

MODUL I

PENGETAHUAN DASAR PEMROGRAMAN C++

A. TUJUAN PRAKTIKUM

1. Mahasiswa mengetahui dasar – dasar penulisan program komputer.
2. Mahasiswa memahami struktur bahasa C/C++.
3. Mahasiswa dapat mengetahui dan bisa menggunakan tipe data tertentu dalam pembuatan program.
4. Mahasiswa bisa memasukkan dan mengambil data dari program.

B. DASAR TEORI

1. Header File

Header file merupakan file yang berisi prototipe dari sekumpulan fungsi fungsi pustaka tersebut. Jadi, *header file* terdiri dari prototipe dari fungsi-fungsi pustaka, sedangkan fungsi-fungsi pustaka disimpan di file pustaka^[2].

Ada 2 cara untuk melibatkan file judul dalam suatu program yaitu menggunakan `<, >` dan menggunakan `" "`. Jika menggunakan `" "` maka kompiler akan mencari file dalam *default directory* dan kemudian ke *directory* file-file pustaka. Sedangkan jika menggunakan `<>` maka kompiler akan mencari di *directory* file-file pustaka saja.^[2]

2. Preprocessor Directive `#include`

Kalimat yang diawali dengan tanda `#` adalah *pre-processor directive*. Bukan merupakan baris kode yang dieksekusi, tetapi indikasi untuk kompiler. File spesifik ini juga termasuk pustaka deklarasi standar I/O pada C++ dan file ini disertakan karena fungsi-fungsinya akan digunakan nanti dalam program.^[3]

3. Fungsi `main ()`

Baris ini mencocokkan pada awal dari deklarasi fungsi utama. Fungsi utama merupakan titik awal dimana seluruh program C++ akan mulai dieksekusi. `main ()` Diletakkan diawal, ditengah atau diakhir program, isi dari fungsi `main` akan selalu dieksekusi pertamakali. Pada dasarnya, seluruh program C++ memiliki fungsi utama.^[3]

`main ()` diikuti oleh sepasang tanda kurung `()` karena merupakan fungsi. pada C++, semua fungsi diikuti oleh sepasang tanda kurung `()` dimana, dapat berisi argumen didalamnya. Isi dari fungsi `main` selanjutnya akan mengikuti, berupa deklarasi formal dan dituliskan diantara kurung kurawal `{ }`.^[3]

4. *Statement*

Statement merupakan bagian perintah yang berupa perintah yang akan dijalankan. Pada bahasa pascal, Perintah ini selalu diawali dengan *begin* dan *end*. Apabila *block statement* merupakan block utama, maka *end* diakhiri dengan titik. Sedangkan blok *statement* yang bukan merupakan blok utama, maka *end* diakhiri dengan tanda `“;”`. Sedangkan pada bahasa C/C++, dimulai deklarasi *variable* hingga akhir *statement* diawali dan diakhiri dengan kurung kurawal `{ }`.^[3]

5. *Semicolon*

Semicolon adalah tanda untuk mengakhiri pernyataan pada program C++. Setiap pernyataan harus diakhiri dengan *semicolon*. Terkecuali jika bagian depannya diberi tanda `#` karena itu merupakan *preprocessor directive*.^[2]

6. *Komentar*

komentar berfungsi untuk membantu ketika dalam proses utama suatu program. Komentar dibagi dua cara yaitu menggunakan `//` dan menggunakan `/* */`. Komentar dengan `//` akan dihitung sebagai baris baru. Sedangkan jika menggunakan `/* */` hanya bisa digunakan untuk membuat blok baru.^[2]

7. *Tipe Data*

Tipe data merupakan tempat untuk menentukan pemberian nilai terhadap suatu variabel yang diberikan oleh pengguna. Tipe data juga bisa diartikan sebagai batasan sebuah fungsi tanda pengenalan terhadap semua nilai yang diterima. Sebagai contoh jika menempatkan tanda pengenalan hanya mengenal angka, maka ketika diberi nilai berupa *string* maka data itu akan ditolak karena data tersebut tidak dikenal.^[2]

Tipe data dalam variabel menentukan data yang disimpan di dalamnya, format data yang disimpan, dan berapa banyak memori yang dialokasikan

untuk menyimpan data. Tipe data merupakan bagian program terpenting karena tipe data mempengaruhi setiap struktur yang dilaksanakan *computer*. Fungsinya untuk merepresentasikan nilai dari suatu variabel maupun konstanta, penyimpanan data di memori, dan menentukan nilai yang dapat diisikan ke dalam sebuah variable.^[2]

a. Tipe Data Dasar

Dalam program C++, tipe data dasar terdiri dari 5 bagian yaitu tipe data *integer* (nilai numerik bulat yang dideklarasikan *int*), *floating point* (nilai numerik pecahan ketetapan tunggal yang dideklarasikan dengan *float*), *double-precision* (nilai numerik pecahan ketetapan ganda yang dideklarasikan dengan *double*), karakter (yang dideklarasikan dengan *char*), dan kosong (dideklarasikan dengan *void*).^[2]

Berikut ini merupakan tabelnya:

Tabel 1.1 Tipe data dasar.

Tipe	Lebar	Range
Int	16 bit	-32768 sampai 32767
signed int	16 bit	-32768 sampai 32767
short int	16 bit	-32768 sampai 32767
signed short int	16 bit	-32768 sampai 32767
unsigned int	16 bit	0 sampai 65535
Unsigned short long int	16 bit	0 sampai 65535
Long int	32 bit	-2147483648 sampai 2147483649
signed long int	32 bit	-2147483648 sampai 2147483649
Usignet long int	32 bit	0 sampai 4294967296
Float	32 bit	3.4E-38 sampai 3.4E+38(7 digit)
double	64 bit	1.7E-308 sampai 1.7E+308(15 digit)
Long double	80 bit	3.4E-4932 sampai 1.1E+4932(19 digit)
Char	8 bit	-128 sampai 127

signed char	8 bit	-128 sampai 127
unsigned char	8 bit	0 sampai 225

Objek bertipe `void` tidak dapat didefinisikan, karena ukurannya belum diketahui. Tipe ini digunakan untuk `return value` fungsi atau untuk mendeklarasikan pointer yang objeknya belum diketahui.^[1]

Jenis-jenis tipe data dasar sebagai berikut:

1) Tipe data bilangan bulat

Tipe data ini digunakan untuk angka-angka yang tidak memiliki angka yang ada di belakang koma atau menyatakan bilangan bulat. Perubahan tanda bilangan pada bilangan bulat dapat diset dengan 2 tipe, yaitu: bilangan bulat bertanda (*signed integer*) yang memiliki *rangenegative* dan *positif* dan bilangan bulat tidak bertanda (*unsigned integer*) yang memiliki *range* bilangan positif saja.^[2]

2) Tipe data bilangan *real* atau pecahan

Tipe ini merepresentasikan bilangan desimal, pecahan, maupun eksponensial. Tipe data yang termasuk ke dalam kategori ini adalah `float` dan `double`.^[2]

3) Tipe data logika

Tipe data logika digunakan untuk merepresentasikan data-data yang mengandung nilai data *Boolean* yaitu 0 dan 1 atau yang sering disebut juga *true* dan *false*.^[2]

4) Tipe data karakter

Karakter adalah sembarang huruf, angka, atau tanda baca tunggal. Tipe data karakter merupakan kumpulan bermacam-macam karakter yang terdiri dari alfabet. Dimana karakter antara lain: alfabet desimal (0,1,2,3,4,5,6,7,8,9), alfabet huruf latin besar (A,B,C,D,E,...), alfabet huruf latin kecil (a,b,c,d,e,...) dan tanda baca tunggal (!,@,#,\$,...)^[2].

b. Tipe Data Bentuk

Tipe bentuk adalah tipe yang didefinisikan sendiri oleh pemrogram. Ada dua macam tipe bentuk, yaitu :

1. Tipe dasar yang diberi nama baru

Contoh :

Tipe `def int` bilangan

2. Rekaman (*struct*)

Nama diberikan kepada peubah (variabel), konstanta, tipe bentukan, nama fungsi, dan nama prosedur^[3].

Aturan penamaan :

- a. Diawali dengan huruf alfabet.
- b. Huruf besar atau kecil dibedakan.
- c. Nama tidak boleh mengandung operator aritmatika, operator relasional, tanda baca, spasi.^[3]

Berikut merupakan jenis-jenis tanda baca bentukan:

- 1) Tipe data enumerasi

Enumerasi adalah serangkaian simbol berurutan yang menspesifikasi konstanta bertipe *integer*. Dalam bahasa C tidak terdapat tipe *boolean*, sehingga untuk merepresentasikan *true* dengan nilai *integer* bukan 0 (1, 2, dst), sedangkan *false* dengan nilai 0^[2].

- 2) Tipe data *array*

Array adalah sekelompok data yang bertipe sama yang menduduki lokasi memori yang berurutan. Jumlah elemen *array* dinyatakan dengan menggunakan tanda `[]`.^[2]

- 3) Tipe data *string*

String adalah deretan karakter yang diakhiri dengan sebuah karakter kosong. Konstanta bertipe ditulis diantara tanda “ ”. dalam bahasa C *string* merupakan larik atau *array* dari tipe data *char*.^[2]

- 4) Tipe data struktur

Tipe data ini digunakan untuk mendeklarasikan sekelompok data yang memiliki tipe yang berlainan. *struct*: elemennya berada dilokasi memori yang berbeda, dan *union*: elemennya ada dilokasi memori yang sama.^[2]

8. Variabel

Variabel adalah suatu lambang dari sebuah lokasi yang berada di memori utama komputer yang berisi suatu nilai. Nilai yang berada di lokasi memori bisa berubah selama program dieksekusi. Fungsinya adalah sebagai tempat sementara dalam menyimpan data yang sedang diolah. Pemberian nilai pada variabel mempunyai bentuk penulisan yang berbeda-beda pada setiap bahasa pemrograman meskipun memiliki arti yang sama.^[1]

Variabel dibedakan menjadi variabel *numerik* dan variabel *string*. Variabel *numerik* mengandung nilai numerik atau angka. Sedangkan variabel *string* berisi huruf.^[1]

9. Operator

Bahasa C++ menyediakan beberapa operator untuk memanipulasi data. Secara umum, terdapat tiga jenis operator: *unary*, *binary* dan *ternary*. Istilah tersebut mencerminkan jumlah operand yang dibutuhkan. Operator *unary* hanya memerlukan satu operand. Misalnya, mempertimbangkan ekspresi berikut: `-5`. Dalam contoh diatas perlu dipahami bahwa mewakili angka lima bernilai negatif. Konstanta 5 adalah diawali dengan tanda *minus*. Tanda *minus*, bila digunakan dengan cara seperti ini, yang disebut penyangkalan operator. Karenanya memerlukan satu operand, hal tersebut merupakan operator *unary*. Operator *binary* bekerja dengan dua operand. Tugas operator ini biasanya pada operasi *aritmetik* yang hal tersebut sudah sangat umum dalam bahasa pemrograman.^[2] Berikut merupakan jenis-jenis operator:

a. Operator penugasan

Operator penugasan dalam bahasa C berupa tanda `=`. contoh:

```
nilai = 80
```

artinya variabel "nilai" diisi dengan variabel "A".^[2]

b. Operator aritmatika

Bahasa C menyediakan 5 operator aritmatika, yaitu^[2]:

Tabel 1.2 Operator aritmatika.

*	untuk perkalian
/	Pembagian
%	untuk sisa pembagian

+	untuk pengurangan
-	untuk penjumlahan

c. Operator hubungan

Operator hubungan digunakan untuk membandingkan hubungan antara 2 nilai atau variabel yang akan menghasilkan nilai *true* atau *false*. Dalam operator hubungan, tanda < memiliki arti lebih kecil, tanda > memiliki arti lebih besar, tanda <= memiliki arti lebih kecil sama dengan, tanda >= memiliki arti lebih besar sama dengan, tanda == memiliki arti sama dengan, dan tanda != memiliki arti tidak sama dengan.^[2]

d. Operator logika

Operator logika digunakan untuk membandingkan logika hasil dari operator-operator hubungan. Operator logika ada 3 macam, yaitu: && (AND), || (OR), dan ! (NOT). Operator ! (NOT) hanya memiliki satu operand yang berada di kananya.^[2]

e. Operator *bitwise*

Operator ini digunakan untuk memanipulasi bit-bit dari nilai data yang ada di memori.^[2] Operator *bitwise* dalam bahasa C yaitu:

Tabel 1.3 Operator *bitwise*.

<<	pergeseran bit ke kiri
>>	pergeseran bit ke kanan
&	bitwise AND
^	bitwise XOR
	bitwise OR
~	bitwise NOT

f. Operator kombinasi

Operator ini digunakan untuk memendekkan penulisan operasi penugasan.^[2]

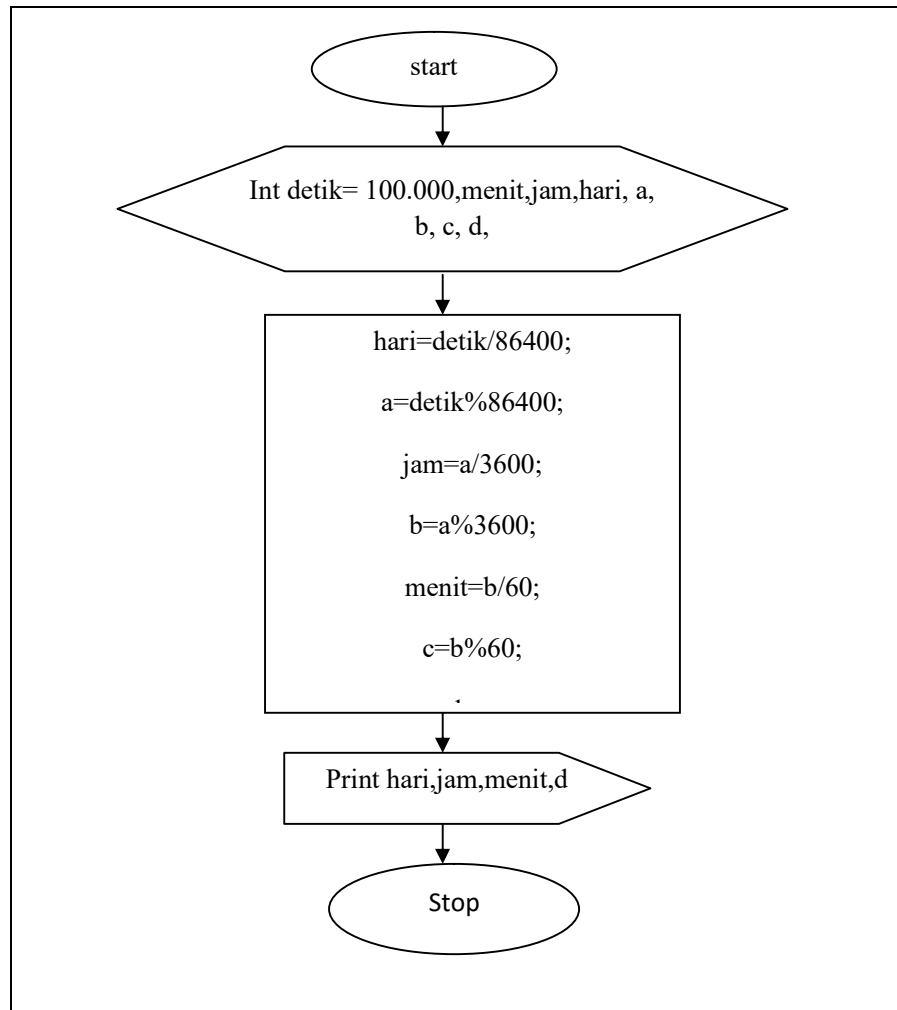
x=x+5;
bisa ditulis menjadi
x+=5;

C. PERMASALAHAN

1. Membuat program untuk mengetahui hari, jam, menit, detik.
(dengan *input* detik statis 100.000)
2. Membuat program untuk meng-input nilai a,b,c,d,e. kemudian menukar nilai pada *variable* diatas dengan ketentuan $a=e$, $b=a$, $c=b$, $d=c$, $e=d$.
3. Pak Rudi baru saja membangun kolam renang dengan lebar 10m, panjang 100m dengan kedalaman 2m. Kolam tersebut masih kosong dan ia ingin mengisi kolam tersebut dengan 10 kran air dari pipa PDAM yang memiliki debit masing-masing 10 liter perdetik.
Buatlah program yang dapat menghitung waktu yang diperlukan oleh pak Rudi untuk mengisi kolam tersebut sampai penuh dalam hitungan jam.
($\text{debit} = \text{volume} / \text{waktu}$).

D. HASIL PERCOBAAN

1. Permasalahan 1
 - a. Algoritma
 1. Menginisialisasi variabel detik= 100000, menit, jam, hari, a, b, c, d.
Variabel menit mewakili nilai menit, variabel jam mewakili nilai jam, variabel detik mewakili nilai detik, variabel a mewakili nilai sisa hari, variabel b mewakili nilai jam, variabel b mewakili nilai jam, variabel c mewakili nilai menit, variabel d mewakili nilai sisa detik.
 2. Menghitung hari, $\text{hari} = \text{detik} : 86400$.
 3. sisa hari, $a = \text{detik} \% 86400$.
 4. Menghitung jam, $\text{jam} = a : 3600$.
 5. Menghitung sisa jam, $b = a \% 3600$.
 6. Menghitung menit, $\text{menit} = b : 60$.
 7. Menghitung sisa menit, $c = b \% 60$.
 8. Menghitung sisa detik, $d = c$.
 9. Menampilkan "hari=" yang diambil dari hari.
 10. Menampilkan "jam=" yang diambil dari jam.
 11. Menampilkan "menit=" yang diambil dari menit.
 12. Menampilkan "menit=" yang diambil dari menit.
 13. Menampilkan "detik=" yang diambil dari d.

b. *FlowChart***Gambar 1.1** Flowchart program menghitung waktu.c. *Pseudocode*

Step1: detik=100.000
 Step2: hari \rightarrow detik/86400
 Step3: a \rightarrow detik%86400
 Step4: jam \rightarrow a/3600
 Step5: b \rightarrow a%3600
 Step6: menit \rightarrow b/60
 Step7: c \rightarrow b%60
 Step8: d \rightarrow c
 Step9: print hari, jam, menit, d

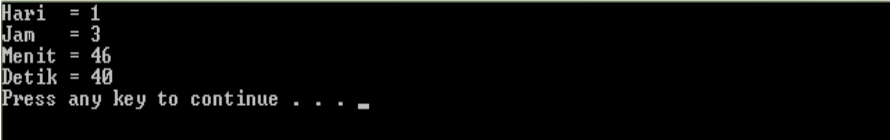
d. *Sourcecode*

```

#include <iostream.h>
#include <conio.h>
#include <stdlib.h>
int main()
{
    int detik,menit,jam,hari,a,b,c,d;
    detik=100000;
    hari=detik/86400;
    a=detik%86400;
    jam=a/3600;
    b=a%3600;
    menit=b/60;
    c=b%60;
    d=c;
    cout<<"Hari   = "<<hari<<endl;
    cout<<"Jam    = "<<jam<<endl;
    cout<<"Menit  = "<<menit<<endl;
    cout<<"Detik  = "<<d<<endl;
    system("pause");
}

```

e. Hasil Program



```

Hari   = 1
Jam    = 3
Menit  = 46
Detik  = 40
Press any key to continue . . .

```

Gambar 1.2 Hasil program konversi waktu

Program dimulai dengan mendeklarasikan variabel detik = 100.000, menit, jam, hari, a, b, c, d dengan menggunakan tipe data integer. Kemudian menentukan hari dengan melakukan proses detik dibagi 86400, kemudian modulus detik dengan 86400 dimasukkan di variabel a. Setelah itu, menentukan jam dengan membagi a dengan 3600, kemudian sisa pembagian jam dimasukkan ke variabel b, menentukan menit dengan membagi b dengan 60, sisa pembagian menit dimasukkan ke variabel c, kemudian, nilai c dimasukkan ke dalam variabel d. Setelah itu keluarkan hari, jam, menit, dan d.

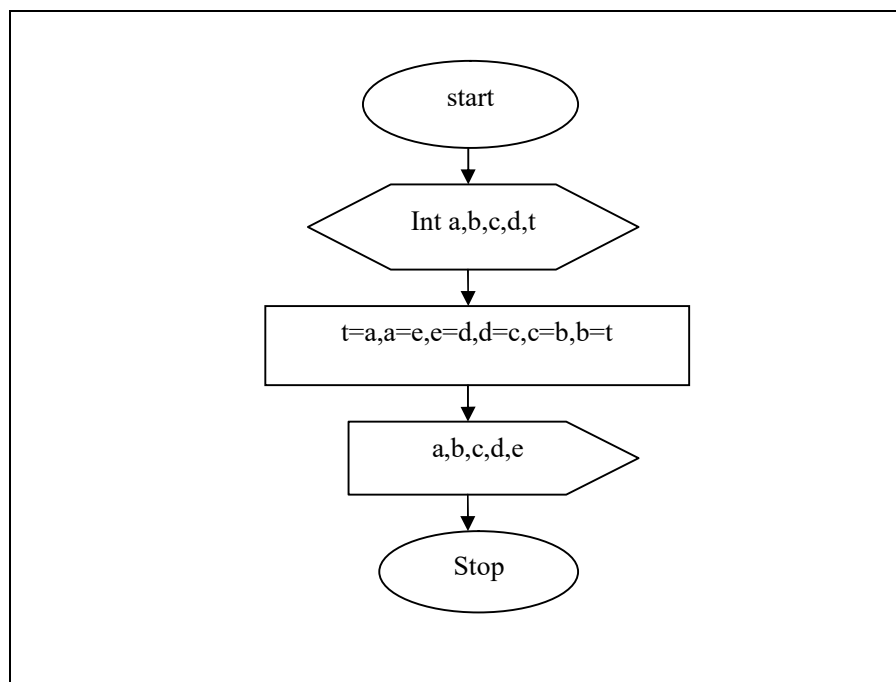
2. Permasalahan 2

a. Algoritma

1. Mendeklarasikan variabel a, b, c, d, e dan deklarasikan variabel t sebagai variabel tambahan untuk menampung nilai a

2. Memasukkan nilai pada masing-masing variabel dengan cara statis kecuali variabel tambahan
3. Memindahkan nilai pada setiap masing-masing variabel dengan ketentuan $a=e$, $b=a$, $c=b$, $d=c$, $e=d$
4. Memindahkan nilai pada variabel a ke variabel tambahan, memasukkan nilai variabel a dengan nilai variabel e , memasukkan nilai variabel e dengan nilai variabel d , memasukkan nilai variabel d dengan nilai variabel c , memasukkan nilai variabel c dengan nilai variabel b , memasukkan nilai variabel b dengan nilai variabel t .
5. Tampilkan hasil penukaran nilai dari masing-masing variabel a, b, c, d, e .

b. *Flowchart* permasalahan kedua



Gambar 1.3 Flowchart program tukar nilai.

c. *Pseudocode* permasalahan kedua

Step1: input a, b, c, d, e

Step2: $t \rightarrow a$

Step3: $a \rightarrow e$

Step4: $e \rightarrow d$

Step5: $d \rightarrow c$

Step6: $c \rightarrow b$

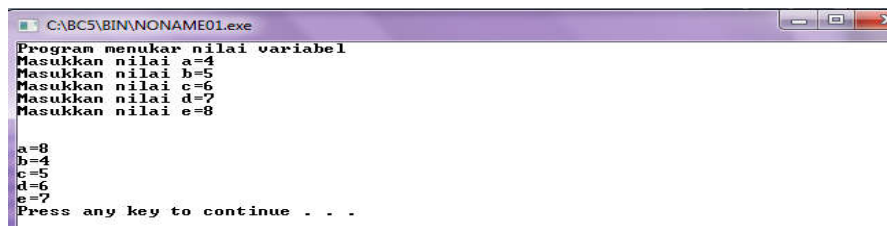
Step7: $b \rightarrow t$

Step8: *print a,b,c,d,e*

d. *Source code* permasalahan kedua

```
#include <iostream.h>
#include <stdlib.h>
void main ()
{
    int a,b,c,d,e,t;
    cout<<"Program menukar nilai variabel"<<endl;
    cout<<"Masukkan nilai a="<<cin>>a;
    cout<<"Masukkan nilai b="<<cin>>b;
    cout<<"Masukkan nilai c="<<cin>>c;
    cout<<"Masukkan nilai d="<<cin>>d;
    cout<<"Masukkan nilai e="<<cin>>e;
    cout<<endl<<endl;
    t=a;
    a=e;
    e=d;
    d=c;
    c=b;
    b=t;
    cout<<"a="<<a<<endl;
    cout<<"b="<<b<<endl;
    cout<<"c="<<c<<endl;
    cout<<"d="<<d<<endl;
    cout<<"e="<<e<<endl;
    system("pause");
}
```

e. Hasil program



```
C:\BC5\BIN\NONAME01.exe
Program menukar nilai variabel
Masukkan nilai a=4
Masukkan nilai b=5
Masukkan nilai c=6
Masukkan nilai d=7
Masukkan nilai e=8

a=8
b=4
c=5
d=6
e=7
Press any key to continue . . .
```

Gambar 1.4 Hasil program tukar nilai.

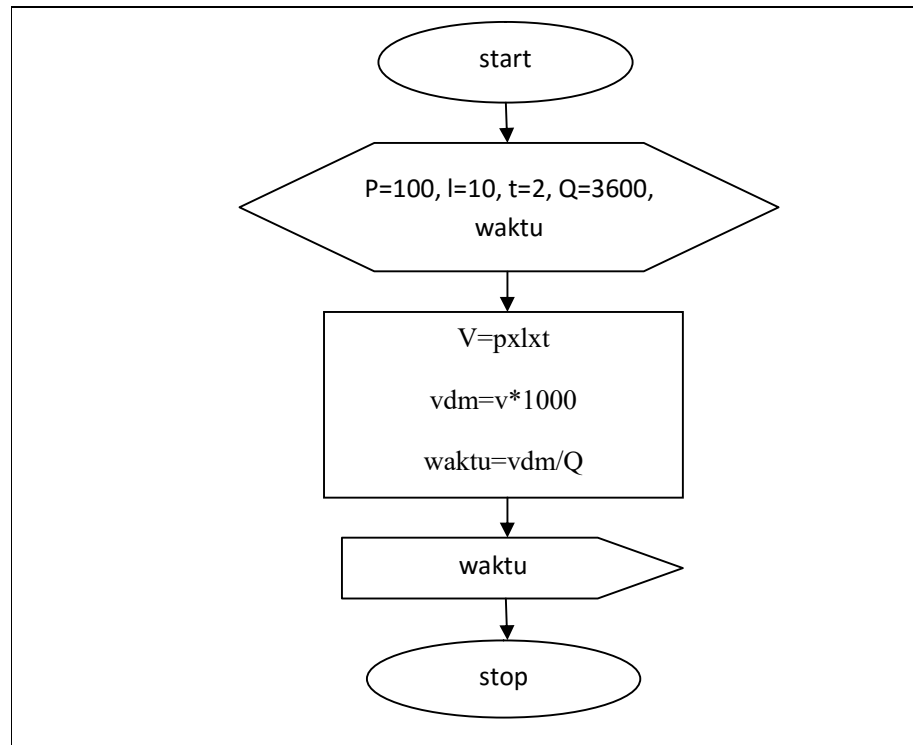
Program dimulai dengan mendeklarasikan variabel a,b,c,d,e, dan t. Kemudian meminta masukan untuk variabel a,b,c,d, dan e. Setelah itu, t digunakan untuk menampung nilai a, karena nilai a telah kosong, nilai e dimasukkan ke nilai a, kemudian nilai d dimasukkan ke nilai e, nilai c dimasukkan ke nilai d, nilai b dimasukkan ke nilai c nilai t dimasukkan

ke nilai b. Setelah itu keluarkan nilai a,b,c,d, dan e. Program tukar nilai pun selesai

3. Permasalahan 3

a. Algoritma

1. Menginisialisasi variabel p, l, t, v, vdm, Q, waktu. Variabel p mewakili nilai panjang, variabel l mewakili nilai lebar, variabel t mewakili nilai kedalaman, variabel v mewakili nilai volume, variabel vdm mewakili nilai volume dalam dm^3 , variabel Q mewakili nilai debit dalam liter/jam, variabel waktu mewakili lamanya waktu yang dibutuhkan untuk mengisi penuh bak.
2. Memasukkan nilai 100 pada variabel p, memasukkan nilai 10 pada variabel l, memasukkan nilai 2 pada variabel t, dan memasukkan nilai 360000 pada Q
3. Melakukan proses dengan rumus $v = p \times l \times t$
4. Melakukan proses dengan rumus $\text{vdm} = v \times 1000$
5. Melakukan proses $\text{waktu} = \text{vdm} / Q$
6. Melakukan keluaran “waktu yang dibutuhkan” yang diambil dari waktu

b. *Flowchart* program ketiga**Gambar 1.5** Flowchart program menentukan waktu.c. *Pseudocode*

Step1: $p \rightarrow 100, l \rightarrow 10, t \rightarrow 2, Q \rightarrow 3600$

Step2: $v \rightarrow p * l * t$

Step3: $vdm \rightarrow v * 1000$

Step4: $waktu \rightarrow vdm / Q$

Step5: *print waktu*

d. *Source Code*

```

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

void main() {
    int p=100, l=10, t=2, v, vdm;
    double Q=360000, waktu;

    v=p*l*t;
    vdm=v*1000;
    waktu=vdm/Q;

    cout<<"waktu yang dibutuhkan= "<<waktu;
    cout<<endl;
    system("pause");
}
  
```

e. Hasil Program



Gambar 1.6 Program menentukan waktu.

Program dimulai dengan variabel $p=100$, $l=10$, $t=2$, v , vdm dengan tipe data integer, dan variabel $Q=360000$ dan waktu dengan tipe data double. $Q=360000$ diperoleh dari konversi liter/detik menjadi liter/jam. Kemudian proses v dengan rumus volume balok yaitu $v=p \cdot l \cdot t$, karena masih dalam bentuk m^3 maka dilakukan proses $vdm=v \cdot 1000$ sehingga menjadi dm^3 . Kemudian menentukan waktu dengan membagi waktu dengan Q yang telah ditetapkan sebelumnya. Kemudian keluarkan waktu, dan program pun selesai.

E. ANALISA DATA

1. Analisis Permasalahan 1

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

- “`iostream.h`” merupakan *header file* yang mendeklarasikan *statement* “`cout`” dan “`cin`” yakni *statement* yang bertugas sebagai *statement inputoutput (I/O)* pada program.

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

- “`stdlib.h`” merupakan *header file* yang mendeklarasikan *statement* “`system("pause")`” yakni suatu *statement* perintah menunda proses *compiler* program.

```
void main () {
```

- “`main ()`” merupakan fungsi yang akan pertama kali di panggil pada saat program dijalankan (*compile*).
- “`void`” adalah tipe data yang brarti data atau nilai di dalam program tidak memiliki nilai balik.
- “`{`” merupakan penunjuk awal dari fungsi “`main`”.

```
int detik=100000,menit,jam,hari,a,b,c,d;
```

- “`int`” merupakan tipe data dasar yang berupa bilangan bulat.
- “`detik=100000`” adalah variabel yang diberi nilai 100000.
- “`menit,jam,hari,a,b,c,d;`” merupakan variabel.
- Tanda “`;`” adalah *semicolon* yang berfungsi untuk mengakhiri *statement*.

```
hari=detik/86400;
```

- *Script* di atas adalah rumus untuk menentukan hari.
- “`/`” merupakan salah satu bentuk operasi aritmatika yaitu pembagian.

```
a=detik%86400;
```

- “`a`” digunakan sebagai penampung nilai dari hasil “`detik%86400`”.
- “`%`” merupakan salah satu bentuk operasi aritmatika yang merupakan sisa bagi.

```
jam=a/3600;
```

- “`jam`” digunakan sebagai penampung nilai dari hasil “`a/3600`”.

```
b=a%3600;
```

- “`b`” digunakan sebagai penampung nilai dari hasil “`a%3600`”.

```
menit=b/60;
```

- “`menit`” digunakan sebagai penampung nilai dari hasil “`b/60`”.

```
c=b%60;
```


- “b” digunakan sebagai penampung nilai dari hasil “a%3600”.

```
d=c;
```

- “d” digunakan sebagai penampung nilai dari “c”.

```
cout<<" Hari  = "<<hari<<endl;
```

- *Script* di atas akan menampilkan *statement* “Hari = ”, dimana nilainya di simpan dalam *variable* “hari” yang di peroleh dari rumus “hari=detik/86400”.

- “endl” digunakan untuk membuat baris baru pada program.

```
cout<<"Jam  = "<<jam<<endl
```

- *Script* di atas akan menampilkan *statement* “Jam = ”, dimana nilainya di simpan dalam *variable* “jam” yang di peroleh dari rumus “jam=a/3600”.

```
cout<<"menit  = "<<menit<<endl
```

- *Script* di atas akan menampilkan *statement* “menit = ”, dimana nilainya di simpan dalam *variable* “menit” yang di peroleh dari rumus “menit=b/60”.

```
cout<<"detik  = "<<d<<endl
```

- *Script* di atas akan menampilkan *statement* “detik = ”, dimana nilainya di simpan dalam *variable* “d”.

```
system("pause");
}
```

- “system (“pause”)” merupakan fungsi penunda dimana proses akan terhenti sampai *statement* ini “system (“pause”)”, sehingga pada saat program di jalankan program tidak langsung tertutup.
- “}” berfungsi untuk mengakhiri fungsi “main()”.

2. Analisis Permasalahan 2

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

- “iostream.h” merupakan *headerfile* yang mendeklarasikan *statement* “cout” dan “cin” yakni *statement* yang bertugas sebagai *statement inputoutput* (I/O) pada program.

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

- “stdlib.h” merupakan *headerfile* yang mendeklarasikan *statement* “system(“pause”)” yakni suatu *statement* perintah menunda proses *compiler* program.

```
void main () {
```

- “main ()” merupakan fungsi yang akan pertama kali di panggil pada saat program dijalankan (*compile*).
- “void” adalah tipe data yang brarti data atau nilai di dalam program tidak memiliki nilai balik.
- “{” merupakan penunjuk awal dari fungsi main.

```
int a,b,c,d,e,t;
```

- “int” merupakan tipe data dasar yang berupa bilangan bulat.
- “a,b,c,d,e,t” merupakan variabel.
- Tanda “;” adalah *semicolon* yang berfungsi untuk mengakhiri *statement*.

```
cout<<"Program menukar nilai variabel"<<endl;
```

- *Script* di atas akan menampilkan *statement* “Program menukar nilai variabel”
- “endl” digunakan untuk membuat baris baru pada program.
- “<<” berfungsi untuk menginput data yang mengikutinya.

```
cout<<"Masukkan nilai a=";<<cin>>a;
```

- *Script* di atas akan menampilkan *statement* “Masukkan nilai a=”, dimana nilainya akan disimpan dalam variabel “a”.
- “cin>>a” merupakan fungsi input untuk memasukkan nilai “a”.

```
cout<<"Masukkan nilai b=";<<cin>>b;
```

- *Script* di atas akan menampilkan *statement* “Masukkan nilai b=”, dimana nilainya akan disimpan dalam variabel “b”.
- “cin>>b” merupakan fungsi input untuk memasukkan nilai “b”.

```
cout<<"Masukkan nilai c=";<<cin>>c;
```

- *Script* di atas akan menampilkan *statement* “Masukkan nilai c=”, dimana nilainya akan disimpan dalam variabel “c”.
- “cin>>c” merupakan fungsi input untuk memasukkan nilai “c”.

```
cout<<"Masukkan nilai d=";<<cin>>d;
```

- *Script* di atas akan menampilkan *statement* “Masukkan nilai d=”, dimana nilainya akan disimpan dalam variabel “d”.
- “cin>>d” merupakan fungsi input untuk memasukkan nilai “d”.

```
cout<<"Masukkan nilai e=";<<cin>>e;
```

- *Script* di atas akan menampilkan *statement* “Masukkan nilai e=”, dimana nilainya akan disimpan dalam variabel “e”.

- “cin>>e” merupakan fungsi input untuk memasukkan nilai “e”.

```
cout<<endl<<endl;
```

- Script di atas berfungsi untuk memberikan baris baru sehingga terdapat jarak baris dengan *statement* sebelumnya.

```
t=a;
```

- “t” digunakan untuk menampung nilai “a”.

```
a=e;
```

- “a” digunakan untuk menampung nilai “e”.

```
e=d;
```

- “e” digunakan untuk menampung nilai “d”.

```
d=c;
```

- “d” digunakan untuk menampung nilai “c”.

```
c=b;
```

- “c” digunakan untuk menampung nilai “b”.

```
b=t;
```

- “b” digunakan untuk menampung nilai “t”.

```
cout<<"a="<<a<<endl;
```

- *Script* di atas akan menampilkan *statement* “a=”, dimana nilainya disimpan dalam variabel “a”. Sehingga nilai dari variabel “a” dapat ditampilkan.

```
cout<<"b="<<b<<endl;
```

- *Script* di atas akan menampilkan *statement* “b=”, dimana nilainya disimpan dalam variabel “b”. Sehingga nilai dari variabel “b” dapat ditampilkan.

```
cout<<"c="<<c<<endl;
```

- *Script* di atas akan menampilkan *statement* “c=”, dimana nilainya disimpan dalam variabel “c”. Sehingga nilai dari variabel “c” dapat ditampilkan.

```
cout<<"d="<<d<<endl;
```

- *Script* di atas akan menampilkan *statement* “d=”, dimana nilainya disimpan dalam variabel “d”. Sehingga nilai dari variabel “d” dapat ditampilkan.

```
cout<<"e="<<e<<endl;
```

- *Script* di atas akan menampilkan *statement* “e=”, dimana nilainya disimpan dalam variabel “e”. Sehingga nilai dari variabel “e” dapat ditampilkan.

```
system("pause");
}
```

- “system (“pause”)” merupakan fungsi penunda dimana proses akan terhenti sampai *statement* ini “system (“pause”)”, sehingga pada saat program di jalankan program tidak langsung tertutup.
- “}” berfungsi untuk mengakhiri fungsi “main()”.

3. Analisis Permasalah 3

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

- “iostream.h” merupakan *headerfile* yang mendeklarasikan *statement* “cout” dan “cin” yakni *statement* yang bertugas sebagai *statement inputoutput* (I/O) pada program.

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

- “stdlib.h” merupakan *headerfile* yang mendeklarasikan *statement* “system(“pause”)” yakni suatu *statement* perintah menunda proses *compiler* program.

```
void main () {
```

- “main ()” merupakan fungsi yang akan pertama kali di panggil pada saat program dijalankan (*compile*).
- “void” adalah tipe data yang brarti data atau nilai di dalam program tidak memiliki nilai balik.
- “{” merupakan penunjuk awal dari fungsi main.

```
int p=100,l=10,t=2,v,vdm;
```

- “int” merupakan tipe data dasar yang berupa bilangan bulat.
- “p=100” merupakan variabel yang diberi nilai 100.
- “l=10” merupakan variabel yang diberi nilai 10.
- “t=2” merupakan variabel yang diberi nilai 2.
- “v” dan “vdm” merupakan variabel.
- “;” adalah *semicolon* yang berfungsi untuk mengakhiri *statement*.

```
double Q,waktu;
```

- “double” merupakan tipe data dasar yang berupa bilangan riil.
- “Q” dan “waktu” merupakan variabel.

```
v=p*l*t;
```

- *Script* di atas adalah rumus untuk menentukan volume dari bak.

```
vdm=v*1000;
```

- *Script* di atas adalah rumus untuk mengubah satuan volume dari m^3 menjadi dm^3 .

```
waktu=vdm/Q;
```

- *Script* di atas adalah rumus untuk menentukan waktu sampai bak terisi penuh.

```
cout<<"waktu yang dibutuhkan= "<<waktu;
```

- *Script* di atas akan menampilkan *statement* “"waktu yang dibutuhkan= ””, dimana nilainya disimpan dalam variabel “waktu”, sehingga nilai dari variabel “waktu” dapat ditampilkan.

```
system("pause");  
}
```

- “system (“pause”)” merupakan fungsi penunda dimana proses akan terhenti sampai *statement* ini “system (“pause”)”, sehingga pada saat program di jalankan program tidak langsung tertutup.
- “}” berfungsi untuk mengakhiri fungsi “main()”.

F. KESIMPULAN

Kesimpulan dari laporan ini adalah:

1. Dasar program komputer terdiri dari algoritma, *flowchart*, *pseudocode*, dan *source code*. Dalam penulisan program komputer, struktur dan bahasa program harus benar sehingga tidak terjadi eror.
2. Bahasa C++ memiliki struktur dasar yaitu preprocessor directive yang diikuti header file, fungsi, dan statement.
3. Dalam pemrograman C++ terdapat dua tipe data yaitu tipe data dasar dan tipe data bentukan. Tipe data dasar terdiri dari `char`, `float`, `int`, `double`, `void`. Tipe data bentukan terdiri dari struktur enumerasi.
4. Pada program C++, proses memasukkan data (*input*) dapat dilakukan dengan menggunakan `cin`, sedangkan untuk pengeluaran data (*output*) menggunakan `cout`.

DAFTAR PUSTAKA

- [1] Davis,Stephen Rendy. 2004. *C++ for Dummies 5th Edition*. Amerika: Wiley publishing,inc.
- [2] Suryani,Des.2013..*Jurnal Informasi dan Pendidikan Volume 6*.Riau: Universitas Islam Riau.
- [3] Wisdasari,Dian.2010..*Membuat Program dengan Menggunakan Bahasa “C” Volume 8*.Jakarta.Penerbit:Guna Darma.