int main()
7       {
8       printf("Nilai TES= %d", TES);
9       return 0;
10      }
```

#include adalah pengarah preprosesor yang berfungsi menginstruksikan kepada kompiler untuk menyisipkan file lain saat program dikompilasi (yaitu file-file yang berisi stream dan fungsi-fungsi input/output). Biasanya file-file yang disisipkan adalah file-file header, Pustaka input/output merupakan pustaka. Pustaka ini telah ada di header **stdio.h** dan **iostream.h**.
void didepan **main()**, untuk menyatakan bahwa fungsi main() tidak memiliki nilai balik.

main() menjadi awal dan akhir eksekusi program C++, sehingga sebuah program dalam C++ mengandung sebuah fungsi main()
Main →nama judul fungsi
{ →awal tubuh fungsi/awal eksekusi program
→tubuh fungsi/blok
} →akhir tubuh fungsi/akhir eksekusi program
Tanda **()** digunakan untuk mengapit argumen fungsi, yaitu nilai yang akan dilewatkan
ke fungsi.

Blok pernyataan: merupakan satu atau beberapa buah statemen/pernyataan yang pada setiap akhir baris pernyataan diakhiri dengan titik koma (;).

Contoh program:

```
#include <iostream.h>
#include <conio.h>
void main()
{
clrscr(); //membersihkan layar
cout << "Selamat Belajar C++\n";
getch();
}
```

Hasil eksekusi:

Selamat Belajar C++

Mengenal cout

Pengenal **cout** (baca: **c out**) merupakan objek dalam C++ yang digunakan untuk mengarahkan data ke standar output (layar). Atau dapat juga di tuliskan dengan ***printf.*** Tanda << (dua tanda kurang dari berurutan) adalah operator "penyisipan/peletakan" yang akan mengarahkan operand (data) yang terletak di sebelah kanannya ke objek yang terletak di sebelah kirinya.

Pustaka input/output merupakan pustaka yang berisi stream dan fungsi-fungsi input/output. Pustaka ini telah ada di header **stdio.h** dan **iostream.h**.

Dalam bahasa C pustaka I/O yang digunakan adalah **stdio.h** yang berisi fungsi-fungsi seperti **printf** (coutdan **scanf**. Scanf(cin) digunakan untuk memasukkan sebuah nilai ke variable dan printf(cout) digunakan untuk mencetak suatu nilai dari variable maupun konstanta.

Pada contoh di atas

"Selamat Belajar C++\n" diarahkan ke cout, yang memberikan hasil berupa tampilan
string tersebut ke layar. **\ n**, dapat juga dituliskan **<<endl;** adalah perintah untuk pointer/karakter pindah baris (new line).

#include <iostream.h>

#include <iostream.h> menginstruksikan kepada kompiler untuk menyisipkan file iostream.h pada saat program dikompilasi tanpa diakhiri titik koma. File iostream.h perlu disertakan pada program yang melibatkan cout. Tanpa #include <iostream.h> akan terjadi kesalahan saat program dikompilasi. Sebab file iostream.h berisi deklarasi yang diperlukan oleh cout dan berbagai objek yang berhubungan dengan operasi masukan–keluaran.

clrscr();

Pernyataan yang diperlukan untuk menghapus layar. Apabila menggunakan pernyataan ini maka harus disertakan file header conio.h.

Komentar

Komentar diperlukan untuk menjelaskan mengenai program atau bagian-bagian
dalam program. Isi penjelasan berupa:

- Tujuan/fungsi program
- Saat program dibuat/direvisi, bias berupa tanggal dibuatnya program tersebut.
- Keterangan-keterangan lain tentang kegunaan sejumlah pernyataan dalam program.

Tanda awal komentar dalam program C++ ada dua cara :

- Diawali tanda // (dua tanda garis miring/double slash), untuk komentar yang hanya satu baris. Semua tulisan setelah tanda // dianggap sebagai komentar dan tidak akan dieksekusi oleh C++.
- /*mulai blok komentar pada baris ini tidak akan dieksekusi sampai ditemui akhir blok komentar */.

Contoh program :

```
//Program yang mengandung komentar
#include <iostream.h>
#include <conio.h>
void main()
{
clrscr(); //membersihkan layar
```

```
/*mulai blok komentar pada baris ini tidak akan dieksekusi
sampai ditemui akhir blok komentar */
cout << "Selamat Belajar C++\n";
getch();
}
```

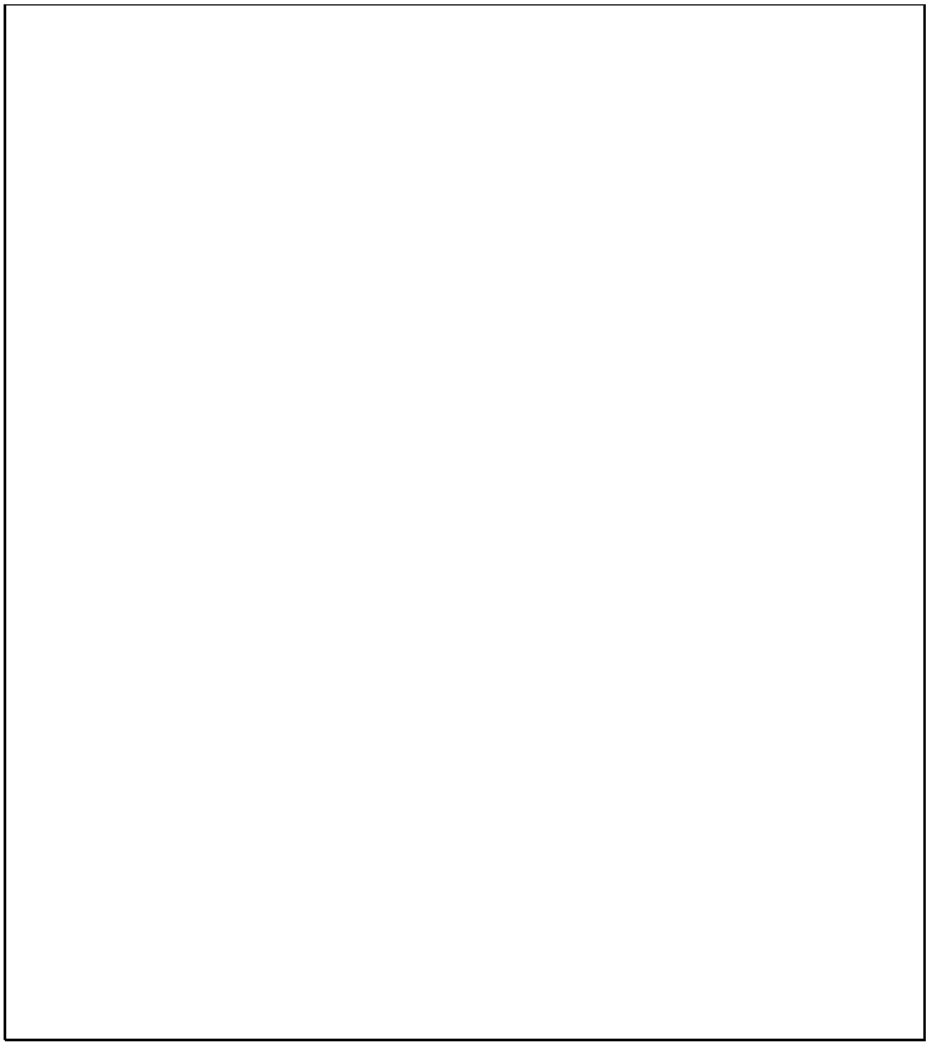
Note :

Sebelum mulai melakukan coding program, sebaiknya diingat bahwa C/C++ bersifat "case sensitive", yang berarti ***huruf besar*** *dan* ***huruf kecil*** *nantinya akan dianggap* ***berbeda*** *oleh program.*

E. Tugas

1. Coba anda buat program tampilan, dengan menampilkan:
   a. Nama: .....
   b. Fakultas: .....
   c. Universitas: .....
   d. Kota: .....
   e. .....

Jawaban:



Bab II

PERNYATAAN, TIPE DATA, VARIABEL & KONSTANTA

A. Pernyataan, Tipe Data, Variabel, dan Konstanta

Pernyataan dalam bahasa C++ merupakan serangkaian atau kelompok rangkaian perintah/simbol standar C++, perintah/simbol yang didefinisikan pemrogram, variabel/ konstanta, dan atau operator yang diakhiri dengan tanda ; (titik koma/*semi-colon*).

```
pernyataan;
```

Variabel pada bahasa C++ adalah bagian dari memori yang hanya dapat menampung satu data/informasi pada satu waktu, dimana data/informasi tersebut dapat berubah setelah dideklarasikan selama pengeksekusian program. Variabel dilambangkan dengan simbol yang didefinisikan oleh pemrogram dan variabel harus dideklarasikan terlebih dahulu sebelum dapat digunakan. Pendeklarasian variabel pada bahasa C++ adalah sebagai berikut :

```
tipe_data nama_variabel;
```

Variabel dapat langsung diberi nilai awal pada saat pendeklarasiannya. Pemberian nilai tersebut dapat dilakukan dengan cara sebagai berikut,

```
tipe_data nama_variabel=nilai_awal;
```

Konstanta merupakan variabel yang data/nilainya tidak dapat diubah setelah dideklarasikan selama pengeksekusian program. Cara mendeklarasikan konstanta adalah sebagai berikut,

```
const tipe_data nama_konstanta=nilai_konstanta;
```

Cara lain untuk mendeklarasikan konstanta adalah dengan mendeklarasikannya sebagai makro, yaitu sebagai berikut,

```
#define nama_makro definisi_makro;
```

Besarnya bagian memori yang digunakan oleh variabel atau konstanta untuk menampung data/informasi tergantung dari tipe data/informasi yang digunakan. Tipe-tipe data standar yang terdefinisikan pada bahasa C++ dapat dilihat dalam tabel berikut :

Tabel : Tipe-tipe data standar pada bahasa C++

Tipe data	Ukuran Presisi	Jangkauan nilai	Jumlah digit presisi
1. char	1 byte	-128 sampai +127	-
2. int	2 byte	-32768 sampai +32767	-
3. long int	4 byte	-2147438648 sampai +2147438647	-
4. float (desimal)	4 byte	$3.4x10^{-38}$ hingga $3.4x10^{+38}$	6-7
5. double	8 byte	$1.7x10^{-308}$ hingga	15-16
6. long double	10 byte	$1.7x10^{+308}$ $3.4x10^{-4932}$hingga $1.1x10^{+4932}$	19

B. Pemodifikasian tipe data unsigned dan signed

Untuk pemodifikasian **tipe unsigned** diterapkan pada data bertipe bilangan bulat (char, short, int dan long). Variable yang ditambah unsigned akan **menyebabkan nilai yang terkandung didalamnya selalu bernilai positif, sehingga jangkauannya akan berubah.**

Untuk pemodifikasian tipe **signed** merupakan default dari tipe data dasar, yang menyatakan data bernilai positif maupun negative. Pemodifikasian tipe data dapat dilihat pada tabel berikut

Tabel 1.2. Tipe-tipe data pemodifikasian pada bahasa C++

1. char/signed char
2. unsigned char
3. int/signed int
4. short int/signed short int
5. unsigned int
6. long int/signed long int
7. unsigned long int

Untuk mendefinisikan suatu tipe data baru selain tipe-tipe data standar pada bahasa C++ digunakan perintah berikut:

```
typedef tipe_data tipe_data_baru;
```

Tabel: Notasi Bahasa C++

Kelompok	Notasi C++	Ket.
1. Tipe Dasar	**Boolean** **Byte** **Shortint** **Word** **Integer** **Longint** **Real** **Double** **Extended** **Char** **String** **string[N]** **Record** field1: **type**; field2: **type**; . . . fieldN: **type**; end;	 1 byte 1 byte 2 byte 2 byte 4 byte
2. Operator a. Aritmatika	 + -	 Penjumlahan Pengurangan

	*	Perkalian
	/	Pembagian
	Div	Hasil bagi
	Mod	Sisa bagi
b. Perbandingan	<	Lebih kecil dari
	<=	Lebih kecil atau sama dengan
	>	Lebih besar dari
	>=	Lebih besar atau sama dengan
	=	Sama dengan
	<>	Tidak sama dengan
c. Logika	**not**	
	and	
	or	
	xor	
3. Komentar	{ komentar }	
	(* komentar *)	
4. Lain-lain	**Const**	
	Type	
	True	
	False	

C. Contoh penggunaan Input Output

1. Contoh penggunaan printf dan scanf

```
#include <stdio.h>
int datal;
int main(void)
{
  printf("nilai awal variabel adalah %d",datal);
  printf("\n sebelum proses memasukkan nilai \n
  masukkan nilai");
  scanf("%d",&datal);

  printf("sesudah proses memasukkan nilai \n");
  printf("maka nilai input menjadi %d\n",datal);
}
```

2. **Contoh penggunaan cout dan cin**

```
#include <iostream.h>
int data1;
int main(void)
{
  cout<<"sebelum      proses      memasukkan      nilai,
  data1=0"<<endl;
  cout<<"sekarang masukkan nilai"<<endl;
  cout<<"nilai yang akan masuk adalah = ";cin>>data1;
  cout<<" "<<endl; //memberikan baris kosong
  cout<<"sesudah proses memasukkan nilai ");
  cout<<"maka nilai data1 menjadi = "<<data1<<endl;
}
```

D. Tugas

1. Menggunakan konstanta, yaitu variabel yang setelah dideklarasikan nilainya tidak dapat berubah/tidak boleh diubah. Untuk mendefinisikan konstanta dicontohkan sebagai berikut : ***const** float pi=3.14159;*

```
1.  #include <iostream.h>
2.  #include <conio.h>
3.  const float phi=3.14159;
4.  int main()
5.  {
6.  Float phi = 3.14;
7.  cout<<"Nilai Phi adalah "<<phi;
8.  cout<<"\nMasukkan nilai phi yang baru \n";
9.  cin>>phi;
10. cout<<"\nNilai phi yang baru adalah "<<phi;
11. cout<<"\n";
12. getch();
13. return 0;
14. }
```

Jelaskan output/keluaran dari kode sumber serta fungsi penggunaan konstanta.

Jawaban anda :

2. Tulislah kode program berikut ke dalam file konstantaPajak.cpp

```
#include <iostream.h>
#include <conio.h>
const float mewah=0.005;
const float menengah=0.002;
float pajak;
int harga;

int main()
{
        cout<<"Masukkan harga barang\n";
        cin>>harga;
        if (30000<harga)
        {
 pajak=harga*mewah;
 cout<<"Harga = "<<harga<<"prosentase pajak = "<<mewah<<"\n";
 cout<<"Pajak pembeliannya adalah "<<pajak;
        }
        else if (100000<harga)
        {
pajak=harga*menengah
  cout<<"Harga = "<<harga<<"prosentase pajak = "<<menengah<<"\n";
  cout<<"Pajak pembeliannya adalah <<pajak;
        }
        else
        {
   cout<<"Harga = "<<harga<<"\n";
   cout<<"Barang murah banget, tidak kena pajak";
        }
        cout<<"\n";
        getch();
        return 0;
}
```

Uji program di atas dan perbaiki kesalahan pada kode sumber tersebut.

Kesalahan pada bagian:

Bab III

OPERATOR DAN SELEKSI

A. Operator

Operator merupakan symbol yang biasa dilibatkan atau dipakai dalam program untuk melakukan sesuatu operasi atau manipulasi. **Operand** adalah objek dari operator. **Operator** dan **operand** akan membentuk ekspresi. Ekspresi ini dapat membentuk suatu pernyataan. Berikut ini penggolongan operator-operator standar yang terdapat dalam bahasa pemrograman C++.

a. Operator Aritmatika

Operator aritmatika yang disediakan oleh C++ antara lain perkalian, pembagian, modulus, penjumlahan, pengurangan, yang berturut turut diwakili oleh symbol *,/,%,+,-.

Contoh program :

```
#include <iostream.h>

Int a=12, b=24, c;
void main()
{
  c=a+b;
  cout<<"a + b = "<<c<<endl;
}
```

Dari contoh program diatas, dari dua variabel yang sudah bernilai, kita dapat mengembngkannya dengan berbagai operasi hitung matematika yang lain, perkalian (a*b), pembagian (a/b) dan lainnya.

b. Operator Penugasan

Operator penugasan yang berupa symbol sama dengan (=) berfungsi untuk memberikan suatu nilai ke suatu variable.

c. Operator Penaikan dan Penurunan

Operator ini berhubungan dengan operator aritmatika. Operator penaikan (increment) dan operator penurunan (decrement) berturut-turut diwakili oleh symbol ++ dan --. Operator ini bisa diletakkan di belakang atau didepan operand.

Contoh program :

```
#include <iostream.h>
 Int a=12;
 void main()
 {  a++;
    cout<<"Nilai dari a++ : "<<a<<endl;
    ++a;
    cout<<"Nilai dari a++ : "<<a<<endl;
 }
```

d. Operator Bitwise (Manipulasi Bit)

Untuk keperluan manipulasi data dalam bentuk bit C++ menyediakan enam buah operator yaitu geser bit kekiri, geser bit kekanan, and, or, xor dan nor yang berturut-turut menggunakan symbol <<, >>, &, |, ^, ~

Contoh program :

```
#include <iostream.h>
  Int a=12, b;
  void main()
  {
  b = a >> 1;
  cout<<"Nilai dari a + b = "<<b<<endl;
  }
```

e. Operator Relasi

Operator relasi digunakan untuk membandingkan dua buah perand/VARIABEL. Operator yang digunakan:

>	lebih dari	>=	lebih dari atau sama dengan
!=	tidak sama dengan	<	kurang dari
==	sama dengan	<=	kurang dari atau sama dengan

Contoh program :

```
#include <iostream.h>
Int a;
void main()
{
  a = 12>3;
  cout<<"Nilai dari 12 > 3 = "<<a<<endl;
  a = 12<3;
  cout<<"Nilai dari 12 > 3 = "<<a<<endl;
}
```

- *Hasil yang akan tampil adalah:*
  12 > 3 -→ 1, *yang berarti* ***benar*** *bahwa 12 lebih besar dari 3.*
  12 3 → 0, *yang berarti* ***salah*** *kalau 12 lebih kecil dari 3.*

f. Operator Logika

Operator logika digunakan untuk menghubungkan dua buah operand menjadi sebuah ungkapan kondisi. Operator yang digunakan: && (and), ||(or), !(not).

Contoh program :

```
#include <iostream.h>
  int a=25, b;
  void main()
  {
    b =(a>25)&&(a<30);
    cout<<"=(a>25)&&(a<30)"<<a<<endl;
  }
```

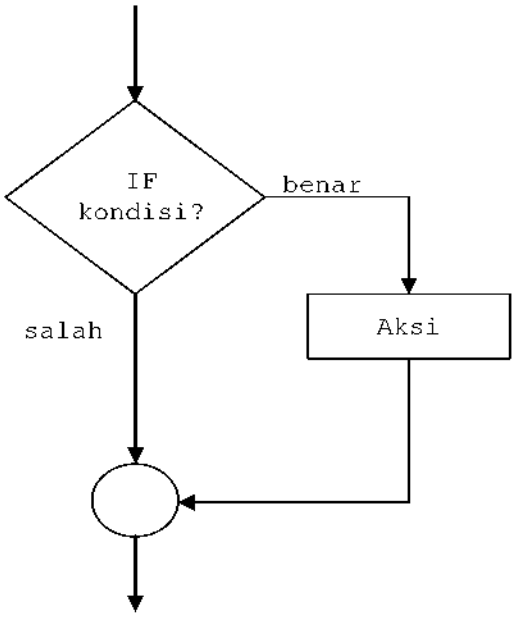
1. Seleksi (Operator Kondisi)

Pada umumnya satu permasalahan yang komplek memerlukan suatu penyelesaian kondisi. Dengan menyeleksi suatu kondisi,

maka selanjutnya dapat ditentukan tindakan apa yang harus dilakukan, tergantung pada hasil kondisi penyeleksian. Jadi suatu aksi hanya dikerjakan apabila persyaratan atau kondisi tertentu terpenuhi.

a. **Penyeleksian Satu Kasus,** dengan menggunakan struktur **if** : Pada penyeleksian satu kasus, *kondisi* akan diseleksi oleh statemen *if*. Bila *kondisi* bernilai benar (*true*), maka *aksi* yang akan dikerjakan dapat dituliskan pada baris pernyataan dalam cout<<"pernyataan";. Bila *kondisi* bernilai salah (*false*), maka tidak ada *aksi* yang akan dikerjakanan dan program selesai.

   Gambar diagram alir penyelesaian satu kasus untuk struktur IF ditunjukkan dalam dan struktur penulisan dalam bahasa C++ dibawah ini.

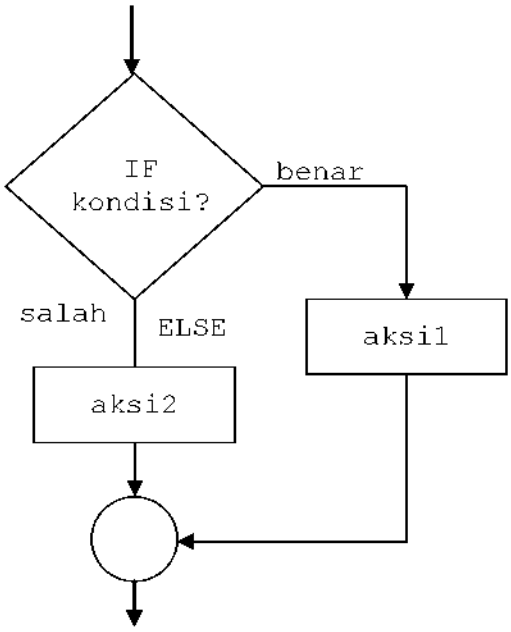


Gambar: Diagram Alir Struktur Penyeleksian Satu Kasus (IF-)

```
      if kondisi
         aksi;

contoh program :
      void main()
      {
        int angka;
        cout<<"angka yang dimasukkan : ";cin>>angka;
        if (angka>0)
            cout<<"Angka anda adalah positif";
}
```

b. **Penyeleksian Dua Kasus,** menggunakan struktur **if - else:**
Dalam struktur **if-else**, *aksi1* akan dilaksanakan jika *kondisi* bernilai benar (*true*), dan jika *kondisi* bernilai salah (*false*) maka *aksi2* yang akan dilaksanakan. Statemen *else* menyatakan ingkaran (*negation*) dari *kondisi*.

Gambar diagram alir penyelesaian dua kasus untuk struktur **if-else** ditunjukkan dalam gambar dan struktur penulisan dalam bahasa C++ dapat dilihat dibawah ini.



Gambar: Diagram Alir Struktur IF-ELSE

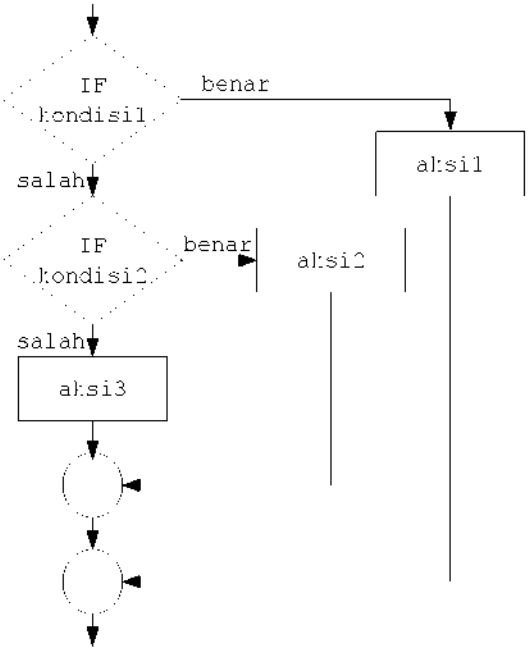
```
      if kondisi then
         aksi1
      else
         aksi2

contoh program :
 #include <iostream.h>
 void main()
 {
 int angka;
 cout<<"angka yang dimasukkan : ";cin>>angka;
 if (angka>0)
      cout<<"Angka anda adalah positif";
 else
      cout<<"Angka anda adalah negatif";
 }
```

Gambar: Struktur Bahasa C Untuk Penyeleksian Dua Kasus (IF-ELSE)

c. Penyeleksian Tiga Kasus atau Lebih (Penyeleksian Tersarang)

Untuk penyeleksian tiga kasus atau lebih juga menggunakan struktur ***if-else*** sebagaimana halnya permasalahan dua kasus.

Gambar diagram alir penyelesaian tiga kasus untuk struktur ***if-else*** dan struktur penulisan dalam bahasa C dapat dilihat dari gambar berikut.



Gambar: Diagram alir dari struktur tiga kasus *if-else if* (tersarang)

```
      if kondisi1 then
        { aksi1 }
      else if (kondisi2)
           {  aksi2   }
      else if (kondisi3)
           {  aksi3   }

         else
           {  aksi..N }
```

contoh program :

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

 void main()
 {
 int nilai;
 cout<<"Nilai Anda : ";cin>>nilai;
 if (nilai>=85)
      { cout<<"NIlai Anda A"; }
 else if (nilai>=66)
      { cout<<"Nilai Anda B"; }
 else if (nilai>=50)
      { cout<<"Nilai Anda C"; }
 else if (nilai>=40)
     { cout<<"Nilai Anda D"; }
 else
      { cout<<"Nilai Anda E"  }
getch();
}
```

Gambar: Struktur Bahasa C untuk Penyeleksian Tiga Kasus IF-ELSE (tersarang)

d. Struktur case. Struktur ini dapat digunakan untuk menganalisa penyeleksian dua kasus atau lebih dan bentuknya adalah lebih sederhana daripada struktur ***if-else*** ataupun bentuk ***if-else*** bersarang.

Jika dalam program, dibutuhkan penyeleksi kondisi dengan banyak alternative (lebih dari dua), biasanya digunakan struktur **case** yang terdapat di dalam pernyataan ***switch*** (nama_variabel_yang_diseleksi. Pernyataan switch dengan ***case*** mempunyai format sebagai berikut:

```
switch (variabel)
{
      case nilai1_variabel : pernyataan_1;break;
      case nilai2_variabel : pernyataan_2;break;
      .
      .
      default : pernyataan_n; break;
}
```

Gambar: Struktur Bahasa C Untuk Seleksi CASE

Pernyataan_1 hanya akan dilaksanakan apabila nilai variabel sama dengan nilai1_variabel. Pernyataan_2 hanya akan dilaksanakan apabila nilai variable sama dengan nilai2_variabel. Pernyataan_n hanya akan dilaksanakan apabila nilai variabel tidak ada yang sesuai dengan nilai-nilai variabel penguji diatasnya.

Jadi :

pernyataan_1 hanya akan dilaksanakan apabila nilai variabel=nilai1_variabel. Pernyataan_2 hanya akan dilaksanakan apabila nilai variable=nilai2_variabel. Pernyataan_n hanya akan dilaksanakan apabila nilai variabel tidak ada yang sesuai dengan nilai-nilai variabel penguji diatasnya.

Contoh program :

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

void main()
{
  int golongan, status;
  float gaji_pokok, tunjangan, total;
  cout<<"Golongan Anda [1,2,3,4]    : ";cin>>golongan;
  cout<<"Status [0=tidak kawin,1=kawin]: ";cin>>status;
```

4. Buatlah program menggunakan kondisi untuk menghitung criteria berat badan menggunakan Indeks Massa Tubuh (IMT), yang bisa dihitung dengan :

   IMT = b/t^2

   b=berat badan (kg)

   t=tinggi badan (m)

   Criteria IMT sebagai berikut :

IMT <= 18.5	Kurus
18.5<IMT<= 25	Normal
25<IMT<= 30	Gemuk
IMT>30	Kegemukan

   Contoh Output:

   Berat badan (kg) = 45

   Tinggi badan (m) = 1.72

   Nilai IMT anda 17.0, anda termasuk criteria kurus

NB: program yang dibuat tidak boleh sama antara anak yang satu dan yang lain serta tidak boleh sama dengan contoh diatas. Setelah praktikum tugas wajib dikumpulkan ke ketua kelas.

Jawaban :

Bab IV

PERULANGAN (LOOPING)

A. Teori Dasar

Terdapat dua model struktur pengulangan, yaitu:

1. Struktur Pengulangan Tanpa Kondisi (*unconditional looping*).
   Di dalam struktur ini, instruksi-instruksi di dalam badan pengulangan diulangi sejumlah kali yang dispesifikasikan (jumlah pengulangan sudah diketahui sebelum eksekusi). Contoh: Struktur ***for***.
2. Struktur Pengulangan Dengan Kondisi (*conditional looping*).
   Di dalam struktur ini, jumlah pengulangan tidak diketahui sebelum eksekusi program. Yang dapat ditentukan hanya kondisi berhenti pengulangan, artinya instruksi-instruksi di dalam badan pengulangan diulangi sampai kondisi berhenti terpenuhi. Contoh: Struktur ***while*** dan Struktur ***repeat***.

B. Macam-macam struktur pengulangan:

1. Perulangan For

Struktur pengulangan ***for*** digunakan untuk mengulang statemen atau satu blok statemen berulang kali. Jumlah pengulangan diketahui atau dapat ditentukan sebelum eksekusi. Untuk mencacah sudah jumlah pengulangan diperlukan sebuah variabel pencacah (*counter*). Variabel ini nilainya selalu bertambah satu setiap kali pengulangan dilakukan. Jika cacah pengulangan sudah mencapai jumlah yang dispesifikasikan, maka proses pengulangan berhenti. Pada struktur ***for***, *pencacah* haruslah dari tipe data yang memiliki predecessor dan successor, yaitu integer atau karakter. Tipe riil tidak dapat digunakan sebagai *pencacah*. *Aksi* adalah satu atau lebih instruksi yang diulang.

Pernyataan for memiliki format sebagai berikut:

```
for (ungkapan1; ungkapan2; ungkapan3)
      pernyataan;
```

Keterangan:

Ungkapan1 : nilai awal suatu perulangan iterasi

Ungkapan2 : kondisi syarat agar perulangan atau iterasi tetap berlangsung

Ungkapan3 : umumnya kontrol perulangan atau iterasi

Note : Perulangan for akan mengeksekusi pernyataan selama bagian pernyataan sesuai dengan ungkapan yang ada pada ungkapan2.

Contoh program :

```
#include <iostream.h>

main()
{
 int i;

 for (i=1; i<=5; i++)
  cout << i << ". Belajar C++" << endl;

  return 0;

}
```

<u>For bersarang (nested loop)</u>

Pada aplikasi tertentu, terkadang kita menggunakan pernyataan for yang berada di dalam pernyataan for. Salah satu contohnya adalah apabila kita ingin menghasilkan tampilan sebagai berikut:*

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

Contoh program :

```
#include <iostream.h>

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

    cout << "tinggi segitiga = ";
    cin >> tinggi;

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

2. Perulangan while

Perulangan dengan while merupakan perulangan yang **memerlukan dan menggunakan** syarat awal. Berikut ini adalah deklarasi perulangan while.

```
while (kondisi)
      Pernyataan;
```

Bagian pernyataan juga dapat berupa pernyataan majemuk, bentuk deklarasinya adalah sebagai berikut:

```
while (kondisi)
{
      Pernyataan_1;
      Pernyataan_2;
      . . . .
      . . . .
      Pernyataan_n;
}
```

```
#include <iostream.h>

main()
{
int n;
float x, total, rata;

total =0;
n = 0      ;
//x = 1;

while (x!=0)
{
   n++;
   cout << "Data ke-" << n <<" : ";
   cin >> x;
   total = total + x;
}

n--;
rata = total /n;

cout << "Ada" << n <<" data " <<endl;
cout << "Total = " << total <<endl;
cout << "Rata-rata = " << rata ;

return 0;
}
```

```
Data ke-1 : 4
Data ke-2 : 5
Data ke-3 : 6
Data ke-4 : 7
Data ke-5 : 8
Data ke-6 : 9
Data ke-7 : 0
Ada6 data
Total = 39
Rata-rata = 6.5
```

3. Perulangan do....while

Sama seperti halnya while, pernyataan Do-While digunakan untuk mengeksekusi blok secara berulang sampai tidak memenuhi kondisi tertentu, yang berarti pernyataan akan dijalankan berulang-ulang sampai kondisi bernilai salah.

```
do
{
      Pernyataan_1;
      Pernyataan_2;
      . . . .
      Pernyataan_n;
}
while (kondisi);
```

Contoh program :

```
#include <iostream.h>
#include <conio.h>
main()
{
  char nama[80];
  int tombol, cacah = 0;
  float nilai, jumlah = 0, rata2;
    cout << "Menghitung rata-rata nilai\n";
    cout << "Masukkan nilai, "
    << "isikan negatif jika selesai\n\n";

    do {
    cacah++;
    cout << "Data ke-" << cacah << " = ";
    cin >> nilai;
    jumlah = jumlah+nilai;
    } while (nilai >= 0);
        rata = jumlah/cacah;
        cout << "\nBanyaknya data = " << cacah;
        cout << "\nJumlah = " << jumlah;
        cout << "\nrata2 = " << rata2;
    }
```

Tugas

1. Tulislah kode program berikut ke dalam file Faktorial.cpp

```
#include <iostream.h>
#include <conio.h>
//Program penghitung faktorial

int main()
{
//kamus
  int   i = 1;
  int hasilfaktorial=1;
  int faktorial;
//Algoritma
   cout<<"Masukkan nilai faktorial\n";
   cin>>faktorial;
  while (i<=faktorial)
  {
        hasilfaktorial=hasilfaktorial*i;
        i=i+1;
  }
   cout<<"Hasilnya adalah <<hasilfaktorial;
   getch();
   return 0;
}
```

Uji program di atas dan perbaiki kesalahan pada kode sumber tersebut.

Kesalahan pada bagian:

Koreksi yang seharusnya:

2. Konversikan kode sumber tersebut ke dalam bentuk perulangan for dan lakukan pengujian.

Kode sumber:

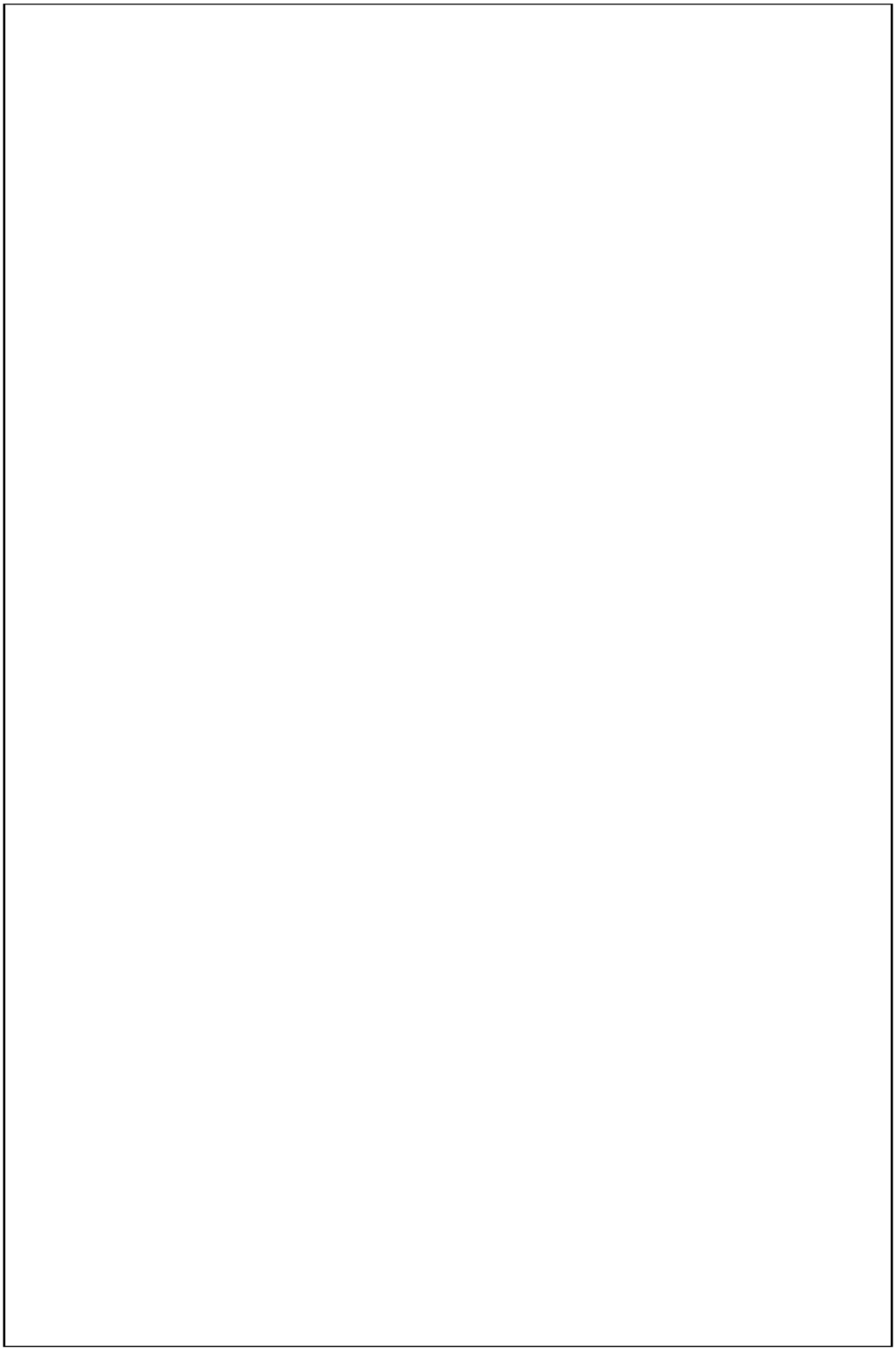
Hasil pengujian program:

C. Tugas

1. Salinlah contoh program while diatas.
2. Coba kerjakan contoh program while tersebut menggunakan do...while.
3. Gunakan for bersarang untuk menghasilkan output:

```
N = 4
1 * * * * * *
  2 2 * * * *
    3 3 3 * *
      4 4 4 4
```

Jawaban



Bab V

ARRAY (LARIK)

A. Dasar Teori

Larik adalah tipe terstruktur yang terdiri dari sejumlah elemen-elemen yang bertipe sama. Banyaknya elemen dalam suatu larik ditunjukkan oleh suatu indeks yang harus merupakan tipe data yang menyatakan keterurutan, misalnya integer atau karakter (karena ia menyatakan posisi data). Tiap-tiap elemen di larik dapat diakses langsung melalui indeksnya. Suatu larik memiliki jumlah elemen yang jumlahnya tetap, sehingga jumlah elemen larik tidak dapat diubah selama pelaksanaan program.

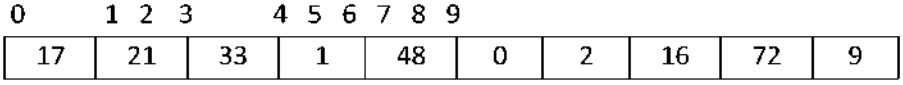
Larik dapat berupa larik satu dimensi ataupun lebih dari satu dimensi. Matriks merupakan contoh larik yang memiliki dimensi lebih dari satu. Struktur penyimpanan matriks di dalam memori utama yang setiap individu elemennya diakses dengan menggunakan dua buah indeks (yang biasanya dikonotasikan dengan baris dan kolom). Karena matriks sebenarnya adalah larik, konsep umum dari larik juga berlaku untuk matriks, yaitu:

- Kumpulan elemen yang bertipe sama, dapat dapat berupa tipe dasar (*integer*, *real*, *boolean*, *char*, dan *string*), atau tipe terstruktur seperti *record*.
- Setiap elemen data dapat diakses secara langsung jika indeksnya (baris dan kolom) diketahui.
- Merupakan struktur data yang statik, artinya jumlah elemennya sudah dideklarasikan terlebih dahulu di dalam bagian DEKLARASI dan tidak bisa diubah selama pelaksanaan program

Elemen matriks diakses melalui indeks baris dan indeks kolomnya. Jika indeks baris dinyatakan dengan i dan indeks kolom dinyatakan dengan j, maka notasi algoritmik untuk mengakses elemen pada baris i dan kolom j adalah nama_matriks[i,j].

B. Array Satu Dimensi

Yaitu kumpulan elemen-elemen identik yang tersusun dalam satu baris. Elemen-elemen tersebut memiliki tipe data yang sama, tetapi isi dari elemen itu boleh berbeda. Berikut ini adalah konsep array satu dimensi:



Bentuk umum :
tipeData NamaArray [JumlahElemen] = {<Inisialisasi>};
int Bola [10] = { 17, 21, 33, dst};

Contoh:
unsigned int TinggiBadan [100];//deklarasi array
bool Hadir [5] = {true, true, false, false};//pemberian nilai array pada inisialisasi

Pendeklarasian array diawali dengan penyebutan **tipe data diikuti nama variabel array, diikuti jumlah elemen**. Jika array hendak diberi nilai awal, nilai-nilai awal dituliskan dalam inisialisasi. Jika inisialisasi kurang dari jumlah elemen array, sisanya akan diinisialisasikan dengan nol.

Elemen array diakses dengan menyebutkan indeks posisi elemen tersebut dalam array. Elemen pertama array memiliki indeks 0.

unsigned int TinggiBadan [100] //tinggi badan memiliki 100 elemen
TinggiBadan [0] = 7; //mengeset elemen pertama TinggiBadan
TinggiBadan [5] = 16; //mengeset elemen keenam
temp = TinggiBadan[99]; //mengopi elemen terakhir ke variabel temp

Contoh program 1

```
#include <iostream.h>
int billy [] = {16, 2, 77, 40, 12071};
int n, result=0;

int main ()
{
  for ( n=0 ; n<5 ; n++ )
  {
    result += billy[n];
  }
  cout << result;
  return 0;
}
```

Result = result + billy[n]

C. Array Dua Dimensi

Sering digambarkan sebagai sebuah matriks dan merupakan perluasan dari sebuah array satu dimensi. Jika array satu dimensi hanya terdiri dari sebuah baris dengan beberapa kolom elemen maka array dua dimensi terdiri dari beberapa baris dan beberapa kolom elemen yang bertipe sama sehingga dapat digambarkan sebagai berikut:

0	1	2	3	4	5	6	7	8	9
0	17	21	33	1	48	0	2	16	72	9
1	87	65	4	6	7	23	81	10		
2										
3										

Int Matriks[4][10]

Bentuk umum :

tipeData NamaArray [JumlahBaris][JumlahKolom] =

```
{
      {<InisialisasiBaris1>},
      {<InisialisasiBaris2>},
      ….
      {<InisialisasiBarisN>},
};
```

Contoh:

double Matrix [4][4];// pemesanan tempat array

bool Papan [2][2] = { {true, false}, {false, true} };

Contoh program 2

Output yang diinginkan :

Tabel 1 : Data Kelulusan

Jurusan	1992	1993	1994	1995
1. Teknik Informatika	35	45	80	120
2. Manajemen Informatika	100	110	70	101
3. Teknik Komputer	10	15	20	17

```
#include <iostream.h>
void main()
{
    int data_lulus[3][4]; //array berdimensi dua
    int tahun, jurusan;
 //memberikan data elemen ke array
    data_lulus[0][0] = 35;          //data TI - 1992
    data_lulus[0][1] = 45;          //data TI - 1993
    data_lulus[0][2] = 90;          //data TI - 1994
    data_lulus[0][3] = 120;         //data TI - 1995
    data_lulus[1][0] = 100;         //data MI - 1992
    data_lulus[1][1] = 110;         //data MI - 1993
    data_lulus[1][2] = 70;          //data MI - 1994
    data_lulus[1][3] = 101;         //data MI - 1995
    data_lulus[2][0] = 10;          //data TK - 1992
    data_lulus[2][1] = 15;          //data TK - 1993
    data_lulus[2][2] = 20;          //data TK - 1994
    data_lulus[2][3] = 17;          //data TK - 1995
    //proses untuk memperoleh informasi kelulusan
    while(1)
    {
        cout << "jurusan (0 = TI, 1 = MI, 2 = TK):  ";
        cin >> jurusan;
        if((jurusan == 0) || (jurusan == 1) || (jurusan == 2))
            break;                  //keluar dari while
    }
    while(1)
    {   cout << "Tahun (1992 - 1995) : ";
        cin >> tahun;
        if ((tahun >= 1992) && (tahun <= 1995))
        {
            tahun -= 1992; //konversi ke 0, 1, 2, atau 3
            break;    //keluar dari while
    }   }
    cout << "jumlah yang lulus = "
         << data_lulus[jurusan][tahun] << endl;
}
```

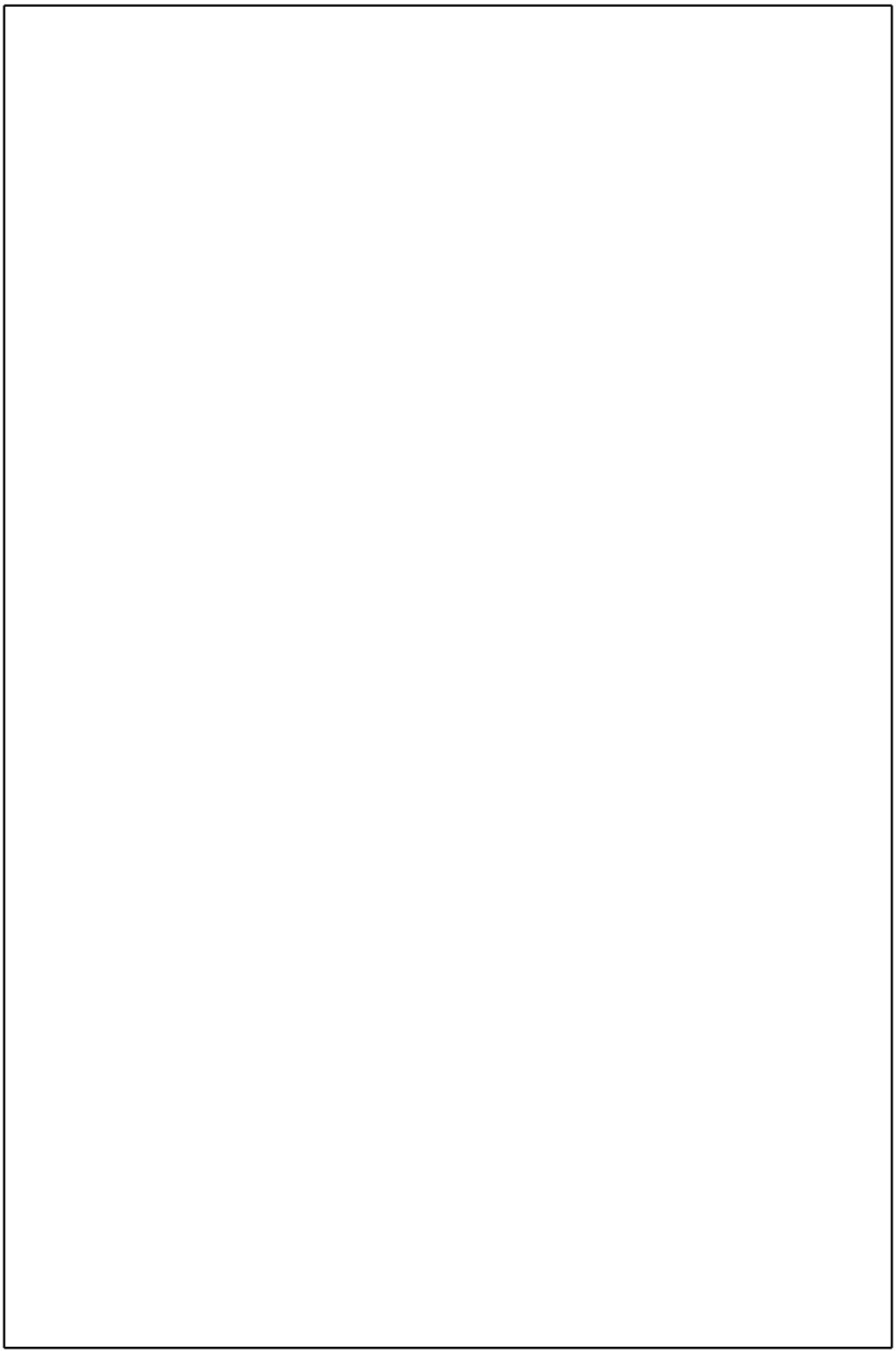
D. Tugas

1. Salinlah program 1.
2. Pada program 1 pengisian array melalui inisialisasi, buatlah program seperti program 1 tetapi pengisian array melalui fungsi utama/pengesetan array.
3. Buatlah program menggunakan for yang menghasilkan output:

   A[1, 1] = 1
   A[1, 2] = 2
   A[2, 1] = 3
   A[2, 2] = 4
   B[1, 1] = 5
   B[1, 2] = 6
   B[2, 1] = 7
   B[2, 2] = 8

   subject: tugas5/kelas/nama

Jawaban



Bab VI

METHOD, FUNGSI & PROSEDUR

A. Method

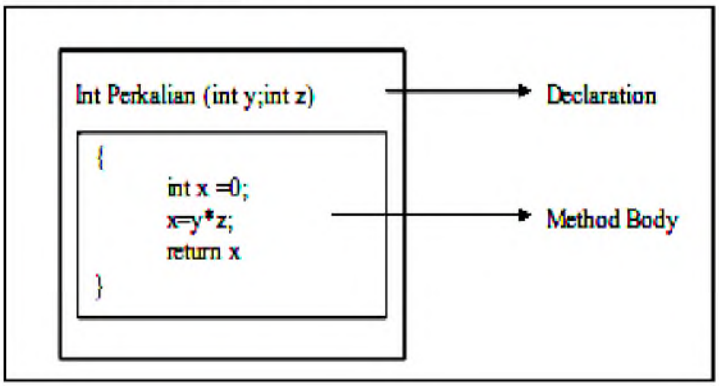
Method adalah fungsi atau prosedur yang dibuat oleh seorang programmer di dalam suatu unit/class. Method dapat dibagi menjadi fungsi dan prosedur. Fungsi adalah bagian atau sub dari program yang mempunyai algoritma tertentu dalam menyelesaikan suatu masalah dengan mengembalikan hasil. Prosedur adalah bagian atau sub dari program yang mempunyai algoritma tertentu dalam menyelesaikan suatu masalah tanpa mengembalikan suatu nilai hasil. Secara umum method dalam C adalah sebuah fungsi.

Deklarasi sebuah method

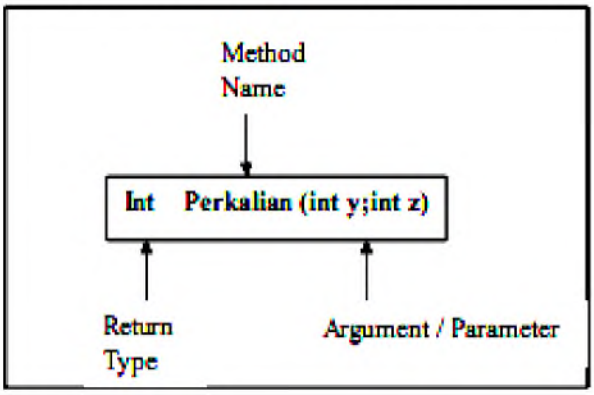
Method terdiri atas dua bagian yakni:

1. Method declaration
2. Method body

- *Contoh sebuah method:*



- *Penggambaran deklarasi method adalah sebagai berikut:*



Return type

Setiap fungsi menghasilkan suatu nilai dengan tipe data tertentu. Apabila terdapat suatu fungsi yang tidak menghasilkan suatu nilai apapun maka bagian return type ini diganti dengan void. Berikut adalah contoh fungsi yang tidak menghasilkan apapun:

```
void CetakLayar()
 {
      printf("Program Sarjana Teknik Informatika");
}
```

Method Name

Setiap fungsi dalam pemrograman mempunyai identitas nama fungsi. Penamaan fungsi / method mengikuti aturan:

a. Tidak menggunakan spasi atau menggunakan spasi dengan diganti "_" (misal method Hitung_Integral).
b. Tidak menggunakan kata – kata yang telah dipakai oleh compiler (reserved words).

Parameter

Bagian parameter diisi denga parameter – parameter fungsi yang diperlukan. Parameter apabila lebih dari satu akan dipisahkan dengan tanda koma (","). Parameter dapat dibagi menjadi 2:

a. Parameter Aktual, adalah nilai yang dimasukkan ke dalam pemanggilan fungsi atau prosedur.
b. Parameter Formal, adalah deklarasi variable yang digunakan sebagai parameter pada suatu fungsi atau prosedur.

B. Fungsi

Apabila kita membuat program yang terlalu panjang, membuat kita akan kesulitan membaca dan memahami program. Sehingga kita perlu memecah program tersebut manjadi beberapa bagian (modul) yang memudahkan kita untuk mencari kesalahan program, memperbaiki, dan membuat dokumentasi program. Untuk membuat modul tersebut C++ menyediakan fungsi.

Fungsi berguna untuk mengumpulkan statemen yang dapat dijalankan menjadi satu dalam suatu blok dan menjalankannya kembali hanya dengan menuliskan nama fungsi yang menampungnya. Fungsi juga dapat dipakai untuk menampung baris-baris perintah yang sering dipakai dalam sebuah program.

Deklarasi fungsi dapat dilakukan dengan dua cara yaitu:

- Header fungsi tanpa parameter atau fungsi tanpa return value
- Header fungsi dengan parameter atau fungsi dengan return value

1. *Header fungsi tanpa parameter atau fungsi tanpa return value*

   Bentuk Umum :

   Tipedata namaFungsi

   Contoh :

   Int JumlahIsi()
   double Random()
   void Clear()

   Blok fungsi tanpa parameter :

   Tipedata namaFungsi ()
   {
   .....
   }

Contoh program 1 :

```
#include <iostream.h>

int a,b;
int tambah()
{
 return a+b;
};

int kali()
{
 return a*b;
};

int main()
{
 cout<<"masukkan nilai a = ";cin>>a;
 cout<<"masukkan nilai b = ";cin>>b;

 cout<<"hasil a+b = "<<tambah<<endl;
 cout<<"hasil a*b = "<<kali<<endl;
 return 0;
}
```

2. Header fungsi dengan parameter atau fungsi tanpa return value

Penulisan header fungsi dengan parameter hampir sama dengan fungsi tanpa parameter, yaitu diawali dengan tipe data, lalu nama fungsi, dan diikuti dengan parameter-parameter (yang berada di dalam tanda kurung) yang masing-masing dipisahkan dengan koma beserta dengan tipenya.

Bentuk Umum :

Tipedata namaFungsi (<daftar_parameter>)

Contoh :

Int Jumlahkan(int a, int b)

double ArcCos(double x)

Blok fungsi tanpa parameter

Tipedata namaFungsi (parameter)

{

.....

}

Contoh program 2 :

```
#include <iostream.h>
int tambah(int a, int b)
{
    return a+b;
};
int kali(int a, int b)
{
    return a*b;
};
int main()
{
    cout<<"masukkan nilai a = ";cin>>a;
    cout<<"masukkan nilai b = ";cin>>b;

    cout<<"hasil a+b = "<<tambah(a,b)<<endl;
    cout<<"hasil a*b = "<<kali(a,b)<<endl;
    return 0;
}
```

C. Tugas :

1. Buatlah file cobaFungsi.cpp lalu jalankan, amati, dan jelaskan.

```
#include <iostream.h>
#include <conio.h>
//Program untuk menampilkan hasil
//Perhitungan segitiga menggunakan prosedur

//Kamus
int Alas, Tinggi;
```

```
void hitungSegitiga (int A, int T) {
      float hasil;
      hasil=0.5*A*T;
   cout<<"Luas Segitiga : "<<hasil<<"\n";
}

//Algoritma
void main() {
      Alas=5;
      Tinggi=6;
   cout<<"Alas : "<<Alas<<"\nTinggi : "<<Tinggi<<"\n";
      hitungSegitiga(Alas,Tinggi);
   getch();
}
```

Tampilkan keluaran program:

Jelaskan bagaimana jalannya prosedur serta parameter – parameter yang terlibat:

2. Buatlah file cobaFungsi.cpp lalu jalankan, amati, dan jelaskan.

```
#include <iostream.h>
#include <conio.h>
//Program untuk menampilkan hasil
//Perhitungan segitiga menggunakan fungsi

int Alas, Tinggi;
float hitungSegitiga (int A, int T)
{
   float hasil;
   hasil=0.5*A*T;
   return hasil;
}
//Algoritma
void main() {
   Alas=5;
   Tinggi=6;
   cout<<"Alas : "<<Alas<<"\nTinggi : "<<Tinggi<<"\n";
   cout<<"Luas Segitiga: "<<hitungSegitiga(Alas,Tinggi)<<"\n";
   getch();
}
```

Tuliskan hasil keluaran program:

Jelaskan bagaimana jalannya Fungsi serta parameter – parameter yang terlibat:

Jelaskan perbedaan fungsi dan prosedur dari contoh kode di atas.

Tugas

1. Buatlah fungsi tanpa parameter menghitung luas lingkaran dan volume tabung.
2. Buatlah fungsi dengan parameter menghitung luas lingkaran dan volume tabung.

   Output:

   Menu

   1. Hitung Luas Lingkaran
   2. Hitung Volume tabung

   Pilihan Anda = 1
   Menghitung Luas Lingkaran
   Masukkan phi = 3.14
   Masukkan jari-jari lingkaran = 10
   Luas lingkaran = 314
   atau
   Pilihan Anda = 2
   Menghitung Volume Tabung
   Masukkan phi = 3.14
   Masukkan jari-jari tabung = 10
   Masukkan tinggi tabung = 10
   Luas lingkaran = 3140

Jawaban :

Bab VII

POINTER & REFERENCE

A. Dasar Teori

Pointer adalah suatu variabel yang fungsinya menyimpan alamat (address). Jadi pointer merupakan fasilitas untuk mengakses suatu variabel dengan memakai address. Pada bab sebelumnya kita sudah mengenal variabel, bedanya variabel hanya berisi nilai, dan bukan alamat. Jika kita mempunyai sebuah variabel dengan tipe data tertentu, maka untuk mendapatkan alamat dari variabel tersebut adalah dengan menggunakan opetaror &. Alamat inilah yang kemudian akan disimpan kedalam variabel yang bertipe pointer.

```
tipe data
*nama_pointer;
```

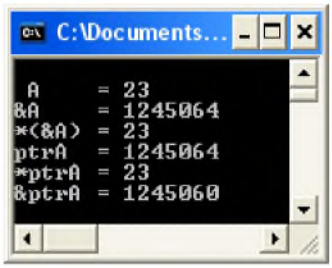
Untuk mendeklarasikan variabel sebagai sebuah pointer, kita harus menambahkan tanda asterisk (*) di depan nama variabel.

```
int A;        //deklarasi variabel biasa
int *ptrA;    //deklarasi variabel pointer
```

Tipe data diatas berguna untuk menyatakan bahwa pointer yang kita deklarasikan akan ditempati oleh data dengan tipe tertentu. Misal kita punya variabel X dengan tipe long, dan punya pointer P dengn tipe long. Kita dapat memerintahkan pointer P untuk menunjuk ke alamat yang ditempati oleh variabel X.

```
long X;          //deklarasi variabel X dengan tipe long
long *P;         //pointer P
P=&X;            //memerintahkan pointer P untuk menunjuk alamat
                   dari variabel X.
```

```
// Contoh program, Nama File : pointer.cpp
#include <stdio.h>
#include <conio.h>

void main()
{
  int A;//deklarasi var biasa
  int *ptrA; //deklarasi var pointer
  clrscr();
  A=23;
  ptrA=&A;
  printf("\n A     = %d",A); //isi dari var A
  printf("\n&A     = %u",&A); //alamat memory var A
  printf("\n*(&A)  = %u",*(&A)); //isi di alamat pada var A
  printf("\nptrA   = %u",ptrA);//isi dari var pointer ptrA
  printf("\n*ptrA  = %d",*ptrA);//isi di alamat pada var ptrA
  printf("\n&ptrA  = %u",&ptrA);//alamat memory var ptrA
  getch();
}
```



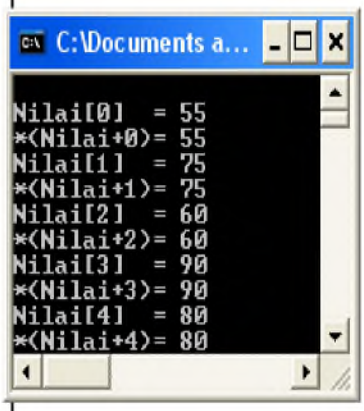
B. Kegunaan Pointer

- Calling by reference.
- Mengembalikan lebih satu nilai dari satu fungsi.
- Mengirimkan array dan string dari suatu fungsi ke fungsi lain.
- Memudahkan dalam manipulasi array.
- Manipulasi memory dalam system.
- Membuat struktur data dinamis seperti linklist, tree, graph dan sebagainya.

- Nama array adalah alamat memory (pointer) pertama dari elemen array tersebut. Jika A adalah array berdimensi satu maka alamat memory dari elemen pertama adalah &A[0] atau A saja. Sehingga &A[i] sama dengan (A+i). Maka A[i] pun sama dengan *(A+i)

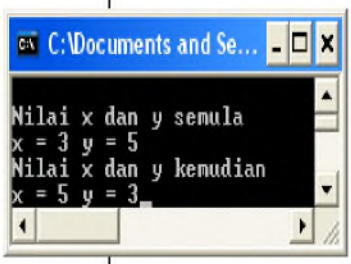
- Berikut contoh program tentang bagaimana mengakses data array menggunakan konsep pointer:

```
#include <stdio.h>
#include <conio.h>
void main()
{
 int Nilai[5]={55,75,60,90,80};
 clrscr();
 for(int i=0;i<5;i++)
  {
   //akses dengan indeks array
   printf("\nNilai[%d]  =
%d",i,Nilai[i]);
   //akses dengan pointer
   printf("\n*(Nilai+%d)=
%u",i,*(Nilai+i));
  }
  getch ();
}
```



- Contoh program penerapan pointer untuk argumen fungsi call by reference:

```
/*Nama File : tukar2.c
  Prinsip Call by reference*/
#include <stdio.h>
#include <conio.h>

void tukar(int *a,int *b)
{ int temp;
   temp=*a;
   *a=*b;
   *b=temp;}

void main()
{ clrscr();
  int x=3, y=5;
  printf("\nNilai x dan y semula");
  printf("\nx = %d y = %d",x,y);
  tukar(&x,&y);
  printf("\nNilai x dan y kemudian");
  printf("\nx = %d y = %d",x,y);
  getch();}
```



Bab VIII

STRUCTURE

A. Dasar Teori

Structure/struktur adalah kumpulan elemen-elemen data yang digabungkan menjadi satu kesatuan. Masing-masing elemen data dikenal dengan sebutan field/kolom. Field data tersebut dapat memiliki tipe data yang sama ataupun berbeda. Walaupun field-field tersebut berada dalam satu kesatuan, masing-masing field tersebut dapat diakses secara individual.

Field-field tersebut digabungkan menjadi satu dengan tujuan untuk kemudahan dalam operasinya. Misalnya kita ingin mencatat data-data mahasiswa dan pelajar dalam sebuah program, untuk membedakannya dapat membuat sebuah **record mahasiswa yang terdiri dari field-field NIM, nama, alamat, dan IPK** serta sebuah **record pelajar yang terdiri dari field-field Nama, NomorUrut, alamat dan JumlahNilai.** Dengan demikian akan lebih mudah membedakan keduanya.

B. Deklarasi struktur

Pendeklarasian structure selalu diawali oleh kata baku **struct** diikuti nama structure dan deklarasi field-field di antara kurung kurawal.

Bentuk Umum:

```
struct namaStruct
{ tipeData1 Field1;
  tipeData2 Field2;
  ........
  tipeDataN FieldN;};
```

Contoh:

```
struct Mahasiswa
{ char NIM[10];
  char Nama[20];
  char Alamat[10];
  float IPK; };
```

Contoh program :

```
#include <iostream.h>

struct Mahasiswa
{
    char Nama[51];
    char NIM[15];
    char Alamat[51];
    float IPK;
};

int main()
{
    Mahasiswa Mhs;

    cout << "Nama   : ";
    cin.getline (Mhs.Nama, 51);
    cout << "NIM    : ";
    cin.getline (Mhs.NIM, 15);
    cout << "Alamat    : ";
    cin.getline (Mhs.Alamat, 51);
    cout << "IPK    : ";
    cin >> Mhs.IPK;

    cout << endl;

    cout << "Nama Anda    : " << Mhs.Nama << endl;
    cout << "NIM Anda : " << Mhs.NIM << endl;
    cout << "Alamat Anda : " << Mhs.Alamat << endl;
    cout << "IPK Anda : " << Mhs.IPK << endl;

    return 0;
}
```

```
Nama     : Lucky
NIM      : 0410960048
Alamat   : Jl. Pisang 16 Mojokerto
IPK      : 3.39

Nama Anda        : Lucky
NIM Anda         : 0410960048
Alamat Anda      : Jl. Pisang 16 Mojokerto
IPK Anda         : 3.39
```

C. Tugas:

1. Salinlah contoh program diatas
2. Buatlah program sehingga memberikan output sebagai berikut :

Form data karyawan

==========

Nama : Agus Hermanto

Gaji : 2000000

Status (0 belum menikah, 1 sudah, 2 janda): 1

Jumlah anak : 3

==========

Masukkan data lagi? (Y/T) : y

==========

Nama : Lia Ilawan

Gaji : 1500000

Status (0 belum menikah, 1 sudah, 2 janda): 2

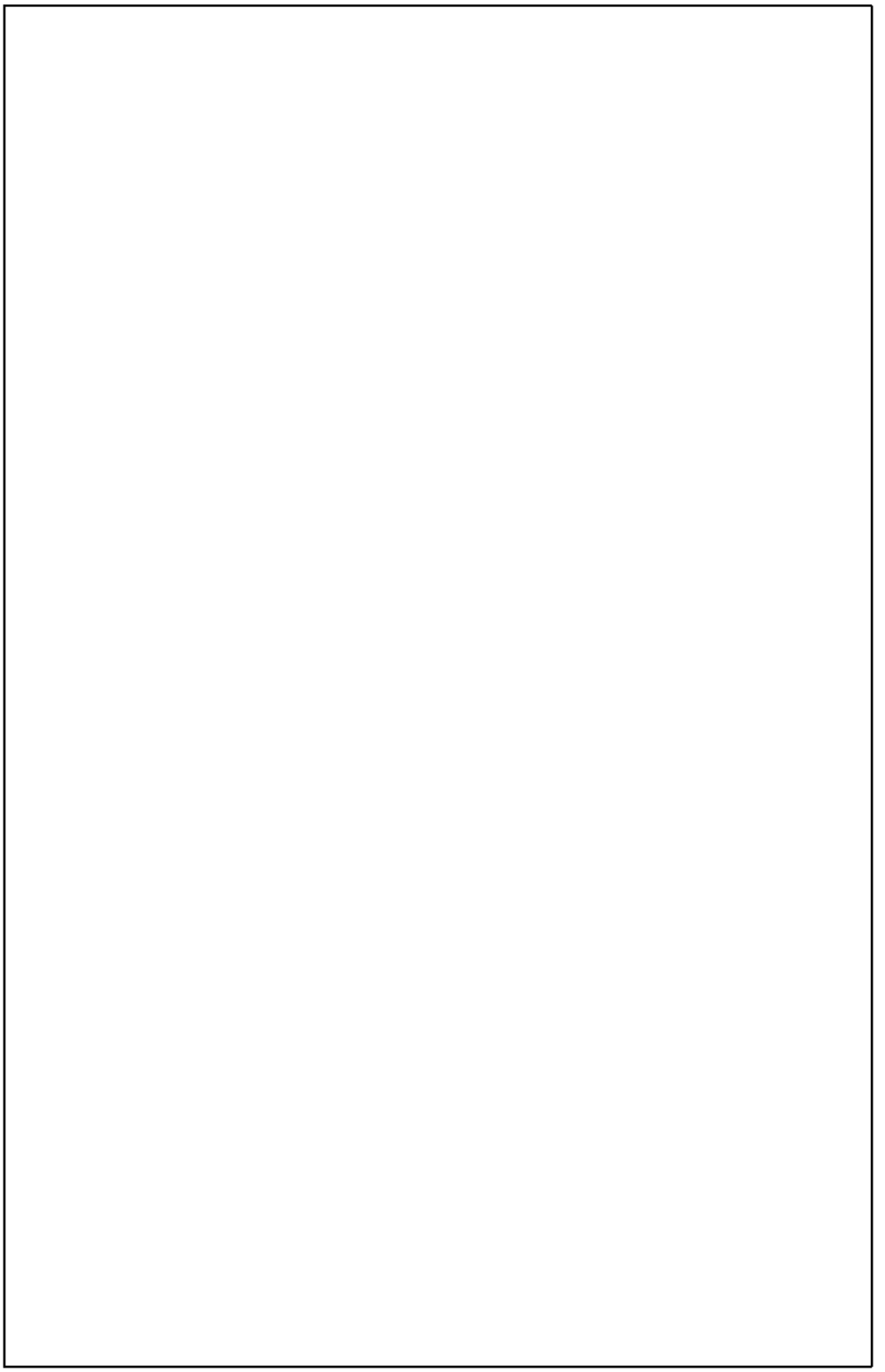
Jumlah anak : 2

==========

Masukkan data lagi? (Y/T) : t

==========

Jawaban :



Bab IX

STRING DAN CHARACTER

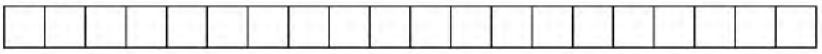
A. Dasar Teori

Pada C++ tidak ada tipe variable elemen yang spesific untuk menyimpan string. Untuk keperluan ini dapat digunakan array dengan tipe **char**, dimana berisi elemen dengan tipe **char**. Perlu di ingat bahwa tipe **char** digunakan untuk menyimpan 1 karakter, karena itu array dari char digunakan untuk menyimpan string.

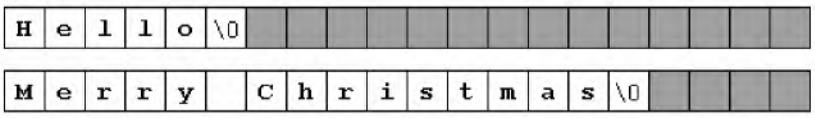
Contoh:

```
char jenny [20];
```

Dapat menyimpan sampai dengan 20 karakter:



Penyimpanan karakter-nya dapat direpresentasikan seperti dibawah ini:



Perhatikan, karakter NULL (**'\0'**) selalu disertakan diakhir string untuk indikasi akhir dari string.

B. Inisialisasi string

Sama halnya seperti array-array sebelumnya, inisialisasi pada string sbb:

```
char mystring[] = { 'H', 'e', 'l', 'l', 'o', '\0' };
```

Contoh diatas, merupakan inisialisasi 6 buah elemen bertipe **char,** yaitu **Hello** dan karakter null **'\0'**. Untuk menentukan nilai konstan, pada string digunakan tanda kutip ganda ("), sedangkan untuk karakter kutip tunggal ('). String yang diapit oleh kutip ganda sudah mengandung karakter Null pada akhir string, contoh:

```
char mystring [] = { 'H', 'e', 'l', 'l', 'o', '\0' };
char mystring [] = "Hello"; //
```

Contoh diatas merupakan deklarasi array **mystring** yang berisi 6 elemen.

Pemberian nilai pada string

Sama halnya seperti pemberian nilai pada array-array sebelumnya, untuk array dengan tipe char dapat dituliskan :

```
mystring[0] = 'H';
mystring[1] = 'e';
mystring[2] = 'l';
mystring[3] = 'l';
mystring[4] = 'o';
mystring[5] = '\0';
```

C. Fungsi-Fungsi dalam String (cstring.h)

1. Fungsi strcpy

Cara pemberian nilai seperti contoh diatas sangat tidak praktis. Umumnya untuk pemberian nilai pada array bertipe char digunakan fungsi **strcpy**. **strcpy** (**string copy**) mendefinisikan **cstring** (**string.h**) library dan dapat dipanggil dengan cara:

```
strcpy (string1, string2);
```

Baris perintah diatas menyebabkan isi dari*string2* di-*copy* ke *string1*. *string2* dapat berupa array, pointer, atau konstanta string.

Contoh:

// setting value to string

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

int main ()
{
  char szMyName [20];
  strcpy (szMyName,"J. Soulie");
  cout << szMyName;
  return 0;
}
```

Output:

J. Soulie

Perhatikan, header**<cstring.h>** harus disertakan agar bisa menggunakan fungsi **strcpy**. Bisa juga menggunakan fungsi sederhana seperti misalkan diberi nama setstring, dengan operasi yang sama seperti **strcpy**.

Contoh : *// setting value to string*

```
#include <iostream.h>

void setstring (char szOut [], char szIn [])
{
  int n=0;
  do {
    szOut[n] = szIn[n];
  } while (szIn[n++] != '\0');
}

int main ()
{
  char szMyName [20];
  setstring (szMyName,"J. Soulie");
  cout << szMyName;
  return 0;
}
```

Output :

J. Soulie

Metode lain yang dapat digunakan untuk inisialisasi nilai yaitu input stream (**cin**). Dalam kasus ini, nilai string ditentukan oleh user saat eksekusi program. Ketika menggunakan **cin,** biasanya digunakan metode **getline**, Pemanggilannya sebagai berikut:

```
cin.getline (char buffer[], int length, char delimiter
= ' \n');
```

dimana, ***buffer*** adalah alamat untuk menyimpan input, ***length*** adalah maksimum panjang buffer,dan ***delimiter*** adalah karakter yang digunakan untuk menentukan input akhir, dengan default–atau dengan (**'\n'**).

Contoh:

// cin with strings

```
#include <iostream.h>

int main ()
{
  char mybuffer [100];
  cout << "What's your name? ";
  cin.getline (mybuffer,100);
  cout << "Hello " << mybuffer << ".\n";
  cout << "Which is your favourite team? ";
  cin.getline (mybuffer,100);
  cout << "I like " << mybuffer << " too.\n";
  return 0;
}
```

Output :

What's your name? Juan
Hello Juan.
Which is your favourite team? Inter Milan
I like Inter Milan too.

Perhatikan kedua pemanggilan **cin.getline**, menggunakan identifier yang sama (**mybuffer**). Sama halnya seperti penggunaan operator extraction, sehingga dapat dituliskan:

cin >> mybuffer;

Instruksi diatas dapat berjalan, hanya saja mempunyai keterbatasan bila dibandingkan dengan **cin.getline**, diantaranya:

- Dapat menerima 1 kata saja (bukan kalimat lengkap).
- Tidak diperkenankan untuk memberikan ukuran buffer. Akan menyebabkan program tidak stabil jika user meng-input lebih besar dari kapasitas array yang ada.

2. Fungsi Strlen

Merupakan fungsi manipulasi string yang digunakan untuk menghitung panjang string. Sehingga fungsi ini memberikan nilai balik panjang string.

Sintaks:

```
size_t strlen (const
char* string);
```

Contoh cara pengoperasiannya, yaitu:

Panjang = strlen(bunga);

3. Fungsi Strcat

Merupakan fungsi yang digunakan untuk menggabungkan string. Fungsi ini memberikan nilai balik string yang sudah digabung.

Sintaks:

```
char* strcat (char* dest,
const char* src);
```

Contoh cara pengoperasiannya, yaitu:

Strcat(str1, str2)

Strcat("aku", "belajar")

4. Fungsi strcmp

Merupakan fungsi yang digunakan untuk membandingkan dua buah string yaitu string1 dan string2.

Sintaks:

strcmp: int strcmp (const char* *string1*, **const char*** *string2*);

Contoh:

Strcmp (str1, str2)

Strcmp("aku", "aku")= 0

5. Strrev

Merupakan string yang digunakan untuk membalik string.

Contoh:

Strrev(str);

Str →kamu

Strrev(str)

Str → umak

Cttn : **char*** sama dengan **char[]**

D. Konversi String Ke Tipe Lainnya

String dapat berisi data dengan tipe lain seperti angka. Contoh **"1977"**. **cstdlib** (**stdlib.h**) library menyediakan 3 fungsi yang dapat menangani hal tersebut:

- **atoi:** converts string to **int** type.
- **atol:** converts string to **long** type.
- **atof:** converts string to **float** type.

Fungsi-fungsi ini menerima 1 parameter dan mengembalikan nilainya kedalam tipe yang diminta (int, long or float). Fungsi ini dikombinasikan dengan metode **getline** pada **cin.**

Contoh:

// cin and atau functions*

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

int main ()
{
  char mybuffer [100];
  float price;
  int quantity;
  cout << "Enter price: ";
  cin.getline (mybuffer,100);
  price = atof (mybuffer);
  cout << "Enter quantity: ";
  cin.getline (mybuffer,100);
  quantity = atoi (mybuffer);
  cout << "Total price: " << price*quantity;
  return 0;
}
```

Output :

Enter price: 2.75

Enter quantity: 21

Total price: 57.75

E. Tugas

1. Salinlah contoh program dibawah ini. Jalankan dan perhatikan hasilnya.

```
//contoh program penggunaan fungsi string dengan
getline. Datagetline.cpp
#include <conio.h>
#include <cstring.h>

int main()
{
string nama,kota,telp,alamat;
 cout<<"Nama Mahasiswa    : "; getline(cin,nama);
 cout<<"Kota Mahasiswa    : "; getline(cin,kota);
 cout<<"Alamat            : "; getline(cin,alamat);
 cout<<"Nomer Telepon     : "; getline(cin,telp);

 clrscr();
 cout<<" "<<endl;
 cout<<"----------------------------------- "<<endl;
 cout<<"Daftar Nama Mahasiswa   "<<endl;
 cout<<"----------------------------------- "<<endl;
 cout<<"   Nama     : "<<nama<<endl;
 cout<<"   Dari Kota      : "<<kota<<endl;
 cout<<"   Alamat   : "<<alamat<<endl;
 cout<<"   Telepon : "<<telp<<endl;
 cout<<"----------------------------------- "<<endl;
getch();
}
```

2. Simpanlah program dengan nama lain, misal **Datapribadi.cpp.**
   Ubahlah program anda dengan ***tanpa*** menggunakan header **<cstring.h>** , gantilah tipe variable yang semula **string** dengan **char,** hilangkan penggunaan **getline** gunakan struktur **cin** biasanya. Jalankan program dan amati hasilnya.

3. Pada program Datapribadi.cpp, isikan data isian dengan lebih dari 2 suku kata. Misalkan nama mahasiswa: Yudha Bimantara Setyawan, lanjutkan untuk isian berikutnya. Amati hasilnya. Apa yang terjadi? Mengapa demikian?

Jawaban:

Bab X

SEARCHING

A. Dasar Teori

Dalam Pencarian Binary Search, Data yang ada harus diurutkan terlebih dahulu berdasarkan suatu urutan tertentu yang dijadikan kunci pencarian. Adalah teknik pencarian data dalam dengan cara membagi data menjadi dua bagian setiap kali terjadi proses pencarian. Prinsip pencarian biner adalah:

Data diambil dari posisi 1 sampai posisi akhir N Kemudian cari posisi data tengah dengan rumus: (posisi awal + posisi akhir) / 2. Kemudian data yang dicari dibandingkan dengan data yang di tengah, apakah sama atau lebih kecil, atau lebih besar? Jika lebih besar, maka proses pencarian dicari dengan posisi awal adalah posisi tengah + 1 Jika lebih kecil, maka proses pencarian dicari dengan posisi akhir adalah posisi tengah – 1 Jika data sama, berarti ketemu.

Contoh Data:
Misalnya data yang dicari 17

0	1	2	3	4	5	6	7	8
3	9	11	12	15	17	23	31	35
A				B				C

Karena 17 > 15 (data tengah), maka: awal = tengah + 1

0	1	2	3	4	5	6	7	8
3	9	11	12	15	17	23	31	35
					A	B		C

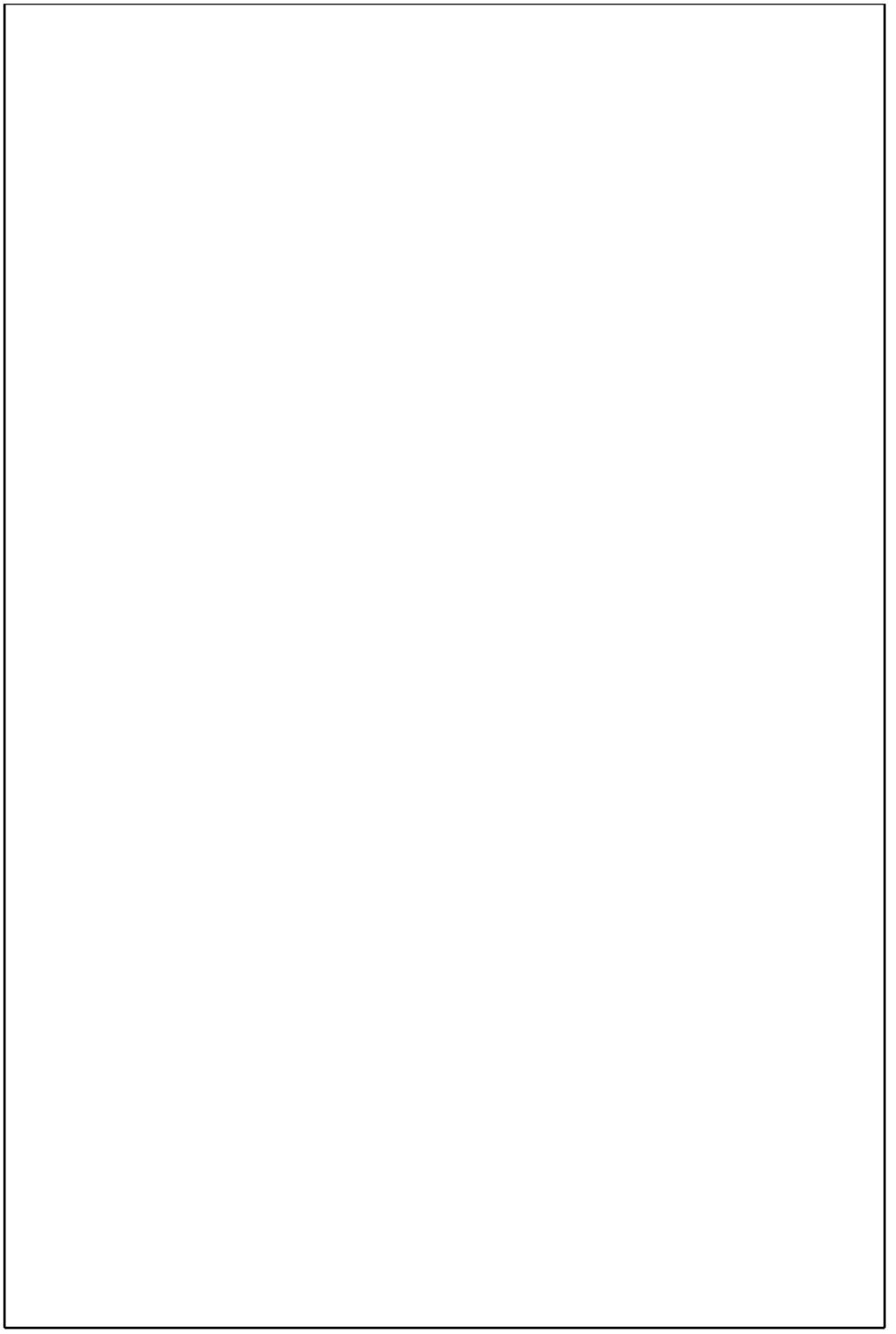
Karena 17 < 23 (data tengah), maka: akhir = tengah – 1

0	1	2	3	4	5	6	7	8
3	9	11	12	15	17	23	31	35
					A=B=C			

Karena 17 = 17 (data tengah), maka KETEMU!

Contoh program binary search :

```
   //binary searching,
//program bisa jalan jika data sudah terurut
#include <iostream.h>
#include <conio.h>

int data[10]={1,3,4,7,12,25,40,65,78,90};
int binary_search(int cari)

{
int l,r,m;
int n=10;
l=0;
r=n-1;
int ketemu=0;
while (l<=r && ketemu==0)
{
m=(l+r)/2;
if (data[m]==cari)
ketemu=1;
else
if(cari<data[m])
r=m-1;
else l=m+1;
}
if(ketemu==1) return 1; else return 0;
}

void main()
{
clrscr();
int cari, hasil;
cout<<"masukan data yang ingin di cari= ";
cin>>cari;
hasil = binary_search(cari);
if(hasil==1)
{
cout<<"data ada!"<<endl;
}
else
if(hasil==0)
cout<<"data tidak ada!"<<endl;
getch();
}
```



Bab XI

SORTING

A. Dasar Teori

Sorting adalah proses menyusun kembali data yang sebelumnya telah disusun dengan suatu pola tertentu ataupun secara acak, sehingga menjadi tersusun secara teratur menurut aturan tertentu.

Pada umumnya ada 2 macam pengurutan, yaitu:

- ✓ Pengurutan secara ascending (urut naik).
- ✓ Pengurutan secara descending (urut turun).

Algoritma untuk melakukan sorting juga ada berbagai macam, antara lain:

- Teoretis : Computational complexity theory, Big O notation, Total order, Stability, Comparison sort.
- Exchange sorts : **Exchange sort**, **Bubble sort**, Cocktail sort, Comb sort, Gnome sort, **Quick sort**.
- Selection sorts : **Selection sort**, Heap sort, Smooth sort.
- Insertion sorts : **Insertion sort**, Shell sort, Tree sort, Library sort, Patience sorting.

B. Sorting Algorithm

- **Bubble Sort**
  - Metode sorting paling mudah, namun paling lambat dibandingkan dengan yang lain.

- Bubble sort mengurutkan data dengan cara membandingkan elemen sekarang dengan elemen berikutnya.
- Bisa dilakukan baik dari kepala array maupun ekor array.
- Proses yang berlangsung, jika:
  - ✓ Ascending: jika elemen sekarang lebih besar daripada elemen berikutnya, maka kedua elemen tersebut ditukar.
  - ✓ Descending: jika elemen sekarang lebih kecil daripada elemen berikutnya, maka kedua elemen tersebut ditukar.
- Hal ini akan terlihat seperti penggeseran angka, perbandingan, kemudian jika memenuhi syarat kemudian tukar.
- Proses penukaran ini akan terus dilakukan hingga seluruh array telah diperiksa.
- Contoh fungsi bubble sort:

```
//Bubble Sort
void bubble (int a[], int n)
{
int i,j;
for (i=n;i>=1;i--)
{
  for (j=2;j<i;j++)
  if(a[j-1]>a[j])
  tukar (a,j-1,j);
  }
}
```

- **Selection Sort**
  - Kombinasi sorting dan searching.
  - Untuk setiap proses, akan dilakukan dengan mencari elemen dari posisi yang belum diurutkan dan kemudian memilih elemen yang memiliki nilai terkecil atau terbesar yang akan ditukarkan ke posisi yang tepat di dalam array.
  - Misalnya untuk putaran pertama, akan dicari data dengan nilai terkecil dan data ini akan ditempatkan pada indeks terkecil, pada putaran kedua akan dicari data kedua terkecil, dan akan ditempatkan di indeks kedua, negitu seterusnya hingga tidak ada data yang dicari lagi.

- Selama proses, pembandingan dan pengubahan hanya dilakukan pada indeks pembanding saja, pertukaran data secara fisik terjadi pada akhir proses.

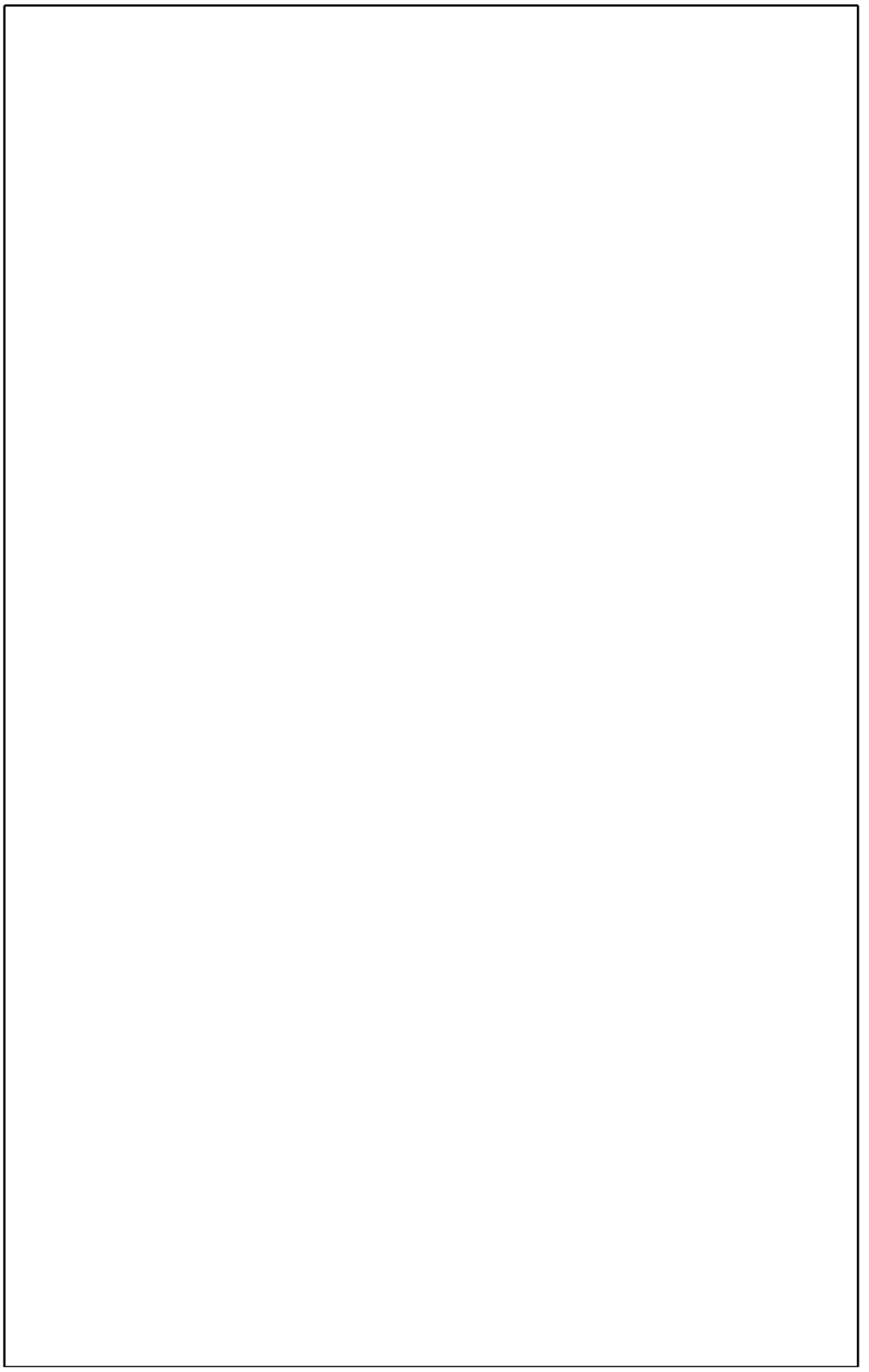
- ***Contoh fungsi selection sort:***

```
//Selection Sort
void selection (int a[],int n) {
int i,j,pos;
for (i=1;i<n;i++)
{
   pos=i;
   for (j=i+1;j<=n;j++)
   if(a[j]<a[pos])
   pos=j;
   tukar(a,pos,i);
   }
}
```

C. Tugas

Buat program dengan inputan NIM (empat digit terakhir), nama, dan fakultas untuk beberapa mahasiswa, kemudian lakukan sorting terhadap inputan berdasarkan NIMnya!

Jawaban :

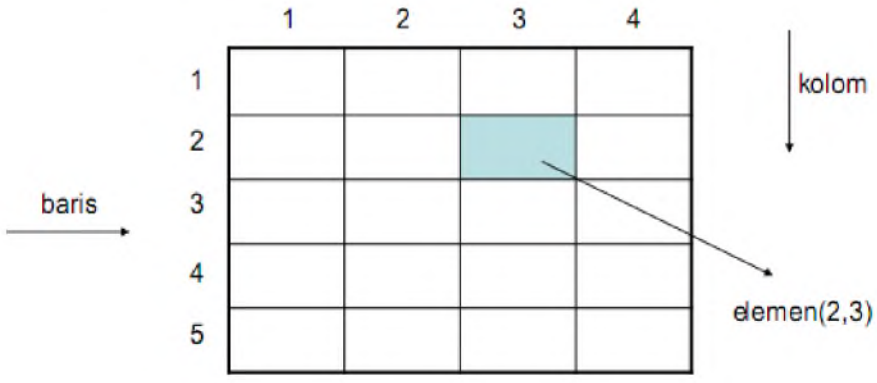


Bab XII

MATRIK

A. Dasar Teori

- Matrik Adalah struktur penyimpanan data di dalam memori utama.
- Setiap individu elemennya diacu dengan menggunakan dua buah indeks (baris dan kolom).
- Merupakan sebuah array yang memiliki dimensi lebih dari satu
- Strukturnya praktis dan mudah digunakan, jika indeksnya diketahui (baris dan kolom), maka akan diketahui pula posisi relatif elemen di dalam kumpulannya.
- Gambar dibawah menunjukan sebuah matriks yang terdiri dari 5 buah baris dan 4 buah kolom.
- Elemen yang diarsir menyatakan elemen matrik untuk baris 2 dan kolom 4.



- Jika matriks pada gambar sebelumnya adalah M, maka elemen yang diarsir adalah M[2,3].
- Cara mengacu dengan peubah indeks hanya benar apabila nilai dari peubah indeks tersebut sudah terdefinisi.
- Misalkan:

M[p,2], M[1,k] , M[p,q]

Hanya benar dengan syarat p, k, q sudah terdefinisi

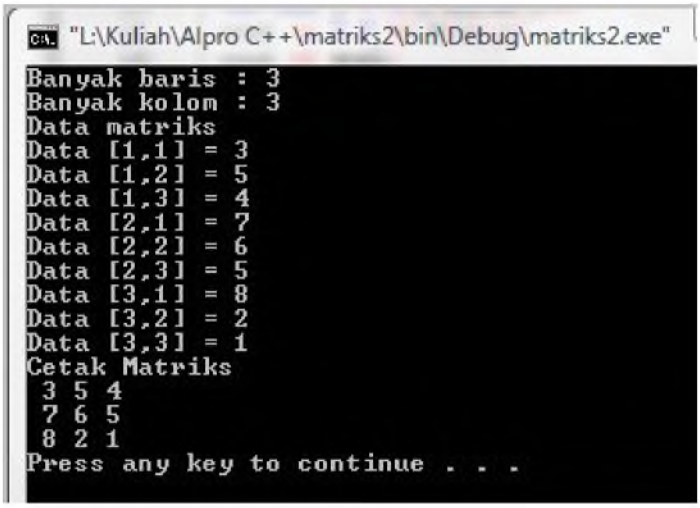
Elemen Matriks

	1	2	3	4
1	M[1,1]	M[1,2]	M[1,3]	M[1,4]
2	M[2,1]	M[2,2]	M[2,3]	M[2,4]
3	M[3,1]	M[3,2]	M[3,3]	M[3,4]
4	M[4,1]	M[4,2]	M[4,3]	M[4,4]
5	M[5,1]	M[5,2]	M[5,3]	M[5,4]

B. Matriks di Memori

- Representasi matriks di dalam memori tetap sebagai deretan sel berurutan
- Contoh representasi matriks 4x4 di memori

- Program Mencetak Matriks dengan bahasa C++ ini, ini dia tampilannya ;



- Kemudian dibawah ini adalah source code/codingnya :

```
#include <iostream>

using namespace std;

void baca_matriks(int mat[10][10], int baris, int kolom)
{
  int i, j;
  for( i =0;i<baris; i++)
  for(j=0;j<kolom;j++)
  {
    cout << "Data [" << (i+1) << "," << (j+1) << "] = ";
    cin >> mat[i][j];
  }
}

void cetak_matriks(const int A[10][10], int baris, int kolom)
{
  for(int i = 0;i<baris;i++)
  {
    for(int j = 0;j<kolom;j++)
    cout << " " << A[i][j] ;
    cout << endl;
  }
}

int main()
{
int m, n;
int matriks[10][10];
int jumlah[10][10];
```

```
cout << "Banyak baris : " ;
cin >> m;
cout << "Banyak kolom : ";
cin >> n;
cout << "Data matriks \n";
baca_matriks(matriks,m,n);
cout << "Cetak Matriks \n";
cetak_matriks(matriks,m,n);
system("PAUSE");
return 0;

}
```

C. Tugas

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} + \begin{bmatrix} 10 & 11 \\ 12 & 13 \\ 14 & 15 \end{bmatrix} =$$

Buat algoritma dengan pseoducode untuk penjumlahan matriks diatas!
Dan diterapkan dalam pemrograman!

Jawaban :

Tentang Penulis

Rina Firliana, memperoleh gelar Sarjana Komputer (S.Kom), Jurusan Teknik Informatika – STT Cahaya Surya Kediri, lulus tahun 2002. Memperoleh gelar Magister Komputer (M.Kom) Program Pasca SarjanaMagister Teknologi Informasi Sekolah Tinggi Teknologi Surabaya, lulus tahun 2012. Saat ini menjadi Dosen di Universitas Nusantara PGRI Kediri.

Patmi Kasih, memperoleh gelar Sarjana Komputer (S.Kom), Jurusan Teknik Informatika – STT Cahaya Surya Kediri, lulus tahun 2002. Memperoleh gelar Magister Komputer (M.Kom) Program Pasca Sarjana Magister Teknologi Informasi Sekolah Tinggi Teknologi Surabaya, lulus tahun 2014. Saat ini menjadi Dosen di Universitas Nusantara PGRI Kediri.

Algoritma & Pemrograman C++

Buku ini cocok untuk anda yang hendak mempelajari penyusunan algoritma dan pemrograman komputer. Algoritma merupakan dasar untuk menyelesaikan permasalahan yang akan dilaksanakan dengan menggunakan komputer, sebelum permasalahan tersebut ditulis ke dalam bahasa pemrograman komputer. Anda yang belum pernah belajar bahasa pemrograman C++ akan dapat mempelajari buku ini tanpa mengalami kesulitan.

Dengan mengikuti pembahasan, contoh program dan juga eksekusinya, anda akan memperoleh kemampuan dalam pembuatan program dengan cepat dan mudah.

Materi-materi yang dibahas:

- **Algoritma dan pemrograman:** membahas tetang pengertian algoritma serta pemrograman.
- **Pernyataan, tipe data, variabel dan konstanta:** Memahami pengertian dan penggunaan pernyataan, tipe data, variabel dan konstanta dalam bahasa C++ dan memahami penggunaan tipe data standar pada bahasa C++ serta memahami cara pendefinisian tipe data baru pada bahasa C++.
- **Operator dan seleksi:** Memahami penggunaan operator unary dan binary pada C++ dan memahami prioritas dan urutan pengeksekusian operator pada C++.
- **Perulangan (looping): Memahami penggunaan jenis perulangan terstruktur yang ada pada bahasa pemrograman C++.**
- **Larik (array):** Memahami penggunaan array (larik/deret) dalam C++ dan mengerti memahami variabel, tipe dan bentuk array dalam Bahasa C++ dan Dapat menggunakan fungsi array dalam pemrograman C++.
- **Method, fungsi dan prosedur:** Mengenal dan memahami fungsi procedure dalam Bahasa C++ dan mengerti seta memahami tentang deklarasi,return, parameter dan bentuk sekaligus dapat menggunakan dalam pemrograman C++.
- **Pointer dan reference:** Dapat menggunakan Pointer, dapat menggunakan Pointer dengan argument fungsi call by reference dan memahami penggunaan pointer ke fungsi.
- **Structure:** Memahami struktur structure dan dapat membuat pengolahan data dengan structure.
- **String dan character:** Memahami penggunaan string dan character dalam C++, serta mengetahui penggunaan fungsi-fungsi untuk manipulasi string.
- **Searching:** Mengerti dan memahami binary search pada bahasa C++ dengan menggunakan binary search dalam pemrograman.
- **Sorting:** Mengerti dan memahami tentang pengurutan dalam bahasa C++, dapat menggunakan sorting bubble sort dan selection short dalam pemrograman C++.
- **Matrik:** Mengenal dan memahami matriks dalam Bahasa C++ dan dapat menggunakan matriks dalam pemrograman C++.

ISBN 978-602-5605-25-3

9 786025 605253

adjie media nusantara
Jl.Demang Palang 9 Watudandang, Nganjuk
Telp.(0358) 792436 / Hunting: 082244863077
email: penerbit@adjiemedianusantara.co.id
website: www.adjiemedianusantara.co.id