KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Alhamdullilah, atas berkat rahmat Allah Yang Maha Kuasa dengan didorongkan oleh keinginan luhur memperluas wawasan dalam pengembangan pengetahuan tentang Algoritma dan Struktur Data I, maka buku petunjuk (modul) praktikum ini disusun/dibuat.

Buku petunjuk (modul) Praktikum Algoritma dan Struktur Data I ini dibuat untuk membantu jalannya Praktikum Algoritma dan Struktur Data I prodi S1 Ilmu Komputer. Buku petunjuk (modul) ini dibuat sedemikian rupa sehingga dapat dengan mudah dipahami dan dipelajari oleh mereka yang belum pernah mengenal Algoritma dan Struktur Data sekalipun, dan bagi mereka yang pernah mengenal membaca buku ini akan menyegarkan ingatan.

Terima kasih kami ucapkan kepada semua pihak yang telah membantu dan mendukung pembuatan buku ini.

Wassalamu'alaikum Wr. Wb.

LABORATORIUM KOMPUTER DASAR
JIKE F.MIPA UGM

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Tujuan Pembelajaran

a. Mahasiswa mengenal definisi algoritma dan pemrograman
b. Mahasiswa mengenal konsep dasar pemetaan algoritma ke dalam bahasa algoritmik.
c. Mahasiswa mampu menyelesaikan kasus atau solusi terhadap suatu masalah yang diberikan.

1.2 Pengenalan Algoritma dan Pemrograman

a. Latar Belakang

Algoritma berarti solusi, solusi yang dimaksud dalam bahasa pemrograman adalah pemecahan masalah yang harus dipecahkan dengan menggunakan komputer. Langkah-langkah apa saja yang dibutuhkan untuk menyelesaikan masalah yang dihadapi. Oleh karena algoritma merupakan inti dari suatu pemrograman, maka algoritma harus dibuat runut agar komputer mengerti dan mampu mengeksekusi program yang dibuat secara benar.

b. Teori

Contoh *real* untuk mengambarkan solusi dari masalah yang akan diselelesaikan, misalnya solusi untuk menghitung jumlah hewan di suatu peternakan:

1. Hitung keseluruhan jumlah ayam yang ada di dalam kandang
2. Apabila ada hewan yang masuk setelah itu, maka jumlah ayam akan ditambah
3. Apabila ada hewan yang diambil untuk penyembelihan, maka jumlah ayam akan dikurangi.
4. Hewan yang ada di peternakan terdiri dari ayam, sapi, kambing, dan bebek

Sebagai manusia, kita sudah pasti mengerti langkah-langkah dalam menghitung jumlah ayam dalam suatu peternakan, namun masalah dalam pemrograman adalah bagaimana kita

bisa membuat komputer mengerti langkah demi langkah yang kita inginkan sehingga menghasilkan hasil yang kita inginkan. Agar dapat dijalankan pada komputer, maka langkah-langkah solusi yang diinginkan harus menggunakan bahasa yang dimengerti oleh komputer yang dikemas dalam bentuk program komputer.

Bahasa algoritmik *(pseudo-code)* adalah sebuah bahasa penengah antara manusia dan komputer. *Pseudo-code* dibuat untuk memudahkan algoritma dengan logika manusia diubah menjadi bahasa pemrograman apapun yang dimengerti oleh komputer. Terdapat banyak sekali bahasa pemrograman yang dikenal oleh komputer, misalnya Pascal, Java, PHP, C#, C++, dan sebagainya.

Program adalah algoritma ditambah dengan struktur data. Struktur data adalah tempat tatanan penyimpanan data yang dibutuhkan program pada komputer. Jika manusia memiliki otak untuk menyimpan data, maka komputer juga membutuhkan tempat untuk menyimpan data yang dibutuhkan. Hal ini disebabkan tempat penyimpanan data pada komputer memiliki kemampuan yang terbatas jika dibandingkan dengan otak manusia, maka diperlukan sebuah tatanan atau struktur agar data yang disimpan mudah untuk diakses.

Belajar pemrograman berarti belajar membuat strategi penyelesaian masalah atau membuat suatu solusi. Sedangkan bahasa pemrograman adalah alat untuk mempelajari pembuatan program. Dalam praktikum ini, kita akan belajar membuat program dengan menggunakan bahasa pemrograman C++.

c. Implementasi

Dengan menggunakan contoh sebelumnya. Kita dapat membuat urutan dalam bentuk algoritma yang baik dan terstruktur.

Task 1 : Deklarasi, merupakan tahapan untuk mendeklarasikan tempat yang dipakai untuk membuat mi.

Step 1 : Mendeklarasikan tempat kosong yang dipakai sebagai kandang tempat menaruh hewan.

Step 2 : Mendeklarasikan kandang di peternakan.

Step 3 : Contoh kode dengan bahasa algoritmik pada tahap deklarasi adalah sebagai berikut :

ayam : integer

sapi : integer

kambing : integer

bebek : integer

Task 2 : Inisialisasi, merupakan tahapan mempersiapkan proses yang dikerjakan untuk menyelesaikan masalah.

Step 1 : Mempersiapkan jumlah hewan yang ada di kandang, dan proses penambahan dan pengambilan hewan.

Step 2 : Contoh kode dengan bahasa algoritmik adalah sebagai berikut :

ayam ← 5

sapi ← 1

kambing ← 1

bebek ← 5

Task 3 : Proses penyelesaian masalah, merupakan tahapan untuk penyelesaian masalah untuk memenuhi tujuan sebuah algoritma dibuat.

Step 1 : Menambah jumlah hewan

hewan ← hewan sekarang + jumlah penambahan ke kandang

Step 2 : Mengurangi jumlah hewan

hewan ← hewan sekarang - jumlah pengurangan ke kandang

Task 4 : Finalisasi, merupakan tahapan bersih-bersih atau tahap akhir misalnya ayam telah habis diambil atau mengetahui jumlah hewan sekarang

Step 1 : Mengosongkan jumlah ayam apabila ayam telah diambil semua.

ayam ← 0

Step 2 : Menghitung jumlah hewan keseluruhan.

hewan ← ayam+kambing+sapi+bebek

Step 3 : Menyajikan hasil perhitungan yang telah dibuat.

Output (“ jumlah hewan sekarang adalah 10 ekor”)

1.3 Aktivitas

1. Mahasiswa memahami implementasi task 1, task 2, task 3 dan task 4
2. Mahasiswa mengerjakan soal latihan

1.4 Latihan

1. Buatlah algoritma untuk melakukan daftar ulang masuk ugm
2. Buatlah algoritma untuk menjumlahkan dua bilangan
3. Buatlah algorima untuk menentukan suatu bilangan termasuk bilangan ganjil atau genap
4. Buatlah algoritma untuk menghitung luas lingkaran

BAB II
PENGENALAN BAHASA C++

2.1 Tujuan Pembelajaran

a. Mahasiswa mengenal dasar-dasar bahasa pemrograman C++.
b. Mahasiswa dapat memetakan bahasa logaritmik ke dalam bahasa pemrograman C++.
c. Mahasiswa mampu membuat program mulai dari kasus sederhana dengan menggunakan bahasa pemrograman C++.
d. Mahasiswa mengenal tipe dasar integer, riil, karakter, string, dan boolean.
e. Mahasiswa paham bagaimana mengimplementasikan suatu operasi dari tipe data dasar tersebut ke dalam bahasa pemrograman C++.
f. Mahasiswa mengenal definisi dan macam-macam operator.
g. Mahasiswa mampu mengimplementasikan operator-operator tersebut pada studi kasus yang berikan.
h. Mahasiswa mengenal konsep dasar Input Output (I/O).
i. Mahasiswa paham bagaimana langkah-langkah mengimplementasikan dengan masukan *(input)* tertentu dan menampilkan keluaran *(output)* dari hasil yang diharapkan pada contoh kasus-kasus yang diberikan.

2.2 Teori

Compiler merupakan perangkat lunak yang digunakan untuk mengubah kode program *(source code)* menjadi bahasa mesin (binary file) agar dapat dieksekusi oleh komputer. Program akan berhasil di-*compile* jika program tersebut tidak mengandung kesalahan secara kaidah sama sekali *(syntax error)*. Contoh compiler C/C++ yang populer adalah minGW dan GCC, biasanya compiler tersebut telah dikemas/bundled bersama dengan software IDE-nya **(Integrated Development Environment).** Software-software IDE merupakan lembar kerja terpadu dengan fasilitasnya termasuk compiler untuk pembuatan atau pengembangan program. Software IDE C/C++ yang cukup populer dan bersifat

opensource(tidak berbayar) diantaranya adalah Codeblock, Dev C++, dan Sublime (untuk Mac OS)

Bahasa pemrograman C++ merupakan bahasa pemrograman yang bersifat case sensitif yang berarti *compiler* membedakan huruf besar dan huruf kecil, misalnya jika kita menuliskan **printf** dan **Printf** pada bahasa C maka *compiler C* akan menganggap kedua tulisan tersebut berbeda maknanya. Dalam praktikum bahasa pemrograman C++ kali ini, kita akan menggunakan **Dev C++** yang merupakan program IDE sekaligus compiler yang bersifat *open source.*

Adapun bagian-bagian yang mendukung dalam pembuatan suatu program yang dibuat dengan bahasa pemrograman C++, antara lain :

- **Komentar**, merupakan bagian kode program yang tidak dieksekusi oleh *compiler.* Komentar dianggap penting untuk memperjelas program agar lebih mudah dimengerti dan memberikan informasi-informasi dari bagian-bagian tertentu kode program.
- ***Identifier,*** nama yang diberikan oleh programmer (orang yang membuat program). Penamaan suatu identifier dapat digunakan pada nama program, nama fungsi, atau obyek-obyek lain yang terlibat dalam bahasa pemrograman, seperti nama variabel, konstanta yang akan dibahas lebih lanjut.
- ***Keyword, kata kunci*** *yang* merupakan kata-kata tertentu yang mengandung arti khusus yang terdapat dalam bahasa pemrograman. Dalam bahasa pemrograman C++, yang dinamakan *keyword* misalnya asm, class, delete, friend, inline, new, operator, private, protected, public, template, this, virtual, dsb. Kata-kata yang dianggap sebagai keyword menurut standar suatu bahasa pemrograman tersebut tidak boleh dipakai sebagai nama **identifier.**
- ***Library function,*** berbeda dengan *keyword, library function* adalah pustaka yang berisi fungsi-fungsi yang telah disediakan oleh bahasa C++ dalam file-file *header* atau *library*-nya. Misalnya salah satu fungsi library yaitu **cout** disimpan pada file iostream, digunakan untuk mencetak ke layar monitor. Fungsi-fungsi bawaan dari bahasa lain juga bisa digunakan asalkan dicantumkan di bagian awal.
-

Struktur program

main.cpp

```
1  // my first program in C++
2  #include <iostream>
3  using namespace std;
4  int main () {
5  cout << "Hello World!";
6  /* Ini juga baris komentar */
7  return 0;
8  }
9
```

- ***// my first program in C++*** , merupakan baris komentar yang diawali dengan dua buah tanda miring (//) atau diapit oleh tanda /**/ dan tidak berpengaruh terhadap program. Dalam kasus ini, baris komentar ini digunakan untuk mendeskripsikan suatu program atau bagian-bagian kode yang dibuat.
- ***#include <iostream>*** , diawali dengan tanda pagar (#) atau, baris ini merupakan baris preprocessor. Dalam kasus ini, *#include <iostream>* menyatakan untuk menyertakan file satandar iostream. Fungsi-fungsi yang sering dipakai oleh pemrogram awal dari pustaka iostream ini diantaranya: cin, cout, system("pause"),
- ***using namespace std;***, baris ini memberitahukan kepada compiler bahwa program yang sedang ditulis menggunakan standar C++ library. Terkadang *dengan* ***using namespace std;*** maka programmer tidak perlu menuliskan .h pada beberapa library standar C++ yang dicantumkan.
- ***int main ()*** , baris ini merupakan fungsi main (). Baris ini merupakan inti dari program. Baris ini terdiri dari serangkaian source kode yang diawali dengan tanda kurung kurawal buka { dan tanda kurung kurawal tutup } dan tanda {} itu menunjukkan dimana fungsi main () berawal dan berakhir atau disebut blok kode.
- *cout << "Hello World!";* , baris ini merupakan statement (pernyataan) C++. Suatu pernyataan adalah ekspresi sederhana yang dapat menghasilkan beberapa efek. Perintah cout merepresentasikan standar ouput dalam C++, cout ini dideklarasikan pada file standar iostream dalam namespace std. Jadi baris kode ini berfungsi untuk menampilkan kalimat "hello world".

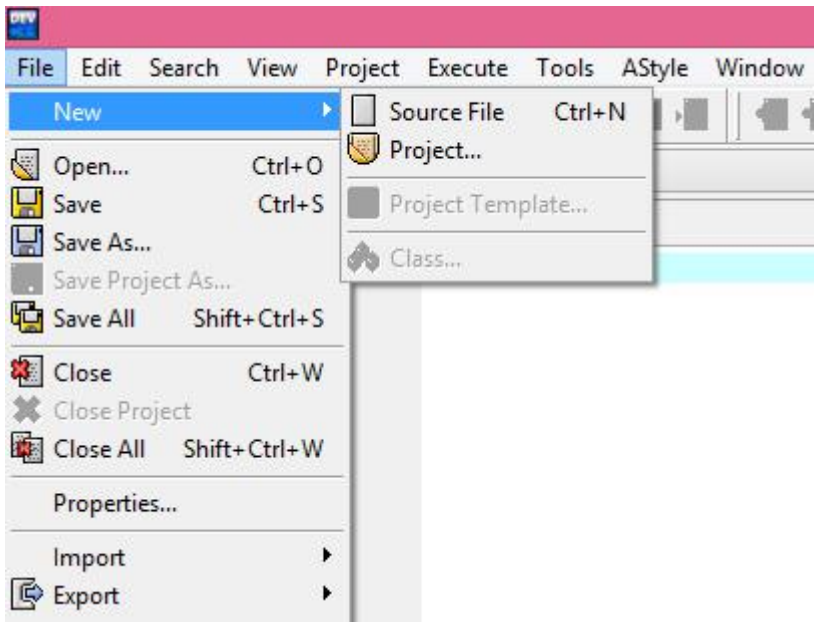
Perhatikan bahwa pernyataan diakhiri dengan karakter titik koma (;). digunakan untuk menandai akhir dari pernyataan dan harus disertakan pada akhir semua pernyataan ekspresi dalam semua C + + program .

- *Return 0;,*intruksi return menyebabkan fungsi main() berakhir dan mengembalikan kode yang mengikuti instruksi tersebut, dalam kasus ini 0. Ini merupakan cara yang paling sering digunakan untuk mengakhiri program tanpa timbal balik apapun.

Skenario

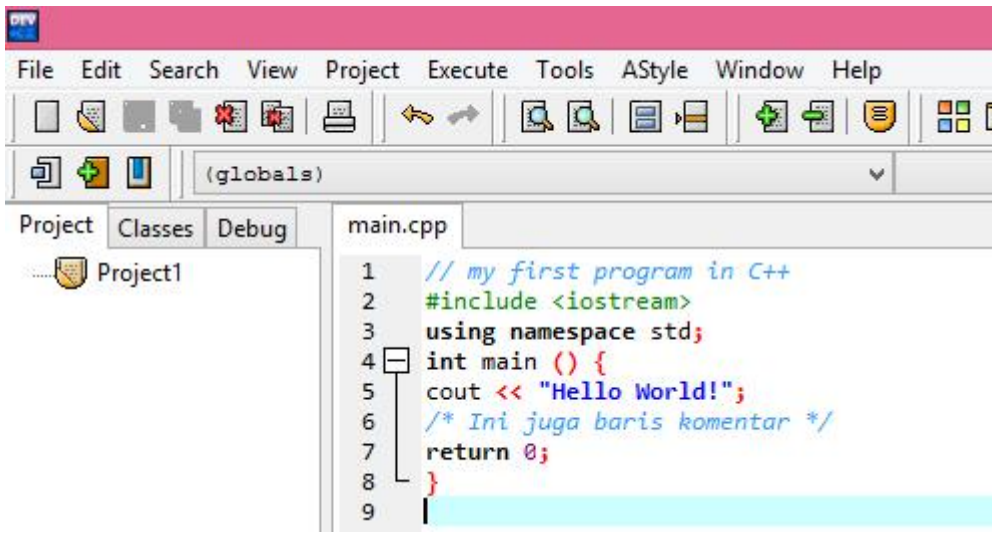
1. **Task 1 : Membuat proyek baru di devc++ dalam OS windows**

   1) **Step 1 : Pada aplikasi c++, pilih File > New Project > New Source File**



Gambar 2.1 Tampilan proyek baru

   2) **Step 2 : Ketikkan kode program pada area kerja**



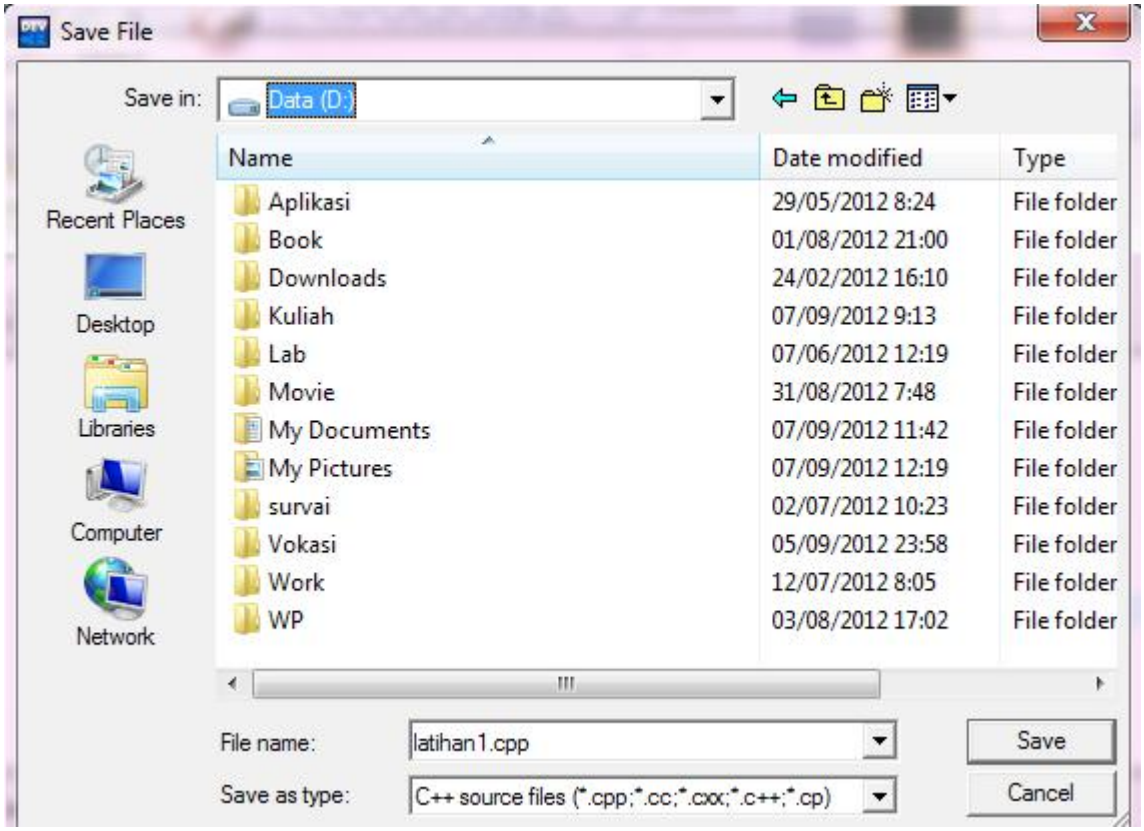
Gambar 2.2 Tampilan source code

Penjelasan tampilan source code

2. Task 2 : Menyimpan proyek baru

1) Step 1 : Pilih Menu File > Save As >

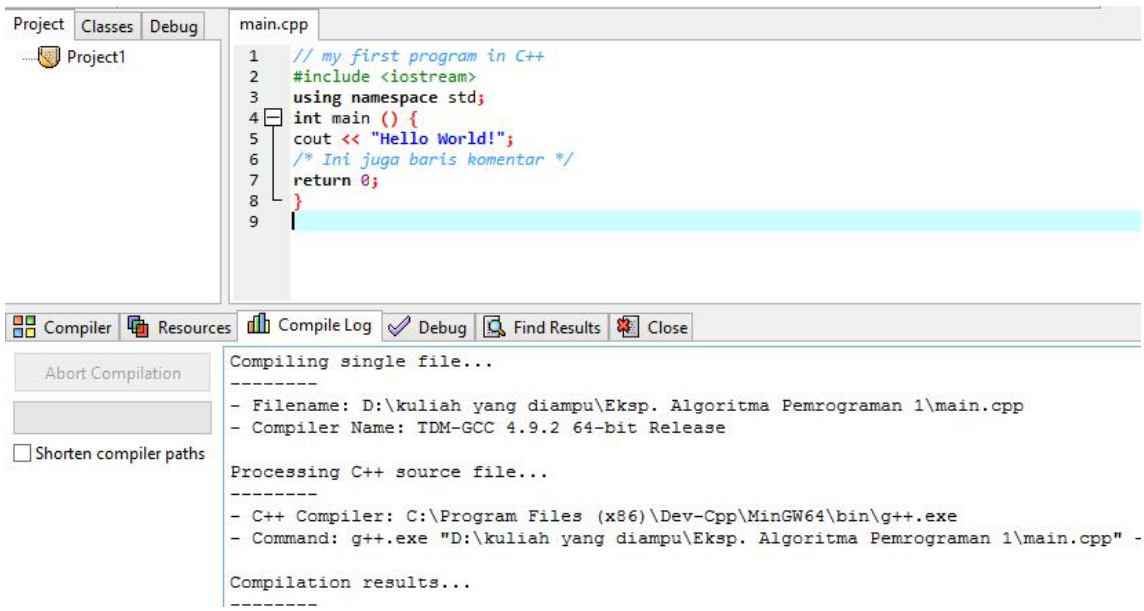
2) **Step 2 : Pilih direktori penyimpanan dan beri nama file**



Gambar 2.3 Tampilan Save As

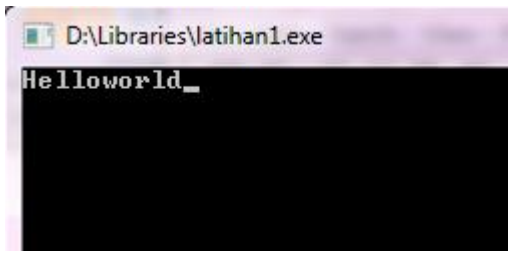
3. Task 3 : Kompilasi dan Eksekusi program

1) Step 1 : Pilih menu Execute > Compile



Gambar 2.4 Tampilan compile program

2) **Step 2 : Pilih menu Execute > Run**



Gambar 2.5 Tampilan running program

4. Task 4 : Membuat proyek baru dalam OS Linux

Pemrograman C++ di dalam sistem operasi Linux menggunakan software g++ %GNU C++%. Kompiler ini dapat secara otomatis terinstall pada saat menginstall sistem operasi, dengan catatan kita memilih opsi untuk menginstall g++.

Kompilasi menggunakan g++ dilakukan di dalam console Linux (terminal). Cara yang dilakukan adalah terlebih dahulu mengetikkan file source C++ pada text editor kemudian disimpan sebagai file berekstensi .cpp, .cp, cxx, c++, cp, atau cc (contoh : coba1.cpp). Lalu tempatkan dalam folder home/<nama user>. Ketikkan perintah

g++ <path/nama file.cpp> -o <file output>

contoh : **g++ coba1 –o keluaran** (kemudian tekan enter).

Path file tidak diperlukan apabila terlebih dahulu kita masuk ke dalam folder tempat file source disimpan. File output bisa tidak diberikan, namun secara default g++ akan membuat file output bernama a.out. Dalam contoh tersebut bila tak ada error maka kompilasi akan menghasilkan(output) file binary dengan nama **keluaran.**

Setelah kompilasi dilakukan, cek apakah kompilasi sudah berhasil dengan melihat pesan hasil kompilasi. Jika terdapat pesan error maka file output belum terbentuk. Kita perlu melakukan editing source(melalui text editor) kemudian compile lagi. Cara lain untuk mengetahui apakah kompilasi berhasil dengan memasuki folder tempat file source di simpan, dan ketikkan perintah

ls **atau** *dir*

Apabila telah muncul file output yang kita definisikan atau file a.out maka kompilasi berhasil. Sebaliknya jika file belum muncul maka kompilasi belum berhasil. Eksekusi file output dengan mengetikkan perintah

./<nama file output>

Contoh : **./keluaran**

Perintah ini membuat kita dapat melihat hasil program yang sudah dibuat.

Tipe Data Dasar *(Integer, Real, Karakter, String, Boolean)*

a. Tujuan Pembelajaran

- Mahasiswa mengenal tipe dasar integer, riil, karakter, string, dan boolean.
- Mahasiswa paham bagaimana mengimplementasikan suatu operasi dari tipe data dasar tersebut ke dalam bahasa pemrograman C++.

b. Latar Belakang

Awal dari pembahasan pemrograman, kita akan menggunakan tipe data dasar terlebih dahulu yang meliputi tipe data integer, real, char, string dan boolean.

c. Teori

Tipe data adalah jenis data berdasarkan isi dan sifatnya. Misalnya kita analogikan dengan contoh kasus sehari-hari yaitu galon air hanya khusus dibuat untuk menampung jenis benda dengan jenis tertentu yaitu benda yang berjenis cair, misalnya air.

Adapun jenis-jenis tipe data dasar pemrograman, antara lain :

- **Integer,** tipe bilangan bulat biasa disebut sebagai integer, namun tipe bilangan bulat tidak hanya terdiri dari integer, masih ada tipe lain seperti short dan long, yang membedakan ketiga tipe tersebut adalah jangkauan bilangannya.
- **Real,** tipe ini digunakan untuk menyatakan bilangan yang membutuhkan ketelitian dengan adanya nilai dibelakang koma. Diantaranya yaitu: double, single, float.
- **Char,** tipe data yang digunakan untuk menyimpan sebuah karakter.
- **String,** tipe data yang berupa kumpulan karakter (satu atau lebih) yang berada di dalam dua buah tanda petik dua (“) dalam bahasa C.
- **Boolean**, tipe data yang digunakan untuk menyatakan pernyataan benar *(true)* atau salah *(false)*.

Jangkauan tiap tipe data berbeda-beda baik dari nilai jangkauannya maupun bahasa pemrograman.

Tabel 2.1 Jangkauan tipe data

Tipe Dasar	Jangkauan Nilai	Jumlah Digit Presisi
Char	-128 hingga +127	-
Int	-32768 hingga +32767	-
Long	-2.147.438.648 hingga 2.147.438.647	-
Float	3,4E-38 hingga 3,4E38	6-7
Double	1.7E-308 hingga 1.7E308	15-16
Long Double	3.4E-4932 hingga 1.1E4932	19

Variabel merupakan tempat untuk menyimpan data dengan tipe tertentu yang isinya bisa diubah-ubah sesuai dengan tipenya dan konstanta sebenarnya adalah variabel yang ditentukan nilai standarnya *(default)* dari awal dan biasanya nilainya tidak diubah-ubah. Variabel dan konstanta harus dideklarasikan terlebih dahulu agar program dapat mengalokasikan memory untuk menampung data yang spesifik dan memprosesnya dalam

program sehingga didapatkan output yang sesuai. Bahasa C++ telah mendukung deklarasi variabel sekaligus inisialisasi, contohnya adalah :

```
1  string tabelmerk[5]={"adidas adizero Indonesia","adidas climacool Vietnam ", "adidas f50 China "};
2  //deklarasi array sebanyak 5 larik,
3     //tiap larik mampu penyimpan string
4  tabelmerk[3]="Nike mercurial Indonesia"; //inisialisasi setiap larik
5  tabelmerk[4]="Nike Air Vietnam";
6
```

Inisialisasi juga merupakan **penugasan** yaitu proses memasukkan nilai kedalam variabel dengan bantuan operator (=). Penugasan sering disebut sebagai "***assignment***". Dari contoh diatas , proses assignment-nya dalah memasukkan data **string** (kumpulan karakter) kedalam setiap larik variabel **tabelmerk[0]** hingga larik **tabelmerk[4].**

Konstanta mirip dengan variabel, namun memiliki nilai tetap. Konstanta dapat berupa nilai interger, float, karakter, dan string. Pendeklarasian konstanta dapat dilakukan dengan dua cara:

- Menggunakan (`#define`)

Deklarasi konstanta dengan cara ini, lebih mudah dilakukan karena akan menyertakan `#define` sebagai preprocessor directive. Dan sintaksnya diletakkan bersama-sama dengan pernyataan `#include` (diatas main( ))

Format penulisannya adalah:

```
#define pengenal nilai
```

Contoh penggunaan :

main.cpp

```
1  #define phi 2.414159265
2  #define Newline '\n'
3  #define lebar 100
4
```

Pendeklarasian dengan `#define` tanpa diperlukan adanya tanda = untuk memasukkan nilai ke dalam pengenal dan juga tanpa diakhiri dengan tanda titik koma atau semicolon (;)

- Menggunakan konstanta (const)

Sedangkan dengan kata kunci const, pendeklarasian konstanta mirip dengan deklarasi variabel yang ditambah kata depan const dan langsung inisialisasi.

Contoh:

```
1   const int lebar = 100;
2   const char tab = 'p';
3   const zip = 912;
4
```

Untuk contoh terakhir, deklarasi variabel zip yang tanpa tipe data, maka compiler akan secara otomatis memasukkannya kedalam tipe **`int`**

d. *Skenario*

Tahapan-tahapan yang terpenting dalam membuat suatu program adalah dimulai dengan mendeklarasikan variabel yang akan dipakai dan jenis tipe datanya, selanjutnya melakukan inisialisasi pada variabel tersebut, kemudian mendefinisikan proses-proses penyelesaian masalah yang dapat berupa rumus-rumus perhitungan atau instruksi dan perintah-perintah yang lain. Pendeklarasian suatu variabel dan tipe datanya pada bahasa C++ dapat dilihat pada step-step berikut :

1. **Task 1 : Membuat program penjumlahan 2 bilangan bulat :**
   1) **Step 1 : Deklarasi variabel-variabel**

      Penjelasan: Variabel-variabel yang digunakan untuk proses penjumlahan dua buah bilangan bulat, adalah a dan b. Oleh karena dua buah bilangan yang diproses merupakan bilangan bulat, maka tipe data yang digunakan pada variabel a dan variabel b adalah integer.
   2) **Step 2 : Inisialisasi**

      Penjelasan: Inisialisasi pada variabel a dan b adalah pemberian nilai awal pada kedua variabel yang akan digunakan.
   3) **Step 3 : Proses**

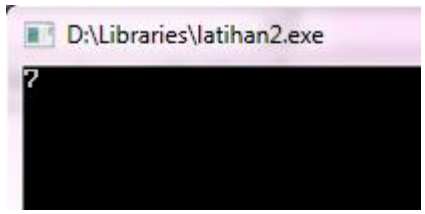
      Penjelasan: Berisi proses penjumlahan.
   4) **Step 4 : Finalisasi**

      Penjelasan: merupakan tahapan untuk mengakhiri program, meliputi tampilan hasil ke output devices (layar), dan return 0 digunakan untuk mengakhiri dan mengembalikan nilai, karena program kita menggunakan INT (lihat pada fungsi main), maka perlu adanya pengembalian nilai. Untuk itu kita gunakan return 0 agar tidak terjadi kesalahpahaman antara kita dengan program.

```
1  // my first program in C++
2  #include <iostream>
3  using namespace std;
4  int main () {
5  //diklarasi
6  int a,b;
7  int hasil;
8
9  //inisialisasi
10 a=5;
11 b=2;
12
13 //proses
14 hasil=a+b;
15
16 //menampilkan hasil
17 cout << hasil;
18 getchar();
19
20 //memberhentikan program
21 return 0;
22 }
23
```

Gambar 2.6 Tampilan kode program

2. **Task 2 : Kompilasi & Eksekusi program**



Gambar 2. Tampilan hasil program

e. Latihan

1. Buatlah program untuk menentukan suatu bilangan termasuk bilangan ganjil atau genap.
2. Buatlah program untuk menentukan suatu bilangan termasuk bilangan prima.
3. Buatlah program untuk menghitung luas lingkaran.

Operator

f. Tujuan Pembelajaran

- Mahasiswa mengenal definisi dan macam-macam operator.
- Mahasiswa mampu mengimplementasikan operator-operator tersebut pada studi kasus yang berikan.

a. Latar Belakang

Untuk memproses suatu pemrosesan yang lebih rumit terkadang kita membutuhkan suatu simbol atau perintah tanda untuk memperlakukan terhadap suatu variabel-variabel yang dinamakan dengan operator.

b. Teori

Operator adalah simbol atau tanda yang jika diletakkan pada dua buah operan dapat menghasilkan sebuah hasil, contohnya pada matematika yakni tanda tambah (+) jika diletakkan di antara dua buah angka akan menghasilkan angka lain hasil pertambahan dua angka tersebut. Tanda tambah inilah yang disebut sebagai operator.

Operator memiliki beberapa jenis sebagai berikut :

- **Operator Aritmatika**

Operator	Deskripsi	Contoh
+	Penjumlahan	a+b
-	Pengurangan	a-b
*	Perkalian	a*b
/	Pembagian	a/b
%	Sisa pembagian (modulo)	a%b
-	Negasi	-a

Operator negasi (-) disebut unary operator, karena membutuhkan hanya satu buah operand , sedangkan operator % (modulus) digunakan untuk mencari sisa pembagian antara dua bilangan.

Misalnya : 9 % 2 = 1, 9 % 3 = 0

- **Operator Relasi**

Operator	Deskripsi	Contoh	
= =	Sama dengan	a = = b	Apakah a sama dengan b

! =	Tidak sama dengan	a != b	Apakah a tidak sama dengan b
>	Lebih besar	a>b	Apakah a lebih besar b
<	Lebih kecil	a<b	Apakah a lebih kecil b
>=	Lebih besar atau sama dengan	a>=b	Apakah a lebih dari atau sama dengan b
<=	Lebih kecil atau sama dengan	a<=b	Apakah a kurang dari atau sama dengan b

- **Operator Increment & Decrement**

Operator	Deskripsi	Contoh	Arti
++	Increment	a ++	a=a+1
- -	Decrement	b - -	b=b-1

- **Operayor Bitwise**

Operator	Deskripsi	Contoh
<<	Geser n bit ke kiri (left shift)	a >> b
>>	Geser n bit ke kanan (right shift)	a << b
&	Bitwise AND	a & b
|	Bitwise OR	a | b
^	Bitwise XOR	a ^ b
~	Bitwise NOT	~ b

- **Operator Logika**

Operator	Deskripsi	Contoh
& &	Logika AND	a && b
| |	Logika OR	a | | b
!	Logika NOT	! b

Pada operator logika diatas seringkali dipakai dalam kondisi percabangan atau perulangan, contoh :

```
1   If ( (a+b>c) && (a+c>b) &&( b+c>a))
2   while(x>='f' || x<='c')
3
```

yang tidak boleh digunakan dalam kondisi percabangan ataupun perulangan ialah operator assignment yaitu '='. Contoh nya:

```
1   if(a=b+4) //ini  Salah,
2   if(a==b+4) // penulisan yang BENAR
3
```

Pada perintah **for** nantinya juga dijumpai assignment namun tidak pada kondisinya, melainkan hanya pada bagian pencacah maju atau mundur. Contoh:

`for(int i=0; i<=9; i=i+2)` // , i=i+2 tidak terletak pada kondisi karena dipisahkan tanda (;)

//tetapi bagian yang menyatakan kondisi adalah pada (i<=9)

- **Operator Kondisi**

Operator kondisi ditandai dengan notasi **?** 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

I/O

a. Tujuan Pembelajaran

- Mahasiswa mengenal konsep dasar Input Output (I/O).
- Mahasiswa paham bagaimana langkah-langkah mengimplementasikan dengan masukan *(input)* tertentu dan menampilkan keluaran *(output)* dari hasil yang diharapkan pada contoh kasus-kasus yang diberikan.

b. Latar Belakang

Ketelitian dalam suatu pemrosesan terhadap masalah terletak pada hasil yang didapatkan apakah sesuai dengan yang dibutuhkan atau tidak (*valid* atau *invalid*). Untuk mendapatkan hasil yang diharapkan, maka harus diperhatikan input (masukan), proses, dan output (keluaran).

c. Teori

Operasi pada input dilakukan untuk membaca data atau nilai yang akan diproses. Nilai dari suatu variabel dapat ditentukan di dalam program atau dimasukkan oleh *user* (pengguna) dari *keyboard* dengan menggunakan fungsi yang telah ada pada *library* pada bahasa pemrograman c++.

Berbeda dengan operasi pada *input*, operasi pada *output* dilakukan untuk mengirimkan atau menampilkan data atau nilai kepada piranti keluaran *(output device)*, misalnya printer atau layar *(monitor)*. Contoh dari operasi ouput yaitu menampilkan kalimat ke layar, biasanya dilakukan untuk menampilkan perintah untuk memasukkan input ke program, atau menampilkan nilai dari suatu variabel ke *monitor* dengan menggunakan library yang ada pada bahasa pemrograman c++, biasanya dilakukan untuk menampilkan hasil suatu perhitungan atau hasil suatu solusi. Dalam library iostream C++, standard operasi input dan output untuk pemrograman didukung oleh 2 data streams: cin untuk input dan cout untuk output.

Standard output (cout), penggunaan cout stream dhubungkan dengan operator overloaded << (Sepasang tanda "less than").

Contoh :

```
1   cout << "Kalimat keluaran"; // mencetak kalimat keluaran pada layar.
2   cout << 120; // mencetak angka 120 pada layar .
3   cout << x; // mencetak isi variabel x pada layar.
4
```

Operator **<<** dikenal sebagai insertion operator, dimana berfungsi untuk menginput data yang mengikutinya. Jika berupa string, maka harus diapit dengan kutip ganda (**"**), sehingga membedakannya dari variable.

Contoh :

```
1   cout << "Hello"; // mencetak kalimat keluaran yaitu Hello pada layar .
2   cout << Hello; // mencetak isi dari variabel Hello pada layar.
3
```

Operator insertion (<<) dapat digunakan lebih dari 1 kali dalam kalimat yang sama, contoh :

```
1   cout << "Hello, " << "Saya " << " kalimat C++";
```

Contoh diatas akan menampilkan Hello, I am a C++ sentence pada layar monitor. Manfaat dari pengulangan penggunaan operator insertion (<<) adalah untuk menampilkan kombinasi dari satu variabel dan konstanta atau lebih,

Contoh 1 :

```
1   cout << "Hello, umurku" << umur << "tahun dan
2   aku angkatan" << angkatan;
3
```

Misalkan variabel umur diisi dengan angka 18, dan variabel angkatan diisi dengan 2012. Maka output yang dihasilkan adalah :

Hello, umurku 18 tahun dan aku angkatan 2012

Contoh 2 :

```
1   cout << "Kalimat Pertama.\n ";
2   cout << "Kalimat Kedua.\nKalimat Ketiga.";
3
```

Maka keluaran yang dihasilkan adalah :

Kalimat Pertama.

Kalimat Kedua.

Kalimat Ketiga.

Selain dengan karakter new-line, dapat juga menggunakan manipulator endl, contoh :

```
1   cout << "Kalimat Pertama. " << endl;
2   cout << "Kalimat Kedua" << endl;
3
```

Output :

Kalimat Pertama.

Kalimat Kedua.

Standard input (cin), penggunaannya dengan menambahkan overloaded operator extraction (>>) pada cin stream. Harus diikuti bengan variable yang akan menyimpan data. Contoh :

```
1   cout << "Kalimat Pertama. " << endl;
2   cout << "Kalimat Kedua" << endl;
3
```

Contoh diatas mendeklarasikan variabel umur dengan tipe int dan menunggu input dari cin (keyborad) untuk disimpan di variabel umur. Perintah cin akan memproses input dari keyboard sekali saja dan tombol ENTER harus ditekan. cin juga dapat digunakan untuk lebih dari satu input :

cin >> a >> b;

Equivalen dengan :

cin >> a;

cin >> b;

Dalam hal ini data yang di input harus 2, satu untuk variabel a dan lainnya untuk variabel b yang penulisannya dipisahkan dengan : spasi, tabular atau newline.

Biasanya cout (standard output stream) ditujukan untuk monitor dan cin (standard input stream) ditujukan untuk keyboard. Dengan menggunakan dua streams ini, maka kita dapat berinteraksi dengan user dengan menampilkan messages pada monitor dan menerima input dari keyboard.

d. *Skenario*

Contoh implementasi dari operasi I/O (Input Output), yaitu program yang digunakan untuk menerima input dari mahasiswa yaitu yang berupa nim, nama, umur, alamat, dan nilai NEM dan menampilkan data diri tersebut masing-masing mahasiswa ke layar *monitor*.

1. **Task 1 : Membuat program data mahasiswa.**
   1) **Step 1 : Mendeklarasikan variabel-variabel yang dipakai beserta tipe datanya.**

      Penjelasan : Variabel-variabel yang digunakan untuk proses menampilkan data diri mahasiswa, yaitu variabel nim, nama, dan alamat yang masing-masing variabel tersebut bertipe data char, yang membedakan hanya ukurannya. Variabel umur bertipe integer, dan variabel nem bertipe float karena terdiri dari angka yang memerlukan ketelitian angka dibelakang koma.
   2) **Step 2 : Inisialisasi.**

Penjelasan : Oleh karena pada program ini menerima masukan langsung dari user, maka tidak pelu diberi inisialisasi awal.

3) **Step 3 : Proses pemecahan masalah.**

Penjelasan : Membaca masukan (input) dari user dengan menggunakan fungsi cin, dan menampilkan hasilnya ke layar dengan menggunakan fungsi cout.

4) **Step 4 : Finalisasi.**

Penjelasan : Menghentikan fungsi main() yaitu dengan perintah return 0.

main.cpp

```
1  // my first program in C++
2  #include <iostream>
3  using namespace std;
4  int main () {
5  char nim[20];char nama[20];char alamat[30];int umur;float nem;
6
7  //Memasukkan data ke variabel
8  cout<<"Masukkan nama Anda :";
9  cin>>nama;
10 cout<<"Masukkan NIM Anda :";
11 cin>>nim;
12 cout<<"Masukkan alamat Anda :";
13 cin>>alamat;
14 cout<<"Masukkan umur Anda :";
15 cin>>umur;
16 cout<<"Masukkan nem Anda :";
17 cin>>nem;
18 //Menampilkan hasil ke layar
19
20
21 cout<<"Halo "<<nama<<", Selamat Datang di Departemen DIKE, UGM" << endl;
22 cout<<"Data Anda adalah"<<endl;
23 cout<<"Nama:"<<nama<< endl;
24 cout<<"NIM:"<<nim<< endl;
25 cout<<"Alamat:"<<alamat<< endl;
26 cout<<"Umur:"<<umur<< endl;
27 cout<<"NEM:"<<nem<< endl;
28 return 0;
29 }
30
```

Gambar 2. Tampilan kode program

2. Task 2 : Compile and Execution

Gambar 2. Tampilan running program

2.5 Latihan

1. Buatlah program penjumlahan 2 buah bilangan dengan menggunakan input (masukan) dari user (pengguna)!
2. Buatlah program untuk menghitung waktu yang ditempuh berdasarkan kecepatan dan jarak yang ada dengan menggunakan input dari user!
3. Buatlah program konversi mata uang rupiah ke dolar dengan menggunakan input dari user yaitu yang berupa nominal mata uang rupiah, jika diketahui untuk 1 USD = Rp 9.600,-!

BAB III
STRUKTUR RUNTUNAN DAN CONTROL STATEMENT

3.1 Tujuan

a. Menerangkan langkah-langkah logis dalam suatu algoritma
b. Menerangkan macam-macam control statement di dalam C++
c. Menerangkan cara menggunakan control statement
d. Menerangkan operasi logika untuk mengatur kondisi salam control statement
e. Praktikan mampu menyelesaikan masalah secara runtun dan logis
f. Praktikan memahami cara menggunakan control statement
g. Praktikan memahami cara mengatur kondisi di dalam control statement

3.2 Teori

a. Pendahuluan

Algoritma merupakan runtunan (sequence) satu atau lebih instruksi atau pernyataan (statement), dan setiap pernyataan dikerjakan secara berurutan sesuai dengan urutan penulisannya, yang berarti bahwa :

1. Setiap instruksi dikerjakan satu per satu
2. Tiap instruksi dilaksanakan tepat sekali (tidak ada instruksi yang diulang)
3. Tiap instruksi dilaksanakan dengan urutan yang sama antara pemroses dengan yang tertulis di dalam teks algoritmanya
4. Akhir dari instruksi terakhir merupakan akhir algoritma.

Sebagai contoh, misalkan saja kita memiliki dua buah gelas, gelas A dan gelas B, yang berisi minuman. Gelas A berisi kopi dan gelas B berisi susu. Jika kita ingin menukar isi dari kedua gelas tersebut, susu menempati gelas A dan kopi menempati gelas B. Bagaimana kita akan melakukannya? Perhatikan algoritma berikut:

1. Tuangkan isi gelas A ke gelas B

2. Tuangkan isi gelas B ke gelas A

Apakah yang akan terjadi pada kopi dan susu tersebut? Apakah susu sekarang berada pada gelas A dan kopi berada pada gelas B?

Yang terjadi adalah ketika anda menuang isi gelas A, kopi, ke gelas B pada langkah pertama, gelas B sedang berisi susu, sehingga anda melakukan pencampuran kopi dengan susu. Pada langkah kedua, kopi susu pada gelas B dipindahkan ke gelas A. Hasil akhirnya adalah gelas A berisi kopi susu sedangkan gelas B akan kosong.

Lalu bagaimana algoritma yang benar? Di sini kita memerlukan 1 gelas tambahan, gelas C, untuk menampung sementara isi salah satu gelas agar tidak terjadi percampuran isi gelas. Algoritmnya akan menjadi:

1. Tuangkan isi gelas A ke gelas C
2. Tuangkan isi gelas B ke gelas A
3. Tuangkan isi gelas C ke gelas B

Apa yang terjadi di sini? Kondisi mula-mula adalah gelas A berisi kopi, gelas B berisi susu dan gelas C kosong. Pada langkah pertama kita menuang kopi ke gelas C yang masih kosong, sehingga gelas C berisi kopi dan gelas A menjadi kosong. Pada langkah kedua kita memindahkan susu dari gelas B menuju gelas A yang sudah kosong karena langkah pertama. Hasil akhirnya adalah gelas A berisi susu, gelas B kosong dan gelas C berisi kopi. Langkah terakhir adalah menuangkan kopi ke gelas B sehingga gelas A berisi susu, gelas B berisi kopi dan gelas C kosong kembali. Hasil akhir sesuai dengan yang kita harapkan.

Lalu apakah masalah selesai hanya dengan menggunakan satu gelas tambahan sebagai penyimpan sementara? Ternyata tidak. Misal kita ubah urutan kedua dan ketiga pada algoritma di atas.

1. Tuangkan isi gelas A ke gelas C
2. Tuangkan isi gelas C ke gelas B
3. Tuangkan isi gelas B ke gelas A

Pada algoritma ini, terjadi hal yang hampir sama dengan algoritma pertama kita. Pada langkah pertama, sudah benar kita menuang kopi ke gelas kosong. Tapi pada langkah

kedua, kita menuang kembali kopi tadi, gelas C, ke gelas B yang masih berisi susu. Sehingga gelas A kosong, gelas B kopi susu dan gelas C kosong. Pada langkah terakhir, kita memindahkan kopi susu pada gelas B ke gelas A, sehingga gelas B dan C menjadi kosong.

Dari ketiga algoritma di atas, ternyata untuk menyelesaikan suatu masalah, menukar isi gelas, selain diperlukan cara yang tepat, ternyata urutan dari langkah-langkahnyapun harus benar. Hal yang sama berlaku pada algoritma suatu program yang memerlukan cara yang tepat serta urutan langkah atau runtunan yang benar untuk mendapatan output atau hasil yang diharapkan.

Suatu permasalahan serupa pada algoritma suatu pemrograman adalah ketika kita ingin menukar isi dari dua buah variabel, seperti kode berikut:

```
// Kita ingin menukar isi dari variabel berikut
int x = 10;
int y = 33;
```

Gambar 3. 1. Source Code tukar nilai

Maka yang kita perlukan adalah sebuah variabel pembantu untuk menyimpan sementara data dari variabel tersebut

```
// perlu 1 variabel pembantu
int z = 0;
```

Gambar 3. 2. Source Code variable pembantu

Jika kita cetak isi dari semua variabel pada tahap inisialisasi, maka akan di dapatkan hasil sebagai berikut

```
inisialisasi
x = 10
y = 33
z = 0
```

Gambar 3. 3. Inisialisasi

Langkah pertama dalam melakukan pertukaran ini adalah dengan memindahkan nilai dari variabel x ke variabel z.

```
// pindahkan nilai x ke var z
z = x;
```

Gambar 3. 4. Pindah nilai

Isi dari masing-masing variabel setelah proses ini adalah sebagai berikut

```
pertukaran pertama x ke z
x = 10
y = 33
z = 10
```

Gambar 3. 5. Pertukaran variabel

Langkah berikutnya adalah memindahkan nilai dari variabel y ke variabel x

```
// pindahkan nilai var y ke var x
x = y;
```

Gambar 3. 6. Pindah nilai dalam variabel

Sehingga masing-masing variabel akan bernilai

```
pertukaran kedua y ke x
x = 33
y = 33
z = 10
```

Gambar 3. 7. Hasil Tukar nilai

Langkah terakhir adalah memindahkan nilai variabel z ke variabel y

```
// pindahkan nilai var z ke var y
y = z;
```

Gambar 3. 8. Pindah Variabel

sehingga hasil dari langkah ini merupakan solusi dari permasalahan awal, yaitu

```
pertukaran terakhir z ke y
x = 33
y = 10
z = 10
```

Gambar 3. 9. Tukar nilai

Berikut seluruh kode dan output dari persoalan di atas

```
#include <iostream>

using namespace std;

int main()
{
    // Kita ingin menukar isi dari variabel berikut
    int x = 10;
    int y = 33;

    // perlu 1 variabel pembantu
    int z = 0;

    cout << "inisialisasi" << endl;
    cout << "x = " << x << endl;
    cout << "y = " << y << endl;
    cout << "z = " << z << endl;

    // pindahkan nilai x ke var z
    z = x;
```

Gambar 3. 10. Source Code Tukar Nilai

```
    cout << "\npertukaran pertama x ke z" << endl;
    cout << "x = " << x << endl;
    cout << "y = " << y << endl;
    cout << "z = " << z << endl;

    // pindahkan nilai var y ke var x
    x = y;

    cout << "\npertukaran kedua y ke x" << endl;
    cout << "x = " << x << endl;
    cout << "y = " << y << endl;
    cout << "z = " << z << endl;

    // pindahkan nilai var z ke var y
    y = z;

    cout << "\npertukaran kedua z ke y" << endl;
    cout << "x = " << x << endl;
    cout << "y = " << y << endl;
    cout << "z = " << z << endl;

    return 0;
}
```

Gambar 3. 11. Source Code Tukar Nilai(lanjt)

```
inisialisasi
x = 10
y = 33
z = 0

pertukaran pertama x ke z
x = 10
y = 33
z = 10

pertukaran kedua y ke x
x = 33
y = 33
z = 10

pertukaran kedua z ke y
x = 33
y = 10
z = 10
```

Gambar 3. 12. Proses Pertukaran

b. Control statement

Control statement atau lebih dikenal dengan sebutan percabangan dapat menjadikan program menjadi fleksibel. Dengan control statement kita bisa mendefinisikan baris program yang akan dieksekusi apabila suatu kondisi terpenuhi, dan tidak dieksekusi apabila kondisi tidak terpenuhi. Dalam C++ control statement ada dua jenis. Pertama, memakai if...else dan yang kedua memakai switch case.

If ... Else

Bentuk umum dari statement if adalah

```
if (kondisi)
{
      statement1;
      statement2;
      ....
      statementN;
}
```

atau :

```
if (kondisi)
{
      statement1;
      statement2;
      ....
      statementN;
}

else if (kondisi)
{
      statement1;
      statement2;
      ....
      statementN;
}

........

else
{
      statement1;
      statement2;
      ....
      statementN;
}
```

Tingkatan if...else disesuaikan dengan kebutuhan.

Kondisi yang harus dipenuhi dapat berbentuk ekspresi dengan operator logika dan operator relasi.

Operator logika yang bisa digunakan antara lain :

Simbol	***Arti***
&&	AND
||	OR
!	NOT

Sedangkan operator relasi antara lain :

Simbol	***Arti***
==	sama dengan
>=	lebih dari sama dengan
>	lebih dari
<=	kurang dari sama dengan
<	kurang dari
!=	tidak sama dengan

Contoh program

```
#include <iostream>

using namespace std;

int main()
{
    int b;
    float jumlah = 0;
    cout << "Masukkan nilai b = ";
    cin >> b;

    if (b > 10)
        jumlah = jumlah + b;
    else
        jumlah = jumlah - b;

    cout << "Jumlah = " << jumlah << endl;

    return 0;
}
```

Gambar 3. 13. Source code I/O

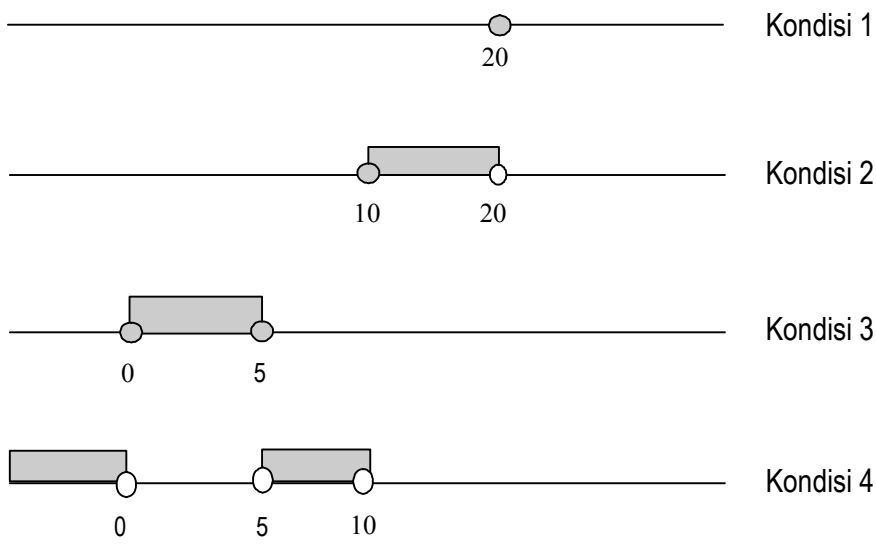
contoh di atas apabila variabel diberi nilai lebih dari 10 maka statement jumlah = jumlah + b akan dijalankan. Namun apabila nilai yang dimasukkan sama dengan 10 atau kurang dari 10, maka statement else dijalankan. Karena statement hanya satu baris maka tanda kurawal bisa dihilangkan.

Kursor kompilasi akan menjalankan statement jumlah = jumlah + b apabila kondisi pertama true, kemudian kebenaran dari else tidak dicek karena kondisi pertama sudah true. Namun jika kondisi pertama tidak true baru else dicek.

Dalam mendesain control statement sebaiknya kita menggunakan garis bilangan supaya mudah untuk mengetahui wilayah true masing-masing kondisi. Semisal kita memiliki control statement dengan kondisi sbb :

```
if (b >= 20)
      ....
else if (b < 20) AND (b >= 10)
      ....
else if (b <= 5) AND (b >= 0)
      ....
else
      ....
```

contoh di atas memiliki 4 kondisi dengan garis bilangan masing-masing

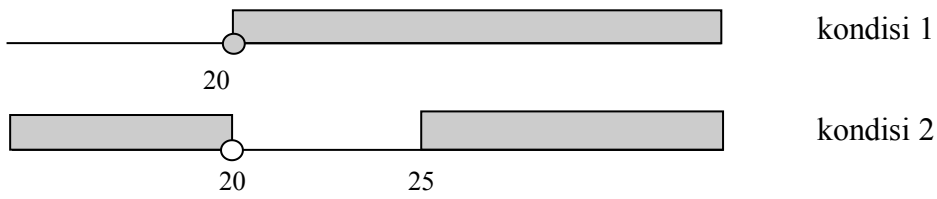


Pengecekan 4 kondisi di atas terjadi pertama kali di kondisi 1. Apabila kondisi 1 sudah true maka kondisi-kondisi berikutnya tidak dicek oleh kursor kompilasi. Namun jika kondisi pertama tidak true, maka kondisi kedua dicek. Kondisi kedua sudah true maka kondisi lainnya di bawah kondisi dua tidak d . Begitu seterusnya hingga kondisi keempat. Pengecekan seperti ini terjadi karena control statement di atas masih dalam satu blok.

Pentingnya menggambarkan garis bilangan juga kita jumpai pada kasus tertentu seperti contoh berikut :

```
if (b >= 20)
      ....
else if (b < 20) AND (b >= 25)
      ....
```

Garis bilangannya sbb :



Terdapat ketimpangan daerah true antara kondisi 1 dan 2. Ketimpangan terjadi pada b >= 25. Kondisi 1 dan 2 memenuhi nilai tersebut. Namun karena kedua kondisi masih dalam satu blok control statement maka kondisi 1 saja yang akan dijalankan statementnya karena kondisi satu lebih dahulu dijumpai oleh kursor kompilasi.

Blok kondisi dimulai dengan kata if dan diakhiri dengan kata else atau adanya blok kondisi baru. Misalnya

```
if (kondisi)
      if (kondisi)
            ....
      else if (kondisi)
            ....
      else if (kondisi) AND (kondisi)
            ....
      else
            if (kondisi)
                  ....
            else if (kondisi)
                  ....
```

Pada contoh di atas terdapat tiga blok control statement. blok 1 adalah baris pertama, blok 2 adalah baris 2-5, dan blok 3 adalah baris 6-7. Blok kondisi berbeda dengan control statement bersarang. Masing-masing blok memiliki kedudukan sejajar. Berikut contoh control statement bersarang

```
if (kondisi)
{
      statement;
      ....

      if (kondisi)
            statement;
      else if (kondisi)
            statement;
      else if (kondisi) AND (kondisi)
```

```
      {
            if (kondisi)
                  statement;
            else if (kondisi)
                  statement;
            else
                  statement;
      }
      else
            statement;
}
```

Contoh di atas memiliki blok control statement yang berbeda. Namun ketiga blok control statement tersebut tidak sama tingkatannya. Blok ketiga merupakan bagian dari blok kedua, dan blok kedua merupakan bagian dari blok pertama.

```
#include <iostream>

using namespace std;

int main()
{
    int input;
    cout << "Masukkan hari ke- = ";
    cin >> input;

    if (input == 1)
    {
        cout << "Anda memasukkan hari minggu";
    }
    else if (input == 2)
    {
        cout << "Anda memasukkan hari senin";
    }
    else if (input == 3)
    {
        cout << "Anda memasukkan hari selasa";
    }
    else if (input == 4)
    {
        cout << "Anda memasukkan hari rabu";
    }
    else if (input == 5)
    {
        cout << "Anda memasukkan hari kamis";
    }
   else if (input == 6)
   {
       cout << "Anda memasukkan hari jumat";
   }
   else if (input == 7)
   {
       cout << "Anda memasukkan hari sabtu";
   }
   else
   {
       cout << "Bukan input hari/salah masukan";
   }
   return 0;
}
```

Gambar 3. 14. Contoh Lain Else-if

Contoh ini akan kita bandingkan dengan control statement menggunakan switch ... case.

c. Switch ... Case

Control statemen menggunakan switch ... case hanya pada kondisi yang memakai operasi relasi sama dengan. Pada contoh terakhir control statement menggunakan if else, semua kondisi menggunakan operasi relasi sama dengan. Oleh karenanya untuk menyederhanakan penulisan, bisa digunakan switch ... case seperti pada contoh

```
#include <iostream>

using namespace std;

int main()
{
    int input;
    cout << "masukkan hari ke- ";
    cin >> input;

    switch(input)
    {
        case 1 : cout << "minggu";
        break;
        case 2 : cout << "senin";
        break;
        case 3 : cout << "selasa";
        break;
        case 4 : cout << "rabu";
        break;
        case 5 : cout << "kamis";
        break;
        case 6 : cout << "jumat";
        break;
        case 7 : cout << "sabtu";
        break;
        default : cout << "bukan input hari/salah masukan";
        break;
    }
    return 0;
}
```

Gambar 3. 15. Contoh switch case

switch case menyederhanakan penulisan berulang-ulang if else dan kondisi-kondisinya. Kesalahan ketik dapat diminimilasir. Selain itu pengecekan kondisi lebih mudah menggunakan switch case. Ada fungsi tambahan yaitu break. gunanya untuk keluar dari switch jika salah satu kondisi telah terpenuhi. Dengan adanya break, apabila salah satu case sudah ada yang terpenuhi maka case lain di bawahnya tidak dicek. Oleh karenanya dengan adanya break kita bisa membedakan apakah kondisi tersebut masih dalam satu blok

atau tidak.

Default merupakan pengganti dari else pada contoh sebelumnya. Jika semua kondisi tidak terpenuhi maka default akan dijalankan. Pada contoh yang berbeda

```
#include <iostream>

using namespace std;

int main()
{
    int input;
    cout << "masukkan hari ke- ";
    cin >> input;

    switch(input)
    {
        case 1 : cout << "minggu";
        case 2 : cout << "senin";
        case 3 : cout << "selasa";
        case 4 : cout << "rabu";
        case 5 : cout << "kamis";
        case 6 : cout << "jumat";
        case 7 : cout << "sabtu";
        default : cout << "bukan input hari/salah masukan";
    }
    return 0;
}
```

Gambar 3. 16. Contoh Switch-case

Karena break pada tiap-tiap pernyataan dihilangkan, maka blok dari masing masing kondisi akan meluas mencakup blok-blok kondisi yang ada di bawahnya. Jika salah satu kondisi ini terpenuhi maka statemen pada kondisi dibawahnya akan dieksekusi juga. Misalnya kita memasukkan 4, maka output akan rabu Kamis Jumat sabtu Bukan masukan hari / salat masukan, semua lima dari case 4 ke bawah.

3.3. Aktivitas

a. Mahasiswa mencoba source code yang terdapat pada gambar 3. Sampai 3.
b. Mahasiswa mengerjakan soal latihan
c. Mahasiswa mencari beberapa kasus yang dapat diselesaikan dengan if-else dan switch case

3.4. Latihan

1. Buatlah program yang bisa mengkonversi nilai angka yang di inputkan user menjadi nilai huruf, dengan kondisi:

- nilai 85 - 100, mendapat A.
- nilai 70 - 84 mendapat B.
- nilai 40 - 69 mendapat C.
- nilai 20 - 39 mendapat D.
- selain nilai tersebut mendapat E.

2. Buatlah sebuah program untuk menentukan bilangan terbesar dari 3 buah bilangan yang di inputkan oleh user.

contoh: inputkan bilangan 1 : 15
inputkan bilangan 2 : 7
inputkan bilangan 3 : 22
bilangan terbesar adalah : 22

BAB IV
STRUKTUR PERULANGAN
(*Looping*)

4.1. Tujuan

a. Menerangkan macam-macam perulangan yang didukung oleh C++
b. Menerangkan cara menggunakan perulangan di dalam program
c. Menerangkan perbedaan dari masing-masing jenis perulangan
d. Mahasiswa mampu menggunakan perulangan di dalam program
e. Mahasiswa mengetahui perbedaan dari masing-masing jenis perulangan yang ada serta mampu memilih perulangan yang akan digunakan sesuai dengan kebutuhan

4.2. Teori

4.2.1. Looping

Looping, adalah suatu bagian yang bertugas melakukan kegiatan mengulang suatu proses sesuai dengan yang diinginkan

Adapun jenis-jenis dari perulangan, antara lain :

- **for** digunakan pada cacah (counter) perulangan yang diketahui. (berapa kali perulangan tersebut akan dilakukan).
- **while**, berfungsi untuk mengulang suatu statement selama kondisi bernilai true.
- **do-while**, hampir sama dengan perulangan while, tetapi kondisi pada perulangan do-while akan dieksekusi setelah pernyataan dijalankan. Jika kondisi bernilai benar maka pernyataan dijalankan, namun jika kondisi bernilai salah maka pernyataan tidak dijalankan.

4.2.2. Kondisi Perulangan

Kondisi yang dipakai di dalam perulangan mempunyai beberapa aturan supaya tidak menimbulkan kesalahan program. Pertama, variabel kondisi perulangan harus bertipe integer atau char. Variabel bertipe float tidak dianjurkan meskipun sebenarnya bisa. Jika

terpaksa menggunakan variabel kondisi bertipe float/pecahan, sebaiknya cek logika kondisi tersebut secara teliti.

•

Khusus perulangan menggunakan for, ketiga parameter dapat dihilangkan satu, dua, atau bahkan semuanya. Namun untuk faktor berhentinya harus tetap ada. faktor tersebut dapat diletakkan di dalam statement perulangan.

4.2.3. Perulangan Bersarang (*Nested Looping*)

Nested looping, adalah perulangan yang berada di dalam perulangan lainnya. Perulangan yang lebih dalam akan diproses terlebih dahulu samapi habis, kemudian perulangan yang lebih luar baru akan bertambah, kemudian mengerjakan perulangan yang lebih dalam lagi mulai dari nilai awal hingga seterusnya. Perulangan di dalam perulangan diperbolehkan pada hampir semua bahasa pemrograman.

Contoh perulangan di dalam perulangan dapat dilihat pada program berikut :

```
#include <iostream.h>
int main()
{
    int nested1, nested2;
    int awal1 = 1, awal2;
    cout << "masukkan nilai nested1 = ";
    cin >> nested1;
    cout << "\nmasukkan nilai nested2 = ";
    cin >> nested2;

    if ((nested1 > 0) && (nested2 > 0))
    {
        while (awal1 <= nested1)
        {
            for(awal2=1; awal2 <= nested2; awal2++)
            {
                cout << "Perulangan ke-" << awal1 * awal2 << endl;
            }
            awal1++;
        }
    }
}
```

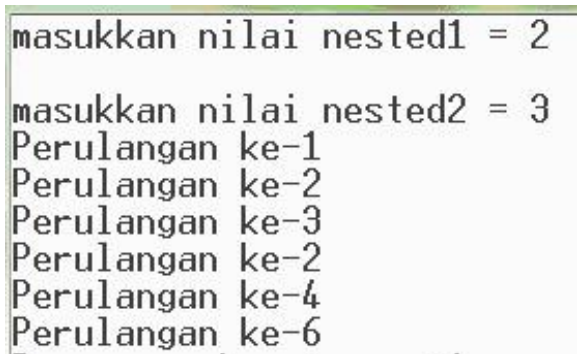
Gambar 4. 1. Source code perulangan bersarang

Pada contoh di atas, terdapat perulangan di dalam perulangan. Semua bentuk perulangan dapat diletakkan pada statement bentuk perulangan yang lain. Seperti pada

program contoh perulangan for di dalam perulangan dengan while. Statement baris ke-18, akan dijalankan sebanyak nested1 x nested2 (sesuai dengan masukan user).

Jalannya program contoh sebagai berikut. Pertama, nilai nested1 dan nested2 dimasukkan oleh user adalah 2 dan 3. Kemudian kondisi if akan dicek apakah nilai nested masing-masing berupa nilai integer positif. Jika true maka perulangan while di jalankan. Di dalam kondisi while nilai awal1 akan dicek apakah kurang dari nilai nested1 (2). Karena kondisi true maka statement while dijalankan. Di dalam statement while ditemukan perulangan lagi, maka kondisi perulangan dicek. Apakah nilai awal2 kurang dari nested2. Karena kondisi true maka statement for dijalankan. Yaitu menampilkan pesan ke layar monitor perulangan ke-...

Statement for akan dijalankan hingga nilai awal2 lebih dari nested2. Dengan kata lain perulangan for statementnya dijalankan sebanyak 3x. Sedangkan perulangan while statementnya dijalankan sebanyak 2x. Kesimpulannya total statement baris ke-18 dijalankan sebanyak 6x (2 x 3). Keluaran program sbb :



Gambar 4. 2. Output Program

•

4.3. Aktivitas

a. Dengan menggunakan perulangan FOR :

1. Task 1 : Membuat algoritma

1) Step 1 : Deklarasi variabel

Penjelasan : Mendeklarasikan variabel yang akan digunakan sebagai index untuk menampilkan deretan angka 1, misalnya i, yang bertipe integer.

```
i : integer
```

2) Step 2 : Inisialisasi

Penjelasan : Mengisi variabel i dengan nilai awal, bertujuan untuk membuat kondisi awal sebelum melakukan perulangan.

```
i ← 1
```

3) Step 3 : Proses

Penjelasan : Berisi bagian perulangan yaitu semua proses yang dilakukan secara berulang-ulang. Proses tersebut adalah menampilkan angka 1 yang diulang sebanyak 6 kali. Pada langkah ini, iterasi terjadi di dalam perulangan, yakni merupakan kondisi pertambahan agar perulangan dapat terus berjalan.

```
for i ← 1 to 6 do
    write ("1")
end for
```

4) Step 4 : Finalisasi

Penjelasan : Kondisi berhenti dari perulangan sangat penting agar tidak terjadi perulangan yang berjalan terus menerus.

2. Task 2 : Membuat program

1) Step 1 : Deklarasi variabel

```
int i;
```

2) Step 2 : Inisialisasi + Proses + Finalisasi

```
for(i=1; i<=6; i++) {
     cout<<"1";
```

}

```
#include <iostream.h>
#include <conio.h>
using namespace std;
int main (){
    int i;
    for (i=1; i<=6; i++)
    cout<<"1";
    getche();
    return 0;
}
```

Gambar 4. 3. Source Code program for

3. **Task 3 : Kompilasi dan Eksekusi program**

Gambar 4. 4. Gambar tampilan running program

b). Dengan menggunakan perulangan WHILE :

1. Task 1 : Membuat algoritma

1. **Step 1 : Deklarasi variabel**

   Penjelasan : Mendeklarasikan variabel yang akan digunakan sebagai index untuk menampilkan deretan angka 1, misalnya i yang bertipe integer.

   ```
   i : integer
   ```

2. **Step 2 : Inisialisasi**

   Penjelasan : Mengisi variabel i dengan nilai awal, bertujuan untuk membuat kondisi awal sebelum melakukan perulangan.

```
i ← 1 ;
```

3. **Step 3 : Proses**

   Penjelasan : Berisi bagian perulangan yaitu semua proses yang dilakukan secara berulang-ulang. Proses tersebut adalah menampilkan angka 1 yang diulang sebanyak 6 kali. Pada langkah ini, iterasi terjadi di dalam perulangan, yakni merupakan kondisi pertambahan agar perulangan dapat terus berjalan.

```
while (i<=6) do {
cout<<"1";
```

4. **Step 4 : Finalisasi**

   Penjelasan : Kondisi berhenti dari perulangan sangat penting agar tidak terjadi perulangan yang berjalan terus menerus.

```
i ← i+1; → iterasi
end while;
```

2. **Task 2 : Membuat program**

   1. **Step 1 : Deklarasi variabel**

```
int i=1;
```

   2. **Step 2 : Inisialisasi + Proses + Finalisasi**

```
while ((i<=6) {
      cout<<”1”;
      i=i+1;
}
```

```
#include <iostream.h>
#include <conio.h>
using namespace std;
int main(){
    int i;
    i=1;
    do {
        cout<<"1";
        i=i+1;
        }
    while (i<=6);
    getche();
return 0;
}
```

Gambar 4. 5. Source Code perulangan while

3. **Task 3 : Kompilasi dan Eksekusi program**

Gambar 4. 6. Hasil *running* program while

c). Dengan menggunakan perulangan DO-WHILE :

1. Task 1 : Membuat algoritma

1. **Step 1 : Deklarasi variabel**

   Penjelasan : Mendeklarasikan variabel yang akan digunakan sebagai index untuk menampilkan deretan angka 1, misalnya i, yang bertipe integer.

   ```
   i : integer;
   ```

2. **Step 2 : Inisialisasi**

   Penjelasan : Mengisi variabel i dengan nilai awal, bertujuan untuk membuat kondisi awal sebelum melakukan perulangan.

   ```
   i ← 1
   ```

3. **Step 3 : Proses**

   Penjelasan : Berisi bagian perulangan yaitu semua proses yang dilakukan secara berulang-ulang. Proses tersebut adalah menampilkan angka 1 yang diulang sebanyak 6 kali. Pada langkah ini, iterasi terjadi di dalam perulangan, yakni merupakan kondisi pertambahan agar perulangan dapat terus berjalan.

   ```
   repeat {
      write("1");
      i ← i+1;
   until (i=6)
   ```

4. **Step 4 : Finalisasi**

   Penjelasan : Kondisi berhenti dari perulangan sangat penting agar tidak terjadi perulangan yang berjalan terus menerus.

2. **Task 2 : Membuat program**

    1. **Step 1 : Deklarasi variabel**

```
int i=1;
```

    2. **Step 2 : Inisialisasi + Proses + Finalisasi**

```
do {
   cout<<"1";
   i=i+1;
   }
while (i<=6);
return 0;
}
```

```
#include <iostream.h>
#include <conio.h>
using namespace std;
int main(){
    int i;
    i=1;
    do {
        cout<<"1";
        i=i+1;
        }
    while (i<=6);
    getche();
return 0;
}
```

Gambar 4. 7. Source Code perulangan do-while

3. **Task 3 : Kompilasi dan Eksekusi program**

Gambar 4. 8. Output Program

4.4. Latihan

1. Buatlah program untuk menampilkan deret angka dengan menggunakan perulangan for, while, do . . while

   1 2 3 4 5 6 7 8 9

2. Buatlah program dengan menggunakan perulangan for, while dan do-while untuk menampilkan deret huruf

   A B C D E F . .

3. Buatlah program untuk menampilkan bintang, sesuai batas jumlah bilangan yang dinputkan, misalnya batas yang diinputkan yaitu 4, seperti pada Gambar berikut

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

4. Buatlah program untuk menampilkan tabel perkalian, seperti contoh :

   * 1 2 3 4 5
   1 1 2 3 4 5
   2 2 4 6 8 10
   3 3 6 9 12 15
   4 4 8 12 16 20
   5 5 10 15 20 25

BAB V
TIPE DATA LARIK (ARRAY)

5.1. Tujuan

1. Menerangkan hakekat dari array
2. Menerangkan seluk-beluk array
3. Menerangkan cara menggunakan array
4. Mahasiswa memahami pengertian dan seluk-beluk dari *array*
5. Mahasiswa mengetahui cara menggunakan array di dalam program

5.2. Teori

5.2.1. Array

Untuk menyimpan sebuah nilai dengan tipe data tertentu, digunakan sebuah variabel.
Sebagai contoh, dideklarasikan:

```
int nilai1, nilai2, nilai3;
```

Artinya terdapat 3 buah variabel yang akan digunakan untuk menyimpan data bertipe integer. Dengan jumlah data sedikit, misal 10, mungkin metode penulisan ini maish dapat digunakan namun penulisan ini menjadi kurang efektif apabila jumlah data yang akan dimasukkan banyak (misal 100), jumlah data dinamis, ataupun tidak diketahui jumlah tepatnya.
Array dapat digunakan sebagai solusi permasalahan tersebut. Berikut adalah contoh penggunaan konsep array.

Nama	Ani	Budi	Cinta	Dani	Edi	Fajar
Nilai	80	90	95	95	80	85

Gambar 5.1 Ilustrasi array pada data nilai

Dari gambar diatas, dapat dilihat terdapat dua buah variabel, Nama dan Nilai yang maisng-masing memiliki sejumlah data bertipe string dan integer. Data dalam array disebut sebagai elemen array yang ditandai dengan suatu index pada tiap-tiap elemennya. Perlu dicatat

bahwa, index array dimulai dari index ke 0. Sehingga bila kita membuat array 6 elemen, makan untuk mengisi nilai elemen pertama digunakan index ke 0.

Dengan membentuk array bertipe data integer misalnya, elemen-elemen nilai dapat disimpan dalam satu identifier yang sama. Fungsi dan ciri utama array adalah menyimpan serangkaian elemen yang bertipe sama dan mempunyai index dapat diakses secara langsung dan acak.

Dilihat dari dimensinya array dapat dibagi menjadi array satu dimensi, dan array multi dimensi. Perbedaannya multi dimensi mempunyai tipe index lebih dari satu atau tipe-komponen berupa array yang lain.

1) **Step 1 : Deklarasi dan Inisialisasi array,** Bentuk variabel dengan tipe array terlebih dahulu, meliputi nama, ukuran, dan isi tipe datanya.

```
Type nama [elemen];
atau
Type nama [elemen] =
{elemen1,elemen2,elemen_n};
```

contoh :

	1	2	3	4	5	6
Nilai	80	90	95	97	88	85

Urutan deret nilai diatas dapat dideklasarikan dengan menggunakan tipe array :

```
int nilai [6];
atau
int nilai [6] = {80,90,95,97,88,85};
```

	nilai[0]	nilai[1]	nilai[2]	nilai[3]	nilai[4]	nilai[5]
Nilai	80	90	95	97	88	85

2) **Step 2 : Pengaksesan array**, Melakukan operasi pada setiap elemen dari kumpulan data secara individual. Format pengaksesan larik didefinisikan sebagai berikut.

```
nama [index];
```

Sebagai contoh, untuk menyimpan nilai kedua yaitu 90 dari deretan data-data tersebut, maka dapat dituliskan dalam bentuk sebagai berikut

```
nilai[1] = 90;
```

Urutan elemen pada array selalu dimulai dari 0, maka nilai 90 diacu dengan nilai index ke-1, sedangkan untuk menampilkan nilai 97 pada index ke-3 dituliskan dalam bentuk

```
read nilai[3];
```

5.2.2. Array Multidimensi

Matriks, adalah array yang memiliki dua atau lebih kolom dengan banyak baris, atau dua atau lebih baris dengan banyak kolom, atau lebih singkatnya adalah array yang ada di dalam array, tergantung bagaimana penggunaanya. Misalnya ada sebuah array dua dimensi dengan ukuran 2x2 yang ditunjukkan dengan contoh sebagai berikut :

Tabel 5. 2. Ilustrasi Matriks

1,1	1,2
2,1	2,2

Contoh array 3x3:

	0	1	2
0			
1			
2			

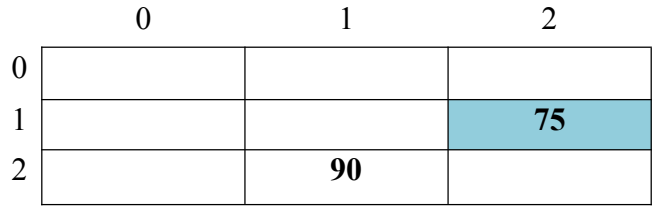
1) **Step 1 : Deklarasi dan Inisialisasi larik,**

```
Type nama [elemen_baris][elemen_kolom];
Int nilai[3][4];
```

2) **Step 2 : Pengaksesan larik,**.

```
nama [index_baris][index_kolom];
```

Sebagai contoh, untuk menyimpan nilai pada baris kedua dan kolom ketiga, maka dapat dituliskan dalam bentuk sebagai berikut.




```
nilai[1][2] = 75;
```

Untuk menampilkan nilai 90 pada baris ketiga dan kolom kedua dapat dituliskan dalam bentuk :

```
read nilai[2][1];
```

5.2.3. Array of Char

Dalam C++ tipe data string dapat diperoleh dengan cara mendefinisikan suatu array bertipe char yang diakhiri dengan karakter null(). Karena pada hakekatnya string merupakan kumpulan dari variabel bertipe char. Meskipun berupa array, perlakukan terhadap array bertipe char sedikit lebih istimewa. Karena terdapat beberapa fungsi yang bisa digunakan untuk memanipulasi langsung array ini. Sedangkan pada array bertipe non char tidak ada fungsi untuk memanipulasi. Fungsi-fungsi untuk array bertipe char diletakkan pada file include string.h.

Sama seperti dengan array biasa, string dapat didefinisikan dalam array dengan menggunakan tipe data char seperti contoh berikut.

```
1   #include <iostream.h>
2
3   using namespace std;
4
5   int main(){
6       char kata1[]={'A','L','P','H','A','B','E','T'};
7       char kata2[4];
8
9
10      for (int i=0; i<8; i++){
11          cout<<kata1[i];
12      }
13      cout<< endl;
14      for (int i=0; i<4; i++){
15          kata2[i]='A'+i;
16          cout<<kata2[i];
17      }
18      cout<< endl;
19      getchar();
20      return 0;
21  }
```

Gambar 5. 1. Source Code Array of char

Contoh pada gambar 5.1. mirip dengan contoh-contoh pada bab sebelumnya. Tipe data char masih satu jenis dengan tipe data integer. Karena tipe data char diisi oleh karakter-karakter ASCII yang dikodekan dengan angka integer dari 0 – 127. Pada perulangan pertama terlihat bahwa elemen kata1 berturut-turut diisi dengan karakter A (ASCII = 65), B (ASCII = 66), C (ASCII = 66), D (ASCII = 66), dan E (ASCII = 66). Keluaran dari program di atas adalah :

Gambar 5. 2. Hasil output array of char

5.2.3. Fungsi-Fungsi Dalam String.h

Sama seperti dengan file include yang lain, string.h terletak di /usr/include. Kita akan melihat fungsi apa saja yang didukung oleh string.h.

```
/* Copy SRC to DEST.  */
extern char *strcpy (char *__restrict __dest, __const char
```

```
*__restrict __src)
     __THROW;
/* Copy no more than N characters of SRC to DEST.  */
extern char *strncpy (char *__restrict __dest,
                      __const  char  *__restrict  __src,  size_t  __n)
__THROW;

/* Append SRC onto DEST.  */
extern  char  *strcat  (char  *__restrict  __dest,  __const  char
*__restrict __src)
     __THROW;
/* Append no more than N characters from SRC onto DEST.  */
extern  char  *strncat  (char  *__restrict  __dest,  __const  char
*__restrict __src,
                 size_t __n) __THROW;

/* Compare S1 and S2.  */
extern int strcmp (__const char *__s1, __const char *__s2)
     __THROW __attribute_pure__;
/* Compare N characters of S1 and S2.  */
extern int strncmp (__const char *__s1, __const char *__s2, size_t
__n)
     __THROW __attribute_pure__;
/* Return the length of S.  */
extern    size_t    strlen    (__const    char    *__s)    __THROW
__attribute_pure__;
__END_NAMESPACE_STD
```

Ketujuh fungsi di atas paling banyak dipakai untuk memanipulasi string. Jika Anda menggunakan Microsoft Visual C++, fungsi manipulasi string lebih banyak, seperti strstr, strupr, dan strlwr.

fungsi	***Parameter***	***Keterangan***
strcpy	source, dest	mengkopi kata dari source ke dalam dest

fungsi	*Parameter*	*Keterangan*
strncpy	source, dest, n	mengkopi kata dari source ke dalam dest sebanyak n karakter
strcat	source, dest	menambahkan string dari source ke string dest di posisi akhir
strncat	source, dest, n	menambahkan string dari source ke string dest di posisi akhir sebanyak b karakter
strcmp	s1, s2	membandingkan string dari s1 dengan string dari s2
strncmp	s1, s2, n	membandingkan string dari s1 dengan string dari s2 sebanyak n karakter
strlen	S	menghasilkan panjang dari string s

Fungsi-fungsi di atas dapat digunakan pada tipe string. Pada contoh sebelumnya, kita menampilkan kata ALPHABET dengan melakukan cout satu per satu dari elemen array. Untuk lebih praktisnya, contoh sebelumnya akan kita ganti dengan program berikut :

```
1   #include <iostream.h>
2   #include <string.h>
3
4   using namespace std;
5
6   int main(){
7       char kata1[]={'A','L','P','H','A','B','E','T'};
8       char kata2[4];
9
10      strcpy(kata2, "ABCD");
11      cout<<kata1<<endl;
12      cout<<kata2<<endl;
13
14      getchar();
15      return 0;
16  }
```

Gambar 5. 3. Source Code array of char

Terlihat bahwa string meskipun berupa array namun memiliki perlakuan yang berbeda. Pendeklarasian sekaligus pengisian array bertipe char berbeda dengan pendeklarasian sekaligus pengisian nilai array bertipe non char. Keluaran program di atas sbb :

Gambar 5. 4. Output Program

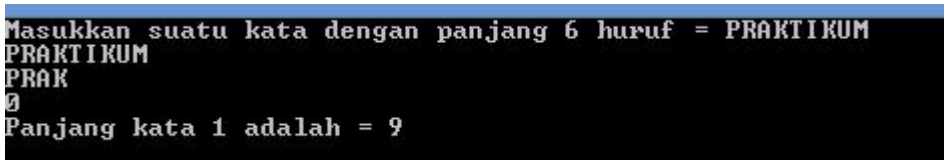
5.2.4. Menangkap Inputan Berupa Karakter

Suatu saat program kita membutuhkan input dari user berupa karakter. Untuk menangkap karakter yang dimasukkan, perintahnya sama seperti menangkap inputan biasa pada tipe data integer atau float. Berikut contoh programnya :

```
1   #include <iostream.h>
2   #include <string.h>
3
4   using namespace std;
5
6   int main(){
7       char kata1[6];
8       char kata2[15];
9
10      cout<<"Masukkan suatu kata dengan panjang 6 huruf = ";
11      cin >> kata1;
12      cout<<kata1;
13      getchar();
14
15      strncpy(kata2, kata1, 4);
16      cout<<"\n"<<kata2<<endl;
17      cout<<strncmp(kata2, kata1, 4) <<endl;
18      cout<<"Panjang kata 1 adalah = "<<strlen(kata1);
19
20
21      getchar();
22      return 0;
23  }
```

Gambar 5. 5. Source Code input karakter

Keluaran dari program Gambar 5.5. adalah :



Gambar 5. 6. Output Program

Pada program terdapat 2 buah array masing-masing kata1, dan kata2. Kata1 akan diberikan inputan oleh user, sedangkan kata2 akan mengcopy isi dari kata1.

Pada baris ke-10 program meminta masukan dari user berupa kata. Karakter yang dapat diterima hanya 6 buah. Meskipun user memasukkan sebanyak 9 karakter, maka hanya 6 karakter pertama yang disimpan. Cara menerima inputan user berupa karakter yaitu dengan perintah sebagai berikut.

```
cin >> kata1;
```

Meskipun kata1 berupa array namun pengaksesan array bertipe char dapat dengan menghilangkan informasi indeksnya, seperti ditunjukkan pada baris ke-11. Untuk menampilkan sebuah string <array bertipe char> tidak perlu menampilkan satu per satu, namun bisa menampilkan sekaligus dengan tidak menyertakan indeks array yang bersangkutan.

Untuk dapat mengcopy isi dari array1 ke array2, digunakan perintaah strncopy sepert ditunjukkan pada baris ke-15. Isi karakter pada array kata1 di copy ke array kata2 sebanyak 4 karakter saja, kemudian ditampilkan isinya.

Pada baris ke-17 array kata1 dan kata2 dibandingkan. Namun pembandingan hanya sebatas 4 karakter saja. Hasilnya adalah nilai 0 yang berarti 'true'. Selanjutnya, menampilkan ukuran dari array kata1 yang berjumlah 9 karakter.

Kesimpulan, Cara akses array bertipe char dengan non char adalah berbeda. C++ memberikan fasilitas khusus untuk array bertipe char. Hal ini untuk mempermudah programmer dalam menggunakan tipe data string. Pengaksesan dengan menghilangkan indeks array. Rahasianya adalah, setiap array char yang kita buat secara otomatis akan

ditambahkan oleh C++ karakter null yang berarti tanda akhir string. Seperti array kata1, oleh C++ alokasi memorinya menjadi :

P	R	A	K	T	I	K	U	M	/0

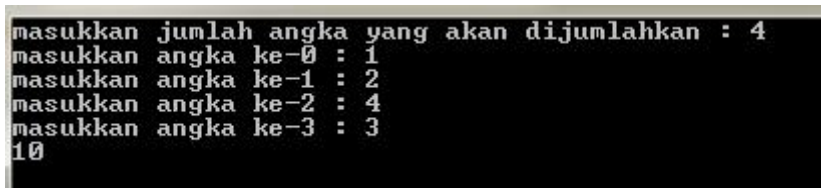
Karakter terakhir adalah tanda akhir string dari kata1. Oleh karenanya apabila dilakukan *cout* maka C++ akan melakukan perulangan dengan menampilkan isi array kata1 hingga dijumpai karakter null. Makanya kenapa indeks array pada saat pengaksesan array bertipe char dapat dihilangkan. Karena patokannya kepada karakter null.

5.3. Aktivitas

1. Mahasiswa mencoba source yang terdapat pada gambar
2. Mahasiswa mengerjakan soal latihan
3. Mahasiswa mencari kasus program yang menggunakan tipe data array

5.4. Latihan

1. Buatlah sebuah program penjumlahan dengan angka yang dijumlahkan sesuai dengan masukan pengguna.



2. Buatlah sebuah program untuk konversi bilangan decimal ke biner. Hasil konversi bisa langsung ditampilkan atau disimpan ke dalam sebuah variabel array of char

BAB VI
TIPE DATA STRUKTUR

6.1. Tujuan

1. Menerangkan hakekat dari struktur
2. Menerangkan seluk-beluk struktur
3. Menerangkan cara menggunakan struktur
4. Mahasiswa memahami pengertian dan seluk-beluk dari struktur
5. Mahasiswa mengetahui cara menggunakan struktur di dalam program

6.2. Teori

Struct atau tipe data striktur merupakan pengembangan dari array. Struct dapat digunakan untuk berbagai tipe data yang berbeda. Sebagai contoh, sebuah record mengenai suatu produk, terdiri dari kode produk, nama produk, harga produk dsb. Semua data tersebut dihimpun dalam satu record. Berikut merupakan langkah-langkah penggunaan struct.

1. Deklarasi struct

Bentuk suatu record terlebih dahulu, meliputi field- field yang ada didalam record beserta tipe datanya untuk masing-masing field.

```
struct namaStruct {
tipedata namafield1;
tipedata namafield2;
tipedata namafield3;
};
```

Contoh:

```
struct ProductRec {
string name;
string idNum;
float price;
};
```

Bentuk variabel dengan tipe record tersebut.

```
namaStruct namaVariabel;
```

Contoh:

```
ProductRec theProduct;
```

2. Pengaksesan struct

Melakukan operasi pada setiap elemen dari record secara individual. Misalnya operasi pengisian nilai pada tiap-tiap elemen. Nilai-nilai tertentu dapat di-assign, dengan aturan pengacuan terhadap field dari sebuah record.

```
nama_variabel.nama_field = nilai;
```

Contoh:

```
theProduct.name = "Orange";
atau
cin >> theProduct.name;
```

Menampilkan data yang ada didalam record .

```
cout<<nama_variabel.nama_field;
```

Contoh:

```
cout<< theProduct.price;
```

3. Nested struct

Elemen suatu struct juga bisa berada di dalam struct lainnya. Sebagai contoh dapat dilihat di bawah ini.

```
struct productBrand {
string productName;
string brandName;
};

struct ProductRec {
productBrand name;
string idNum;
float price;
};
```

4. Array of Struct

Elemen suatu array juga bisa berupa struct. Sebagai contoh dapat dilihat di bawah ini:

```
struct productBrand {
string productName;
string brandName;
};

struct ProductRec {
productBrand name;
string idNum;
float price;
};

ProductRec theProduct[10];
```

6.3. Aktivitas Praktikum

1. Task 1 : Membuat program membuat data pribadi dengan struct.

1. Step 1 : Deklarasi struct

Deklarasikan struct **fullname** dan struct **StudentRec**. Struct **fullname** akan digunakan untuk menyimpan string **firstname** dan string **lastname**. Sementara struct StudentRec akan berisi name dari struct **fullname**, strng **idNum** dan float **gpa.**

```
5   //deklarasi
6   struct fullname {
7       string firstname;
8       string lastname;
9   };
10  struct StudentRec {
11      fullname name;
12      string idNum;
13      float gpa;
14  };
```

2. Step 2 : Deklarasi dan Inisialisasi variabel

```
15  StudentRec theStudent[10];
```

3. Step 3 : Proses

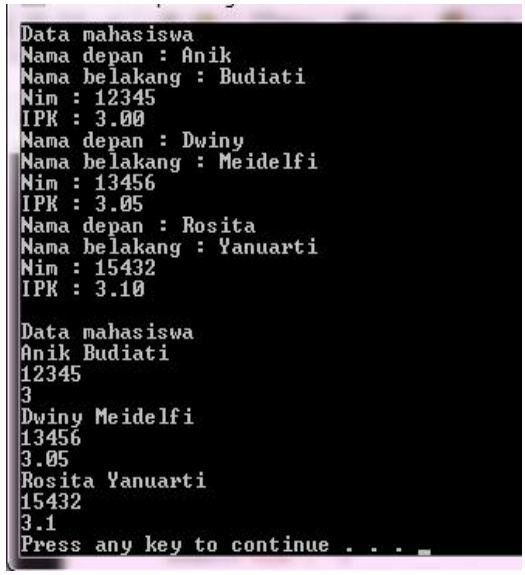
Pada langkah ini, buatlah inputan yang akan memasukkan jumlah mahasiswa yang akan direcord. Selanjutnya, buat input untuk firstname, lastname, NIM, dan IPK. Simpan masing-masing inputan pada sruct yang telah dibuat.

```
18  int main() {
19          int n;
20          //proses
21          cout<<"Masukkan banyaknya mahasiswa : ";
22          cin>>n;
23          cout<<"Data mahasiswa"<<endl;
24          for (int i=0;i<n;i++) {
25          cout<<"Nama depan : ";
26          cin>>theStudent[i].name.firstname;
27          cout<<"Nama belakang : ";
28          cin>>theStudent[i].name.lastname;
29          cout<<"NIM : ";
30          cin>>theStudent[i].idNum;
31          cout<<"IPK : ";
32          cin>>theStudent[i].gpa;
33          }
34          cout<<endl;
35          getchar();
```

4. Step 4 : Finalisasi

```
37          //finalisasi
38          cout<<"Data mahasiswa"<<endl;
39          for (int i=0;i<n;i++) {
40          cout<<theStudent[i].name.firstname<<endl;
41          cout<<theStudent[i].name.lastname<<endl;
42          cout<<theStudent[i].idNum<<endl;
43          cout<<theStudent[i].gpa<<endl;
44          }
45          cout<<endl;
46          getchar();
47          return 0;
48  }
```

Task 2 : Kompilasi dan Eksekusi program



Gambar 6.2 Tampilan *running* program

6.4. Latihan

Buatlah sebuah program pencatatan barang pada swalayan dengan memanfaatkan array dan struct. Program tersebut meminta input berupa kode barang, nama barang, harga barang, dan jumlah barang. Buatlah input barang sejumlah "n" masukan (n ditentukan oleh pengguna sendiri, maksimal 50), kemudian **tampilkan jumlah semua barang dan harga rata-rata barang!**

BAB VII
SUBPROGRAM DAN FUNGSI

7.1. Tujuan

a. Mahasiswa mampu membedakan fungsi dengan nilai balik dan fungsi tanpa nilai balik

b. Mahasiswa mampu mengimplementasikan rekursi dalam kasus program

7.2. Teori

Dalam C++ program merupakan kumpulan dari fungsi-fungsi, baik itu yang didefinisikan langsung dalam program maupun yang disimpan dalam suatu file header. C++ sendiri mempunyai fungsi utama yang disebut fungsi main(). Fungsi main() ini selalu ada dalam setiap program C++ dan kompiler akan menjalankan program melalui perintah-perintah yang terdapat dalam fungsi ini.

Fungsi merupakan subprogram dan berguna untuk menjadikan program dapat lebih bersifat modular sehingga akan mudah dipahami dan dapat digunakan kembali, baik untuk program itu sendiri maupun untuk program lain yang memiliki proses yang sama.

Sebuah fungsi berisi sejumlah pernyataan yang dikemas dalam sebuah nama. Selanjutnya nama ini dapat dipanggil beberapa kali di beberapa tempat dalam program. Fungsi memudahkan dalam mengembangkan program dan menghemat ukuran program.

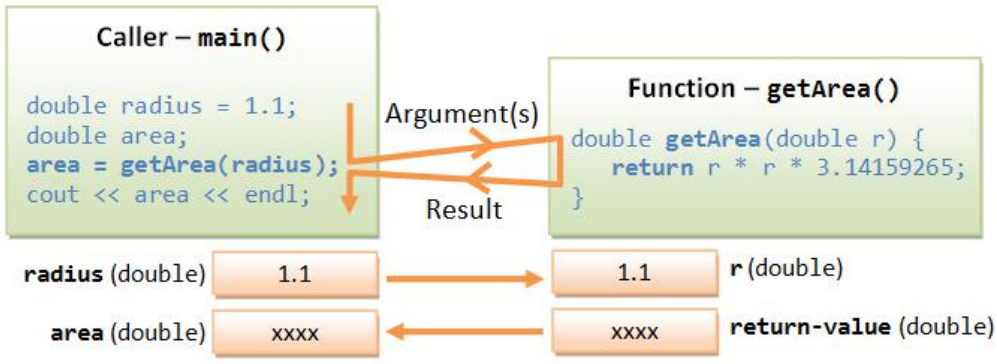
7.3 Fungsi Pertama

Pada contoh yang pertama kita akan menghitung luas lingkaran dengan menggunakan sebuah fungsi getArea(). Kodenya dapat dilihat pada potongan kode di Gambar 7.1 berikut.

```cpp
#include <iostream>
using namespace std;
const int PI = 3.14159265;

// Function Prototype (Function Declaration)
double getArea(double radius);

int main() {
   double radius1 = 1.1, area1, area2;
   // call function getArea()
   area1 = getArea(radius1);
   cout << "area 1 is " << area1 << endl;
   // call function getArea()
   area2 = getArea(2.2);
   cout << "area 2 is " << area2 << endl;
   // call function getArea()
   cout << "area 3 is " << getArea(3.3) << endl;
}

// Function Definition
// Return the area of a circle given its radius
double getArea(double radius) {
   return radius * radius * PI;
}
```

Gambar 7.1. Fungsi Pertama

Fungsi getArea() sebelumnya di definisikan terlebih dahulu, yaitu berisi rumus luas lingkaran. Dengan kemudian dipanggil pada fungsi main. Penjelasannya dapat dilihat pada Gambar 7.2



Gambar 7.2. Penjelasan Fungsi

7.4 Penjelasan Fungsi

7.4.1 Definisi Fungsi

Bentuk fungsi secara umum dapat dibuat sebagai berikut:

```
returnValueType functionName (parameterList) {
   functionBody;
}
```

- ✓ *parameterList* berisi daftar parameter yang dipisahkan dengan koma.
- ✓ *returnValueType* merupakan tipe kembalian dari fungsi. Jika fungsi mempunyai kembalian maka returnValueType nya merupakan sebuah tipe data kembalian, jika tidak mempunyai kembalian maka returnValueType nya void

7.4.2 Pernyataan "return"

Jika fungsi memiliki sesuatu kembalian maka didalam fungsinya wajib ada sintaks return yang akan mengembalikan sebuah ekspresi sesuai dengan *returnValueType* nya.

```
return expression;
```

7.4.3 Prototipe Fungsi

Pada Gambar 6.1 di baris kode ke 6 terdapat sebuah prototipe fungsi. Pada C++ fungsi harus di deklarasikan terlebih dahulu sebelum dipanggil. Bentuk prototipe fungsi secara umum adalah sebagai berikut:

```
returnValueType functionName (parameterList)
```

Contoh lain dari prototipe fungsi dapat dilihat pada potongan kode di Gambar 7.3 berikut. Protipe fungsi berada pada baris kode ke 4 yaitu fungsi maximum.

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

int maximum(int, int); // Function prototype (declaration)

int main() {
   cout << maximum(5, 8) << endl; // Call maximum() with literals

   int a = 6, b = 9, c;
   c = maximum(a, b);             // Call maximum() with variables
   cout << c << endl;

   cout << maximum(c, 99) << endl; // Call maximum()
}

// Function definition
// A function that returns the maximum of two given int
int maximum(int num1, int num2) {
   return (num1 > num2) ? num1 : num2;
}
```

Gambar 7.3. Contoh Fungsi Prototipe

7.5 Fungsi Tanpa Kembalian (Void)

Fungsi dapat juga tidak memiliki nilai kembalian. Fungsi ini hanya menjalankan serangkaian operasi tanpa perlu adanya kembalian ke pemanggilnya. Contoh fungsi void dapat dilihat pada Gambar 6.4 berikut. Pada potongan kode di Gambar 7.4 tersebut juga diperlihatkan bahwa fungsi tidak wajib memiliki prototipe jika definisi fungsi diletakkan di atas blok program main atau pemanggil fungsinya.

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

void printmessage ()
{
  cout << "I'm a function!";
}

int main ()
{
  printmessage ();
}
```

Gambar 7.4. Fungsi Void

7.6 Penggunaan Parameter

Terdapat dua cara parameter dapat diberikan kepada sebuah fungsi yaitu dengan pass by value, atau dengan pass by reference.

7.6.1 Pass By Value

Di dalam pass by value parameter diberikan tanpa merubah nilai dari nilai atau variabel aslinya. Parameter yang berada di dalam fungsi adalah sebuah "copy" nilai dari nilai yang diberikan pada pemanggil fungsinya. Untuk dapat lebih jelasnya dapat dilihat pada potongan kode pada Gambar 7.5 berikut.

```
1  #include <iostream>
2  using namespace std;
3
4  // Function prototypes
5  int inc(int number);
6
7  // Test Driver
8  int main() {
9     int n = 8;
10    cout << "Before calling function, n is " << n << endl; // 8
11    int result = inc(n);
12    cout << "After calling function, n is " << n << endl;  // 8
13    cout << "result is " << result << endl;                // 9
14 }
15
16 // Function definitions
17 // Return number+1
18 int inc(int number) {
19    ++number;  // Modify parameter, no effect to caller
20    return number;
21 }
```

Gambar 7.5. Pass By Value

7.6.2 Pass By Reference

Di dalam pass by reference variabel yang dijadikan parameter pada saat pemanggilan fungsi akan ikut terubah nilainya. Untuk dapat lebih jelas dapat dilihat pada potongan kode Gambar 7.6. Potongan kode tersebut hampir serupa dengan potongan kode pada Gambar 6.5, hanya berbeda pada definisi parameter yang menggunakan tanda reference (&). Jika dieksekusi variabel n akan ikut terubah setelah pemanggilan fungsi inc.

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

// Function prototypes
int inc(int &number);

// Test Driver
int main() {
   int n = 8;
   cout << "Before calling function, n is " << n << endl; // 8
   int result = inc(n);
   cout << "After calling function, n is " << n << endl;  // 9
   cout << "result is " << result << endl;                // 9
}

// Function definitions
// Return number+1
int inc(int &number) {
   ++number;  // Modify parameter, no effect to caller
   return number;
}
```

Gambar 7.6. Pass By Reference

BAB VIII
ALGORITMA PENGURUTAN & ALGORITMA PENCARIAN

8.1. Tujuan

1. Menerangkan algoritma-algoritma pengurutan
2. Menerangkan algoritma-algoritma pencarian
3. Mahasiswa mampu menjelaskan langkah-langkah pada algoritma pengurutan
4. Mahasiswa memahami algoritma pencarian

8.2. Teori

8.2.1. Pengurutan

Pengurutan data merupakan hal yang penting dalam kehidupan nyata untuk memudahkan pengelolaan data. Pengurutan dapat dilakukan dengan urutan menaik atau dengan urutan menurun. Sebagai contoh, jika ada data angka sebagai berikut.

5	3	7	2	0	9	4	1	8	6

Jika data diurutkan secara menaik akan menjadi sebagai berikut.

0	1	2	3	4	5	6	7	8	9

Jika data diurutkan secara menurun akan menjadi sebagai berikut.

9	8	7	6	5	4	3	2	1	0

Ada beberapa metode pengurutan, yaitu:

- Metode penyisipan (*insertion sort*)
- Metode seleksi (*selection sort*)
- Metode gelembung (*bubble sort*)

Pada pembahasan di bawah ini, akan dijelaskan algoritma-algoritma pengurutan, serta penerapan di dalam C++.

8.2.1.1. Metode Penyisipan (*Insertion Sort*)

Metode penyisipan langsung adalah metode pengurutan yang mengambil sebuah data sisip pada data yang diurutkan dan menggeser data yang lebih besar dari data sisip agar data sisip

dapat ditempatkan pada tempat yang benar. Sebagai contoh, jika ada sebuah larik yang berisi angka-angka sebagai berikut.

5	3	7	2	0	9	4	1	8	6

Jika data di atas akan diurutkan dengan urutan menaik menggunakan metode penyisipan langsung maka prosesnya sebagai berikut.

Data Sisip	**Hasil Pengurutan**
3	5 3 7 2 0 9 4 1 8 6 Pada perulangan ke-1, data indeks 1 dijadikan sebagai data sisip, kemudian dibandingkan dengan data sebelumnya. Terdapat satu data yang lebih besar daripada data sisip, maka satu data tersebut harus digeser satu tempat dan data sisip dipindah ke tempat paling depan.
7	3 5 7 2 0 9 4 1 8 6 Pada perulangan ke-2, data indeks 2 dijadikan sebagai data sisip, kemudian dibandingkan dengan data sebelumnya. Jika data sebelumnya tidak ada yang lebih besar dari data sisip maka tidak ada data yang harus digeser ke belakang.
2	3 5 7 2 0 9 4 1 8 6 Pada perulangan ke-3, data indeks 3 dijadikan sebagai data sisip, kemudian dibandingkan dengan data sebelumnya. Terdapat tiga data yang lebih besar daripada data sisip, maka tiga data tersebut harus digeser satu tempat dan data sisip dipindah ke tempat paling depan.
0	2 3 5 7 0 9 4 1 8 6 Pada perulangan ke-4, data indeks 4 dijadikan sebagai data sisip, kemudian dibandingkan dengan data sebelumnya. Terdapat empat data yang lebih besar daripada data sisip, maka empat data tersebut harus digeser satu tempat dan data sisip dipindah ke tempat paling depan.
9	0 2 3 5 7 9 4 1 8 6 Pada perulangan ke-5, data indeks 5 dijadikan sebagai data sisip, kemudian dibandingkan dengan data sebelumnya. Jika data sebelumnya tidak ada yang lebih besar dari data sisip maka tidak ada data yang harus digeser ke belakang.
4	0 2 3 5 7 9 4 1 8 6

<table>
<tr><td></td><td>Pada perulangan ke-6, data indeks 6 dijadikan sebagai data sisip, kemudian dibandingkan dengan data sebelumnya. Terdapat tiga data yang lebih besar daripada data sisip, maka tiga data tersebut harus digeser satu tempat dan data sisip dipindah ke tempat sebelum tiga data yang digeser.</td></tr>
<tr><td>1</td><td><table><tr><td>0</td><td>2</td><td>3</td><td>4</td><td>5</td><td>7</td><td>9</td><td>1</td><td>8</td><td>6</td></tr></table>Pada perulangan ke-7, data indeks 7 dijadikan sebagai data sisip, kemudian dibandingkan dengan data sebelumnya. Terdapat enam data yang lebih besar daripada data sisip, maka enam data tersebut harus digeser satu tempat dan data sisip dipindah ke tempat sebelum enam data yang digeser.</td></tr>
<tr><td>8</td><td><table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>7</td><td>9</td><td>8</td><td>6</td></tr></table>Pada perulangan ke-8, data indeks 8 dijadikan sebagai data sisip, kemudian dibandingkan dengan data sebelumnya. Terdapat satu data yang lebih besar daripada data sisip, maka satu data tersebut harus digeser satu tempat dan data sisip dipindah ke tempat sebelum satu data yang digeser.</td></tr>
<tr><td>6</td><td><table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>7</td><td>8</td><td>9</td><td>6</td></tr></table>Pada perulangan ke-9, data indeks 9 dijadikan sebagai data sisip, kemudian dibandingkan dengan data sebelumnya. Terdapat tiga data yang lebih besar daripada data sisip, maka tiga data tersebut harus digeser satu tempat dan data sisip dipindah ke tempat sebelum tiga data yang digeser.</td></tr>
<tr><td>Hasil akhir</td><td><table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr></table></td></tr>
</table>

Langkah-langkah di atas jika dituangkan di dalam C++ menjadi sebagai berikut.

```
1   #include<iostream>
2   using namespace std;
3   int main(){
4       int data[]={5,3,7,2,0,9,4,1,8,6};
5       int i,dataSize=sizeof(data)/sizeof(data[0]),temp;
6       for(int j=1;j<dataSize;j++){
7           i=j-1;
8           temp=data[j];
9           while(data[i]>temp&&i>=0){
10              data[i+1]=data[i];
11              i--;
12          }
13          data[i+1]=temp;
14      }
15      for(int j=0;j<dataSize;j++){
16          cout<<j<<" ";
17      }
18      return 0;
19  }
```

Gambar 8. 1. Source Code Insertion Sort

8.2.1.2. Metode Seleksi (*Selection Sort*)

Metode seleksi adalah metode pengurutan yang mencari nilai terkecil atau terbesar bergantung pada pengurutan menaik atau menurun yang kemudian ditempatkan pada tempat paling depan, kemudian mencari lagi nilai terkecil atau terbesar kedua sepanjang jumlah elemen larik dikurangi satu. Setelah ditemukan, elemen kedua ditukar dengan nilai tersebut, begitu seterusnya. Sebagai contoh, jika ada sebuah larik yang berisi angka-angka sebagai berikut.

5	3	7	2	0	9	4	1	8	6

Jika data di atas akan diurutkan dengan urutan menaik menggunakan metode seleksi maka prosesnya sebagai berikut.

Nilai Terkecil	Hasil Pengurutan
0	5 3 7 2 0 9 4 1 8 6 Pada perulangan ke-1, dicari nilai terkecil dari elemen larik indeks 0-9 dan ditemukan nilai 0 sebagai nilai terkecil, kemudian tempat nilai 0 ditukar dengan elemen indeks 0.
1	0 3 7 2 5 9 4 1 8 6 Pada perulangan ke-2, dicari nilai terkecil dari elemen larik indeks 1-9 dan ditemukan nilai 1 sebagai nilai terkecil, kemudian tempat nilai 1 ditukar dengan elemen indeks 1.
2	0 1 7 2 5 9 4 3 8 6 Pada perulangan ke-3, dicari nilai terkecil dari elemen larik indeks 2-9 dan ditemukan nilai 2 sebagai nilai terkecil, kemudian tempat nilai 2 ditukar dengan elemen indeks 2.
3	0 1 2 7 5 9 4 3 8 6 Pada perulangan ke-4, dicari nilai terkecil dari elemen larik indeks 3-9 dan ditemukan nilai 3 sebagai nilai terkecil, kemudian tempat nilai 3 ditukar dengan elemen indeks 3.
4	0 1 2 3 5 9 4 7 8 6 Pada perulangan ke-5, dicari nilai terkecil dari elemen larik indeks 4-9 dan ditemukan nilai 4 sebagai nilai terkecil, kemudian tempat nilai 4 ditukar dengan elemen indeks 4.
5	0 1 2 3 4 9 5 7 8 6 Pada perulangan ke-6, dicari nilai terkecil dari elemen larik indeks 5-9 dan ditemukan nilai 5 sebagai nilai terkecil, kemudian tempat nilai 5 ditukar dengan elemen indeks 5.
6	0 1 2 3 4 5 9 7 8 6 Pada perulangan ke-7, dicari nilai terkecil dari elemen larik indeks 6-9 dan ditemukan nilai 6 sebagai nilai terkecil, kemudian tempat nilai 6 ditukar dengan elemen indeks 6.
7	0 1 2 3 4 5 6 7 8 9 Pada perulangan ke-8, dicari nilai terkecil dari elemen larik indeks 7-9 dan ditemukan nilai 7 sebagai nilai terkecil. Elemen telah sesuai dengan tempatnya maka tidak perlu ditukar.
8	0 1 2 3 4 5 6 7 8 9 Pada perulangan ke-9, dicari nilai terkecil dari elemen larik indeks 8-9 dan ditemukan nilai 8 sebagai nilai terkecil. Elemen telah sesuai dengan tempatnya maka tidak perlu ditukar.
Hasil akhir	0 1 2 3 4 5 6 7 8 9

Langkah-langkah di atas jika dituangkan di dalam C++ menjadi sebagai berikut.

```
#include<iostream>
using namespace std;
int main(){
    int data[]={5,3,7,2,0,9,4,1,8,6};
    int dataSize=sizeof(data)/sizeof(data[0]),min,temp;
    for(int i=0;i<dataSize-1;i++){
        min=i;
        for(int j=i+1;j<dataSize;j++){
            if(data[j]<data[min]){
                min=j;
            }
        }
        temp=data[i];
        data[i]=data[min];
        data[min]=temp;
    }
    for(int i=0;i<dataSize;i++){
        cout<<i<<" ";
    }
    return 0;
}
```

Gambar 8. 2. Source Code Selection Sort

8.2.1.3. Metode Gelembung (*Bubble Sort*)

Metode gelembung adalah metode pengurutan yang menukarkan dua buah elemen secara terus menerus sampai pengurutan selesai. Sebagai contoh, jika ada sebuah larik yang berisi angka-angka sebagai berikut.

5	3	7	2

Jika data di atas akan diurutkan dengan urutan menaik menggunakan metode gelembung maka prosesnya sebagai berikut.

5	3	7	2

Perulangan ke-1 membandingkan elemen indeks 0 dengan indeks 1, karena elemen indeks 0 lebih besar dari elemen indeks 1 maka dilakukan pertukaran.

3	5	7	2

Perulangan ke-2 membandingkan elemen indeks 1 dengan indeks 2, karena elemen indeks 1 tidak lebih besar dari elemen indeks 2 maka tidak ada pertukaran.

3	5	7	2

Perulangan ke-3 membandingkan elemen indeks 2 dengan indeks 3, karena elemen indeks 2 lebih besar dari elemen indeks 3 maka dilakukan pertukaran.

3	5	2	7

Perulangan ke-4 membandingkan elemen indeks 0 dengan indeks 1, karena elemen indeks 0 tidak lebih besar dari elemen indeks 1 maka tidak ada pertukaran.

3	5	2	7

Perulangan ke-5 membandingkan elemen indeks 1 dengan indeks 2, karena elemen indeks 1 lebih besar dari elemen indeks 2 maka dilakukan pertukaran.

3	2	5	7

Perulangan ke-6 membandingkan elemen indeks 2 dengan indeks 3, karena elemen indeks 2 tidak lebih besar dari elemen indeks 3 maka tidak ada pertukaran.

3	2	5	7

Perulangan ke-7 membandingkan elemen indeks 0 dengan indeks 1, karena elemen indeks 0 lebih besar dari elemen indeks 1 maka dilakukan pertukaran.

2	3	5	7

Perulangan ke-8 membandingkan elemen indeks 1 dengan indeks 2, karena elemen indeks 1 tidak lebih besar dari elemen indeks 2 maka tidak ada pertukaran.

2	3	5	7

Perulangan ke-9 membandingkan elemen indeks 2 dengan indeks 3, karena elemen indeks 2 tidak lebih besar dari elemen indeks 3 maka tidak ada pertukaran.

Langkah-langkah di atas jika dituangkan di dalam C++ menjadi sebagai berikut.

```
1  #include<iostream>
2  using namespace std;
3  int main(){
4      int data[]={5,3,7,2,0,9,4,1,8,6};
5      int dataSize=sizeof(data)/sizeof(data[0]),temp;
6      for(int i=0;i<dataSize-1;i++){
7          for(int j=0;j<dataSize-1;j++){
8              if(data[j]>data[j+1]){
9                  temp=data[j];
10                 data[j]=data[j+1];
11                 data[j+1]=temp;
12             }
13         }
14     }
15     for(int i=0;i<dataSize;i++){
16         cout<<i<<" ";
17     }
18     return 0;
19 }
```

Gambar 8. 3. Source Code Bubble Sort

8.2.1.4 Metode Penggabungan MERGE SORT

Metode pengurutan merge sort adalah metode pengurutan lanjut, sama dengan metode Quick Sort. Metode ini juga menggunakan konsep devide and conquer yang membagi data S dalam dua kelompok yaitu S1 dan S2 yang tidak beririsan (disjoint). Proses pembagian data dilakukan secara rekursif sampai data tidak dapat dibagi lagi atau dengan kata lain data dalam sub bagian menjadi tunggal. Setelah data tidak dapat dibagi lagi, proses penggabungan (merging) dilakukan antara sub-sub bagian dengan memperhatikan urutan data yang diinginkan (ascending/kecil ke besar atau descending/besar ke kecil). Proses penggabungan ini dilakukan sampai semua data tergabung dan terurut sesuai urutan yang diiginkan. Secara umum, algoritma merge sort dapat diimplementasikan secara rekursif. Fungsi rekursif adalah sebuah fungsi yang didalam implementasinya memanggil dirinya sendiri. Pemanggilan diri sendiri ini berakhir jika kondisi tertentu terpenuhi (terminated condition is true).

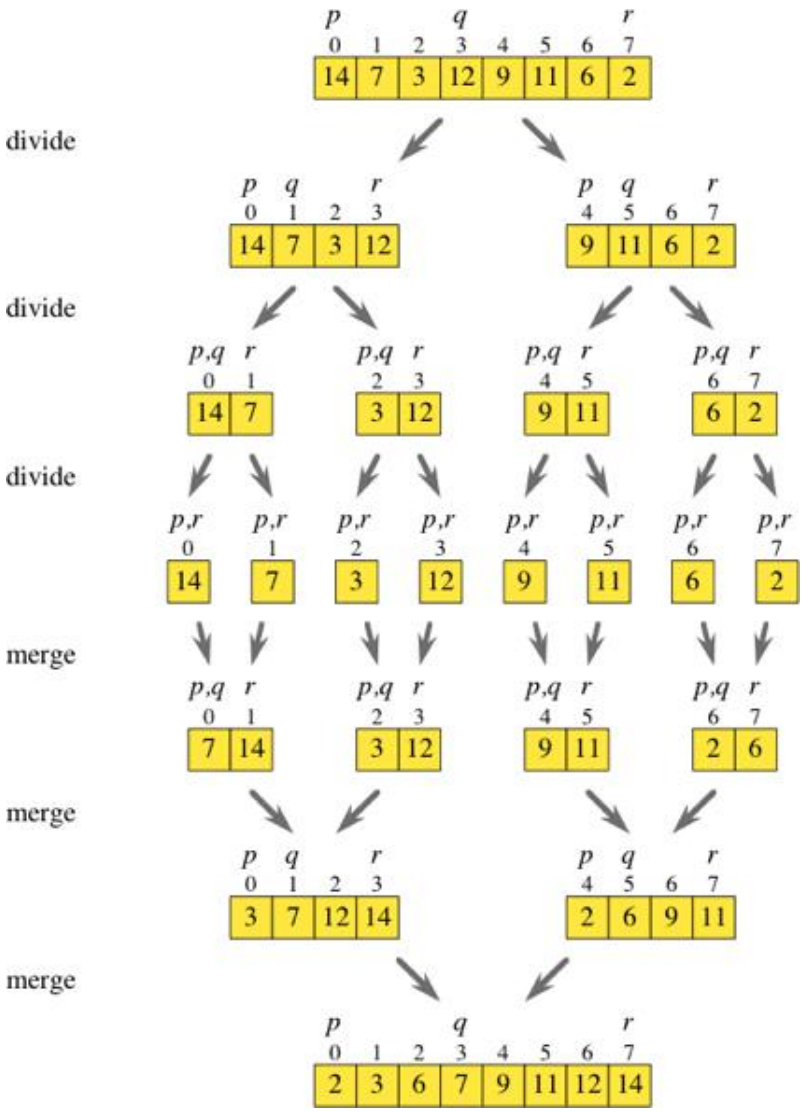
Langkah-langkah merge sort adalah sebaai berikut:

1. Bagi menjadi dua bagian dengan cara menemukan Q yang berada diantara P dan R.
2. Mengurutkan secara rekursif sub array di tiap dua sub problem yang dihasilkan oleh langkah satu.
3. Gabungkan dua sub array yang telah di urutkan menjadi satu array.

Sebagai contoh, jika ada sebuah larik yang berisi angka-angka sebagai berikut.

14	7	3	12	9	11	6	2

Jika data di atas akan diurutkan dengan urutan menaik menggunakan metode merge sort maka prosesnya sebagai berikut.



Gambar 8. 4. Animasi Merge Sort

Langkah-langkah di atas jika dituangkan di dalam C++ menjadi sebagai berikut:

```
1   #include <iostream>
2   #include<stdio.h>
3
4   using namespace std;
5
6   void merge_sort(int low,int high)
7   {
8       int mid;
9       if(low<high)
10      {
11          mid = low + (high-low)/2;
12          merge_sort(low,mid);
13          merge_sort(mid+1,high);
14          merge(low,mid,high);
15      }
16  }
17
```

Gambar 8. 5. Source Code Merge Sort

8.2.1.5 Metode Cepat (*QUICK SORT*)

Quicksort merupakan Algoritma Pembagi. Merupakan membandingkan suatu elemen (disebut juga pivot) dengan elemen yang lain dan menyusunnya sedemikian rupa sehingga elemen-elemen lainnya yang lebih kecil daripada pivot tersebut terletak disebelah kirinya dan elemen-elemen lain yang lebih besar daripada pivot terletak disebelah kanannya.

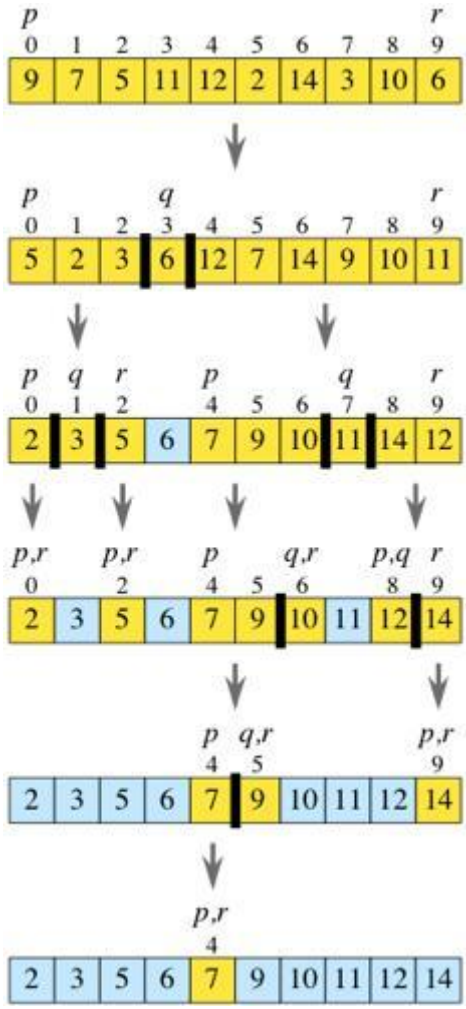
Dengan demikian telah terbentuk dua sublist, yang terletak di sebelah kiri dan kanan dari pivot. Lalu pada sublist kiri dan sublist kanan anggap sebuah list baru dan kerjakan proses yang sama seperti sebelumnya. Demikian seterusnya sampai tidak terdapat sublist lagi sehingga didalamnya terjadi sebuah proses rekursif.

Langkah-langkah merge sort adalah sebaai berikut:

1. Bagi dengan memilih suatu elemen pada subarray [p..r]. Elemen ini disebut sebagai pivot. Atur elemen pada array [p..r] sehingga semua elemen pada array [p..r] lain lebih kecil atau sama dengan pivot yang ada di kiri dan semua elemen didalamnya.
2. Urutkan dengan cara rekursif array[p..q-1](semua elemen yang ada di kiri pivot yang harus lebih kecil atau sama dengan pivot) dan array[q+1..r](semua elemen yang ada di kanan pivot yang harus lebih besar dari pivot).

Sebagai contoh, jika ada sebuah larik yang berisi angka-angka sebagai berikut.

9	7	5	11	12	2	14	3	10	6



Gambar 8. 6. Animasi Quick Sort

Langkah-langkah di atas jika dituangkan di dalam C++ menjadi sebagai berikut:

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

using namespace std;

void QuickSort(int * Data, int a,int b )
{
    int al,bl,pivot;
    al = a; bl = b;
    pivot = Data[(a+b) / 2];
    while (!(al>bl))
    {
        while(Data[al] < pivot) al++;
        while(Data[bl] > pivot) bl--;
        if (al <= bl)
        {
            Tukar(Data[al],Data[bl]);
            al++; bl--;
        }
    }
    if (a<bl) QuickSort(Data,a,bl);
    if (al<b) QuickSort(Data,al,b);
}

```

Gambar 8. 7. Source Code Quick Sort

8.2.2. Pencarian

Pencarian suatu data pada sekumpulan data merupakan proses yang sangat penting dalam kehidupan nyata. Seperti halnya pengurutan, pencarian juga dapat dilakukan dengan beberapa metode pencarian seperti metode pencarian beruntun (*sequential search*) dan metode bagi dua (*binary search*) yang akan dibahas.

8.2.2.1 Pencarian Beruntun (*Sequential Search*)

Pencarian beruntun dapat dilakukan pada data yang belum terurut maupun yang sudah terurut. Pencarian beruntun dilakukan dengan melakukan penelusuran data satu persatu kemudian dicocokkan dengan data yang dicari, jika tidak sama maka penelusuran dilanjutkan, jika sama maka penelusuran dihentikan, berarti data telah ditemukan. Jika ada data sebagai berikut.

Nomor Induk	Nama	Nilai
10101	Adi	64.75
10103	Budi	75.11
10105	Charli	84.63
10102	Dodi	77.07
10104	Edi	66.70

Akan dicari nilai dari Charli dalam hal ini data yang dipunyai adalah nomor induk Charli yaitu “10105”. Jika pencarian beruntun dituangkan di dalam C++ menjadi sebagai berikut.

```
#include<iostream>
using namespace std;
int main(){
    bool found=false;
    char* nama[]={"Nana","Rudi","Dea","Ihsan","Tiara"};
    char* nomorInduk[]={"13507701","13507702","13507703","13507704","13507705"};
    char* query="13507703";
    float nilai[]={64.75,75.11,84.63,77.07,66.70};
    for(int i=0;i<5;i++){
        if(nomorInduk[i]==query){
            cout<<nama[i]<<","<<nomorInduk[i]<<","<<nilai[i]<<endl;
            found=true;
        }
    }
    if(!found){
        cout<<"Tidak ditemukan.";
    }
    return 0;
}
```

Gambar 8. 8. Source Code Binari

8.2.2.2. Pencarian Bagi Dua (*Binary Search*)

Pencarian bagi dua, atau pencarian biner, hanya dapat dilakukan pada data yang sudah terurut. Sebagai contoh, jika ada sebuah larik yang berisi angka-angka sebagai berikut.

5	3	7	2	0	9	4	1	8	6

Jika yang dicari adalah angka 8 maka prosesnya sebagai berikut.

Langkah	Keterangan
1	<table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr></table> 3. Tabel masih dianggap sebagai satu kesatuan yakni table berisi data yang telah terurut.
2	<table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr></table> 4. Tabel dibagi menjadi dua. 5. Dicek nilai paling kanan subbagian kiri dan nilai paling kiri subbagian kanan. 6. Jika nilai yang dicari lebih besar atau sama dengan nilai paling kiri subbagian kanan, cari pada subbagian kanan. Jika nilai yang dicari lebih kecil daripada nilai paling kanan subbagian kiri maka cari pada subbagian kiri, karena yang dicari adalah 8 maka dicari pada subbagian kanan.
3	<table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr></table> 7. Subbagian kanan dibagi menjadi dua. 8. Dicek nilai paling kanan subbagian kiri dan nilai paling kiri subbagian kanan.

<table>
<tr><td></td><td>9. Jika nilai yang dicari lebih besar atau sama dengan nilai paling kiri subbagian kanan, cari pada subbagian kanan. Jika nilai yang dicari lebih kecil daripada nilai paling kanan subbagian kiri maka cari pada subbagian kiri, karena yang dicari adalah 8 maka dicari pada subbagian kanan.</td></tr>
<tr><td>4</td><td>
<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr>
</table>
10. Tabel dibagi menjadi dua.
11. Dicek nilai paling kanan subbagian kiri dan nilai paling kiri subbagian kanan.
12. Jika nilai yang dicari lebih besar atau sama dengan nilai paling kiri subbagian kanan, cari pada subbagian kanan. Jika nilai yang dicari lebih kecil daripada nilai paling kanan subbagian kiri maka cari pada subbagian kiri, karena yang dicari adalah 8 maka dicari pada subbagian kanan.
</td></tr>
</table>

Langkah-langkah di atas jika dituangkan di dalam C++ menjadi sebagai berikut.

```
#include<iostream>
using namespace std;
int main(){
    bool found=false;
    int data[]={0,1,2,3,4,5,6,7,8,9};
    int i=0,j=sizeof(data)/sizeof(data[0]),k,query=7;
    while(!found&&i<=j){
        k=(i+j)/2;
        if(data[k]<query){
            i=k+1;
        }else if(data[k]==query){
            found=true;
        }else{
            j=k-1;
        }
    }
    if(!found){
        cout<<"Tidak ditemukan.";
    }else{
        cout<<"Ditemukan.";
    }
    return 0;
}
```

Gambar 8. 9. Source Code Pencarian

Latihan

Diketahui data siswa sebagai berikut.

NISN	Nama	Nilai
9960312699	Handi Ramadhan	90
9963959682	Rio Alfandra	55
9950310962	Ronaldo Valentino Uneputty	80
9970272750	Achmad Yaumil Fadjri R.	60
9970293945	Alivia Rahma Pramesti	70
9952382180	Ari Lutfianto	65
9965653989	Arief Budiman	60

1. Urutkan data tersebut dengan urutan menaik berdasarkan:
   - NISN
   - Nilai

   dengan menggunakan metode pengurutan penyisipan, seleksi, dan gelembung.
2. Carilah data yang memiliki NISN 9950310962, kemudian tampilkan nilainya menggunakan pencarian bagi dua.
3. Ubah nama data yang memiliki nilai 60 menjadi Joko. Manfaatkan metode pencarian beruntun.

BAB IX

POINTER

7.1. Tujuan

1. Mahasiswa mampu memahami konsep pointer

2. mahasiswa mampu menerapkan pointer pada linked list

7.2. Teori

Pointer merupakan sebuah variable yang berisi alamat dari variable lain. Suatu pointer dimaksudkan untuk menunjukkan ke suatu alamat memori sehingga alamat dari suatu variable dapat diketahui dengan mudah. Konsep pointer sebenarnya cukup sederhana. Pointer sesungguhnya berisi alamat dari suatu data, bukan data sebagaimana variabel yang telah anda kenal.

Mengapa harus menggunakan **POINTER**?

> *Karena dengan menggunakan pointer dapat meningkatkan kinerja untuk operasi yang dilakukan secara berulang. Dengan syarat Kalau mendeklarasikan pointer kedalam array, tidak boleh menggunakan tanda bintang.*
>
> *Pointer juga di gunakan untuk mengalokasikan tempat pada memori secara dinamis yakni dapat diubah-ubah alokasi tempatnnya pada memori data yang dimasukkan sebagai nilai pointer akan selalu tersimpan sehingga diperlukan penghapusan yang tujuannya untuk mengosongkan memori, perintah yang digunakan untuk menghapus memori adalah delete [] nama variable.*

Pointer juga berguna untuk :

1. Mengirimkan "Parameter yang berupa variabel" ke dalam fungsi, artinya nilai variabel bisa diubah di dalam fungsi.

2. Untuk membuat variabel DINAMIS (Bukan variabel Statis)

Penggunaan Awal Pointer

Jika variabel merupakan isi memori, dan untuk mengakses isi memori tersebut diperlukan address, lalu bagaimana cara kita mengetahui alamat dari suatu variabel ? Jawabannya adalah : untuk kebanyakan kasus kita sama sekali tidak perlu tahu alamat dari sebuah variabel. Untuk mengakses sebuah variabel kita hanya perlu nama dari variabel tersebut. Tugas kompiler lah yang mentranslasikan nama ke alamat mesin yang diperlukan oleh komputer.

Akan tetapi terdapat beberapa kasus dimana kita tidak mungkin memberi nama pada sebuah entitas di program kita. Hal ini terjadi terutama saat kita menggunakan data struktur dinamis seperti linked list, resizeable array, tree dan lain sebagainya. Hal ini karena kita tidak

mungkin memberi nama terhadap entitas yang mungkin ada atau tidak ada. Struktur seperti linked list hampir mustahil dibuat tanpa pointer tanpa harus mendefinisikan LISP-like list.

1. Penggunaan Pointer Sebagai Moniker.

Istilah moniker di sini berarti sesuatu yang menunjuk atau mengacu kepada entitas lain. Penggunaan lain pointer sebagai moniker adalah untuk mengatasi kelemahan bahasa C awal : Dahulu fungsi - fungsi di C hanya mengerti pass by value. Pointer digunakan untuk mengemulasi pass by reference karena pointer berisi alamat ke objek lain, sehingga fungsi tersebut dapat mengubah objek tersebut dengan memanipulasi pointer.

2. Operasi pointer arithmatic lain juga didefinisikan untuk pointer.

Yang menarik adalah increment dan decrement. programmer dapat memeriksa semua elemen dalam array dengan cara menginkremen pointer dari pointer penunjuk elemen pertama. Tentu saja hal yang sama dapat dilakukan dengan indexing biasa, ar[idx], akan tetapi dengan operasi pointer bisa lebih efisien. Alasannya terletak pada bagaimana cara komputer membaca data di ar[idx]. Untuk mesin yang memiliki indexed addressing hal ini cukup sederhana dan efisien (ar jadi base, idx jadi index, fetching cukup 1 instruksi mov). Tetapi untuk mesin yang tidak memiliki indexed addressing, akan ada operasi ADD antara ar dan idx, lalu simpan hasilnya ke suatu tempat (register), lalu baru mov. Kadang – kadang register tersebut digunakan untuk operasi ADD sehingga terdapat beberapa mov untuk menyimpan state. Akan tetapi jika menggunakan pointer arithmatic, cukup meng-increase nilai yang sudah ada di register, lalu mov. Tentu saja instruksi di dalam loop juga mempengaruhi efisiensi ini, tetapi untuk mesin yang mendukung operasi increment langsung, iterasi lewat pointer biasanya lebih efisien.

3. Penggunaan Pointer Sebagai Iterator.

Nama iterator diambil dari STL, dan iterator di STL adalah abstraksi dari pointer. Yang menakjubkan adalah konsep iterator, yang digeneralisasi dari pointer, adalah konsep yang cukup powerful untuk merepresentasikan semua algoritma yang bekerja untuk linear container (linear container adalah semua container yang memiliki iterator yang menunjuk pada elemen pertama, memiliki iterator yang menunjuk pada elemen one-past-end, dan semua elemen dapat dicapai dengan melakukan operasi incremen dari iterator penunjuk elemen pertama sebanyak yang diperlukan. Contoh linear container adalah array, vector, linked – list, dan deque. contoh yang bukan linear container adalah graph dan forest.).

Ketiga fungsi pointer di atas memerlukan operasi yang berbeda- beda. Contohnya jika pointer berfungsi sebagai moniker, operasi yang sangat diperlukan adalah fungsi malloc, calloc, free, new, delete, operator ->, operator * dan operator &. sebagai moniker pointer tidak memerlukan konvertability ke integer dan operasi pointer arihmatic (walaupun ada trik mengakses field struct dari pointer dengan mengcast pointer to struct menjadi char*, tambahkan offsetnya, lalu baca dengan operator * dan di cast ke tipe field tersebut. trik ini sangat berbahaya dan sebaiknya tidak dipakai). Jika pointer berfungsi sebagai iterator, operasi pointer arithmatic adalah esensial. Tetapi operasi new dan delete sama sekali tidak di perlukan (kecuali untuk array of pointer). bottom line is: you do not do memory management via iterator.

Sifat konvertibilitas antara integer dan pointer hanya diperlukan jika pointer tersebut dipakai sebagai abstraksi fixed address. Dua fungsi lain tidak memerlukan sifat ini.

Operator Pointer

Terdapat dua macam operator pointer yang ada pada C++ yaitu operator deference (&) dan operator reference (*).

1. Operator deference (&)

Saat mendeklarasikan variabel, kita tidak dapat menentukan alamat memori dari variable tersebut. compiler secara otomatis akan menempatkan variable pada alamat yang kosong. Kita dapat mengetahui alamat dari variable yang telah dibuat dengan menambahkan identifier "&"(*ampersand sign*) di depan nama variable, maka alamat memori variable dapat diketahui. Identifier ini disebut operator deference atau operator alamat. Operator ini hanya dapat digunakan pada variable biasa.

2. Operator reference (*)

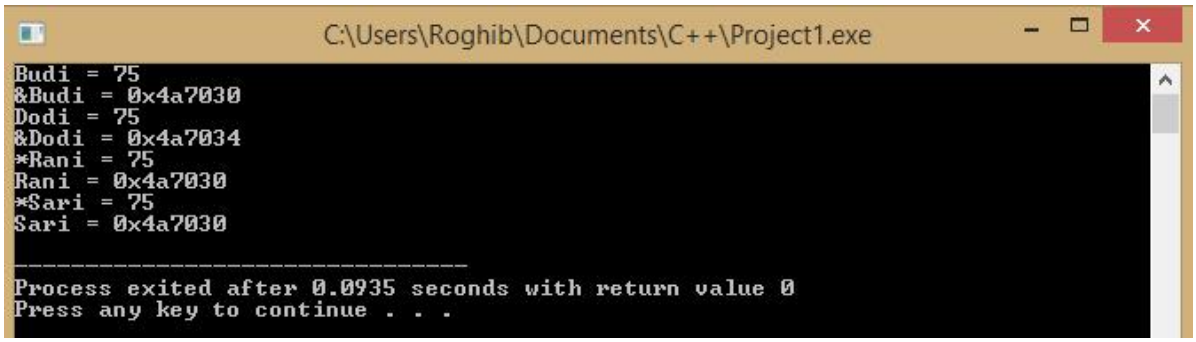
Pada variable pointer, identifier "*"(*asterisk*) perlu ditambahkan untuk dapat mengakses nilai variable pointer. Selain itu, tanda *asterisk* juga diperlukan pada deklarasi variable untuk membedakan variable pointer dengan variable biasa.

Sebagai contoh dapat dilihat pada listing program pada gambar 1. Variable biasa adalah Budi dan Dodi, sedangkan variable pointer adalah Rani dan Sari. Deklarasi variable pointer harus ditambahkan tanda *asterisk* di depannya seperti terlihat pada gambar 1 baris 6 dan 7.

```
2   #include<iostream>
3   using namespace std;
4   int Budi;
5   int Dodi;
6   int *Rani;
7   int *Sari;
8   int main() {
9
10      Budi = 75;  //nilai variabel Budi (75)
11      cout<<"Budi = "<<Budi<<endl;    //isi variabel Budi
12      cout<<"&Budi = "<<&Budi<<endl;  //alamat variabel Budi
13
14      Dodi = Budi ; // Nilai Dodi sama dengan Budi (75)
15      cout<<"Dodi = "<<Dodi<<endl;    //isi variabel Dodi
16      cout<<"&Dodi = "<<&Dodi<<endl;  //alamat variabel Dodi
17
18      Rani = &Budi; //Variabel pointer Rani sama dengan alamat variabel Budi
19      cout<<"*Rani = "<<*Rani<<endl;  //isi variabel Rani
20      cout<<"Rani = "<<Rani<<endl;    //alamat variabel Rani
21
22      Sari = Rani ; //alamat variabel pointer Sari sama dengan alamat pointer Rani
23      cout<<"*Sari = "<<*Sari<<endl;  //isi variabel Sari
24      cout<<"Sari = "<<Sari<<endl;    //alamat variabel Sari
25
26      return 0;
27  }
```

Gambar 1. Listing program operator pointer

Pada gambar 1 baris ke 12 digunakan operator deference untuk dapat melihat alamat dari variabel Budi. Kemudian pada baris ke 19 digunakan operator reference untuk melihat nilai dari variabel pointer Rani yang menunjuk ke alamat variabel Budi. Pada Gambar 2 dapat dilihat hasil output program.



Gambar 2. Output program pada gambar 1.

Mendeklarasikan Variabel Pointer

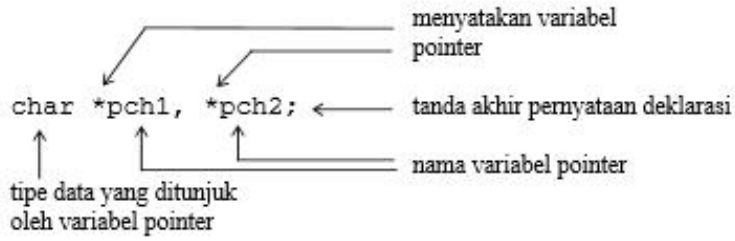
Suatu variabel pointer dideklarasikan dengan bentuk sebagai berikut :

```
tipe *nama_variabel
```

dengan tipe dapat berupa sembarang tipe data yang sudah dibahas pada bab-bab sebelumnya. Adapun nama_variabel adalah nama dari variabel pointer. Sebagai Contoh

```
int px;                  / *contoh 1 */
char *pch1, *pch2;       / *contoh 2 */
```

Contoh pertama menyatakan bahwa **px** adalah variabel pointer yang menunjuk ke suatu data bertipe *int*, sedangkan contoh kedua masing **pch1** dan **pch2** adalah variabel pointer yang menunjuk ke data bertipe *char*.



Gambar 3. Ilustrasi pendeklarasian variabel pointer

Mengatur Pointer agar Menunjuk ke Variabel Lain

Agar suatu pointer menunjuk ke variabel lain, mula-mula pointer harus diisi dengan alamat dari variabel yang akan ditunjuk. Untuk menyatakan alamat dari suatu variabel, operator **&** (operator alamat, bersifat unary) bisa dipergunakan, dengan menempatkannya di depan nama variabel. Sebagai contoh, bila **x**dideklarasikan sebagai variabel bertipe *int*, maka :

```
&x
```

berarti "alamat dari variabel **x**". Adapun contoh pemberian alamat **x** ke suatu variabel pointer **px** (yang dideklarasikan sebagai pointer yang menunjuk ke data bertipe *int*) yaitu :

```
px = &x;
```

Pernyataan di atas berarti bahwa **px** diberi nilai berupa alamat dari variabel **x**. Setelah pernyataan tersebut dieksekusi barulah dapat dikatakan bahwa **px** menunjuk ke variabel **x.**

Mengakses Isi Suatu Variabel Melalui Pointer
Jika suatu variabel sudah ditunjuk oleh pointer, variabel yang ditunjuk oleh pointer tersebut dapat diakses melalui variabel itu sendiri (pengaksesan langsung) ataupun melalui pointer (pengaksesan tak langsung). Pengaksesan tak langsung dilakukan dengan menggunakan operator indirection (tak langsung) berupa simbol *(bersifat unary).
Contoh penerapan operator * yaitu :

```
*px
```

yang menyatakan "isi atau nilai variabel/data yang ditunjuk oleh pointer **px**" . Sebagai contoh jika **y** bertipe *int*, maka sesudah dua pernyataan berikut:

```
px = &x;
y = *px;
```

y akan berisi nilai yang sama dengan nilai **x**.

Mengakses dan Mengubah isi Suatu Variabel Pointer
Contoh berikut memberikan gambaran tentang pengubahan isi suatu variabel secara tak langsung (yaitu melalui pointer). Mula-mula **pd**dideklarasikan sebagai pointer yang menunjuk ke suatu data bertipe *float* dan **d** sebagai variabel bertipe *float*. Selanjutnya :

```
d = 54.5;
```

digunakan untuk mengisikan nilai 54,5 secara langsung ke variabel **d**. Adapun :

```
pd = &d;
```

digunakan untuk memberikan alamat dari **d** ke **pd**. Dengan demikian **pd** menunjuk ke variabel **d**. Sedangkan pernyataan berikutnya :

```
*pd = *pd + 10; (atau:*pd += 10; )
```

merupakan instruksi untuk mengubah nilai variabel **d** secara tak langsung. Perintah di atas berarti "jumlahkan yang ditunjuk **pd** dengan 10 kemudian berikan ke yang ditunjuk oleh **pd**", atau identik dengan pernyataan :

```
d = d + 10;
```

Akan tetapi, seandainya tidak ada instruksi :

```
pd = &d;
```

maka pernyataan :

```
*pd = *pd + 10;
```

tidaklah sama dengan :

```
d = d + 10;
```

Pointer dan Array

Hubungan antara pointer dan array pada bahasa C sangatlah erat. Sebab sesungguhnya array secara internal akan diterjemahkan dalam bentuk pointer. Pembahasan berikut akan memberikan gambaran hubungan antara pointer dan array. Misalnya dideklarasikan di dalam suatu fungsi :

```
int tgl_lahir[3] = { 01, 09, 64 };
```

dan

```
int *ptgl;
```

Kemudian diberikan instruksi :

```
ptgl = &tgl_lahir[0];
```

maka **ptgl** akan berisi alamat dari elemen array **tgl_lahir** yang berindeks nol. Instruksi di atas bisa juga ditulis menjadi :

```
ptgl = tgl_lahir;
```

sebab namaarray tanpa tanda kurung menyatakan alamat awal dari array. Sesudah penugasan seperti di atas,

```
*ptgl
```

dengan sendirinya menyatakan elemen pertama (berindeks samadengan nol) dari array **tgl_lahir.**

Praktikum

Program 1

```
#include <iostream>
#include <stdio.h>
using namespace std;

int main() {
    int x = 3, y = 4;
    int *ip;
    ip = &x;
    y = *ip;
    x = 10;
    *ip = 3;
    printf("x = %d, y = %d", x, y);
    return 0;
}
```

Program 2

```
#include <iostream>
#include <stdio.h>
using namespace std;

int main() {
    int  count = 16, sum = 17, *x, *y;
    x = &sum;
    *x = 27;
    y = x;
    x = &count;
    *x = count;
    sum = *x / 2 * 3;
    printf("count = %d, sum = %d, *x = %d, *y = %d\n",  count,sum,*x,*y);
}
```

Program 3

```
#include <iostream>
#include <stdio.h>
using namespace std;

int r, q = 8;
int spesial(int *, int *);
main()
{
    int *ptr1 = &q;
    int *ptr2 = &q;
    r = go_crazy(ptr2, ptr1);
    printf("q = %d, r = %d, *ptr1 = %d, *ptr2 = %d\n",  q, r,*ptr1,*ptr2);
    ptr1 = &r;
    spesial(ptr1, ptr2);
    printf("q = %d, r = %d, *ptr1 = %d, *ptr2 = %d\n",  q, r,*ptr1,*ptr2);
}

int spesial(int *p1, int *p2)
{
    int x = 5;
    r = 12;
    *p2 = *p1 * 2;
    p1 = &x;
    return *p1 * 3;
}
```

Program 4

```
#include <iostream>
#include <stdio.h>
using namespace std;

main()
{
    int var_x = 273;
    int *ptr1;
    int **ptr2;
    ptr1 = &var_x;
    ptr2 = &ptr1;
    printf("Nilai var_x = *ptr1  = %d\n", *ptr1);
    printf("Nilai var_x = **ptr2 = %d\n\n", **ptr2);
    printf("ptr1 = &var_x = %p\n", ptr1);
    printf("ptr2 = &ptr1  = %p\n", ptr2);
    printf("       &ptr2  = %p\n", &ptr2);
}
```

TUGAS

1. Buat program untuk menampilkan tulisan: "**Praktikum Pemrograman**" menggunakan variabel pointer.
2. Buat Program untuk dapat mencetak huruf keempat dari kata: "**Gadjah Mada**" dengan Variabel Pointer.
3. Buat program dengan fungsi dan variabel pointer untuk dapat menukar dua isi variabel biasa.

DAFTAR PUSTAKA

Binanto, Iwan, 2003, Pemrograman C++ di Linux, Penerbit Andi, Jogjakarta.

Binanto, Iwan, 2004, Lebih Lanjut dengan Pemrograman C++ di Linux, Penerbit Andi, Jogjakarta.

Kadir, Abdul, 2004, Pemrograman Visual C++, Penerbit Andi, Jogjakarta.

Shalahuddin, M. dan A. S., Rosa, 2009, *Belajar Pemrograman dengan Bahasa C++ dan Java*, Penerbit Informatika, Bandung.

Java 2 TM Fourth Edition, McGraw Hill, New York

www.juicy.com/tutorial/c++

http://www.gamedevid.org/forum/showthread.php?t=5075

http://didicross.wordpress.com/2009/05/30/pointer-pada-c/

http://en.wikipedia.org/wiki/Pointer_(computing)

http://komunitas.coder.web.id/showthread.php?p=4280

Dasar Pemrograman I ITS

https://www.ntu.edu.sg/home/ehchua/programming/#Cpp