Modul Praktikum

Pemrograman Komputer

Diploma 4

Penyusun

Ir. Widya Cahyadi, S.T., M.T.

PROGRAM STUDI DIPLOMA 4

FAKULTAS TEKNIK

Universitas Jember

Daftar Isi

Praktikum Bab 1

ALGORITMA DAN FLOW CHART

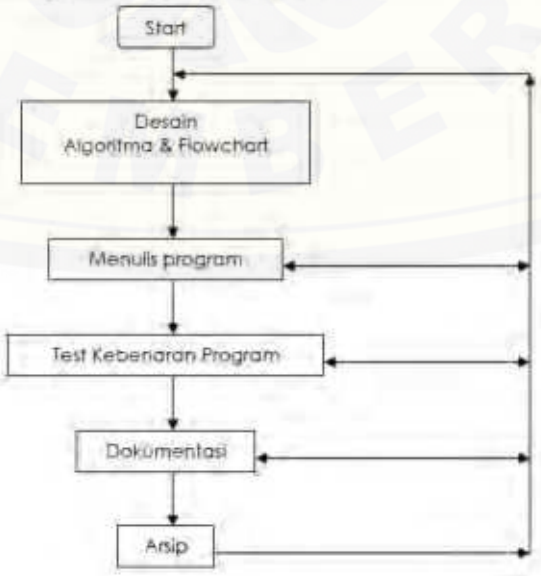
A. TUJUAN

- Mahasiswa mampu menyusun urutan langkah logis yang digunakan untuk menyelesaikan suatu masalah
- Mahasiswa mampu mendesain solusi permasalahan yang dituangkan ke dalam algoritma
- Mahasiswa mampu menotasikan algoritma yang telah disusun menggunakan notasi diagram alir (Flow chart)

B. DASAR TEORI

Beberapa tahapan proses membuat suatu program :

1. **Mendefinisikan masalah dan menganalisanya**
   - ✓ Mengidentifikasikan masalah antara lain tujuan dari pembuatan program, parameter-parameter yang digunakan, fasilitas apa saja yang akan disediakan oleh program.
   - ✓ Menentukan metode atau algoritma apa yang akan diterapkan untuk menyelesaikan masalah tersebut.
   - ✓ Menentukan bahasa program yang digunakan untuk pembuatan program.
2. **Merealisasikan** dengan langkah-langkah berikut :



Algoritma

Algoritma adalah urutan langkah-langkah logika yang menyatakan suatu tugas dalam menyelesaikan suatu masalah atau problem.

- Adalah inti dari ilmu komputer.
- Algoritma adalah urutan-urutan dari instruksi atau langkah-langkah untuk menyelesaikan suatu masalah.
- Algoritma adalah blueprint dari program.
- Sebaiknya disusun sebelum membuat program.
- Kriteria suatu algoritma:

  Ada input dan output

  –Efektivitas dan efisien

  Terstruktur

Macam struktur dasar algoritma :

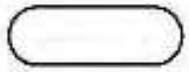
Pada dasarnya terdapat tiga buah struktur dasar yang menyusun suatu algoritma. Ketiga struktur dasar tersebut, yaitu :

- Sekuensial (runtun)
- Seleksi
- Pengulangan

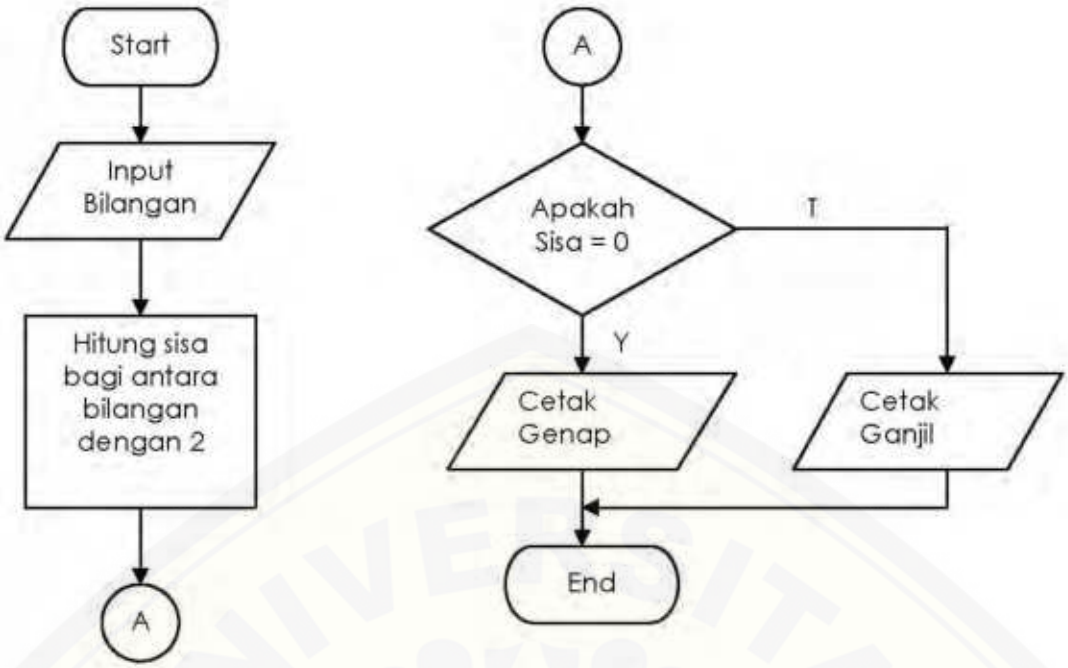
Contoh : Buat algoritma untuk menentukan apakah suatu bilangan merupakan bilangan ganjil atau bilangan genap.

Algoritmanya :

1. Masukkan sebuah bilangan sembarang
2. Bagi bilangan tersebut dengan bilangan 2
3. Hitung sisa hasil bagi pada langkah 2.
4. Bila sisa hasil bagi sama dengan 0 maka bilangan itu adalah bilangan genap tetapi bila sisa hasil bagi sama dengan 1 maka bilangan itu adalah bilangan ganjil.

SIMBOL	NAMA	FUNGSI
	**TERMINATOR**	Permulaan/akhir program
	GARIS ALIR (FLOW LINE)	Arah aliran program
	**PREPARATION**	Proses inisialisasi/ pemberian harga awal
	**PROSES**	Proses perhitungan/ proses pengolahan data
	**INPUT/OUTPUT DATA**	Proses input/output data, parameter, informasi
	**PREDEFINED PROCESS (SUB PROGRAM)**	Permulaan sub program/ proses menjalankan sub program
	**DECISION**	Perbandingan pernyataan, penyeleksian data yang memberikan pilihan untuk langkah selanjutnya
	**ON PAGE CONNECTOR**	Penghubung bagian-bagian flowchart yang berada pada satu halaman
	**OFF PAGE CONNECTOR**	Penghubung bagian-bagian flowchart yang berada pada halaman berbeda

Dari contoh algoritma di atas tentang menentukan apakah suatu bilangan adalah bilangan ganjil atau bilangan genap, flowchart dari program adalah sebagai berikut :



C. Alat-alat dan Komponen

- Komputer / Laptop
- Software program Dev C++

D. TUGAS PENDAHULUAN

Desainlah algoritma dan flowchart dari Prosedur Percobaan berikut ini :

E. PROSEDUR PERCOBAAN

1. Buat konversi dari suhu Celcius ke Fahrenheit
2. Tentukan bilangan terbesar dari dua buah bilangan x dan y
3. Menampilkan 4 buah baris yang berisi tulisan "Praktikum Pemrograman Komputer" secara berulang.

F. DATA HASIL PERCOBAAN

Praktikum Bab 2

PEMROGRAMAN BAHASA C

A. TUJUAN

- Mahasiswa mampu mengenal sintaks dan fungsi-fungsi dasar dalam bahasa C
- Mahasiswa mampu menyusun algoritma dan flowchart untuk memecahkan suatu masalah serta implementasi dalam bahasa C.

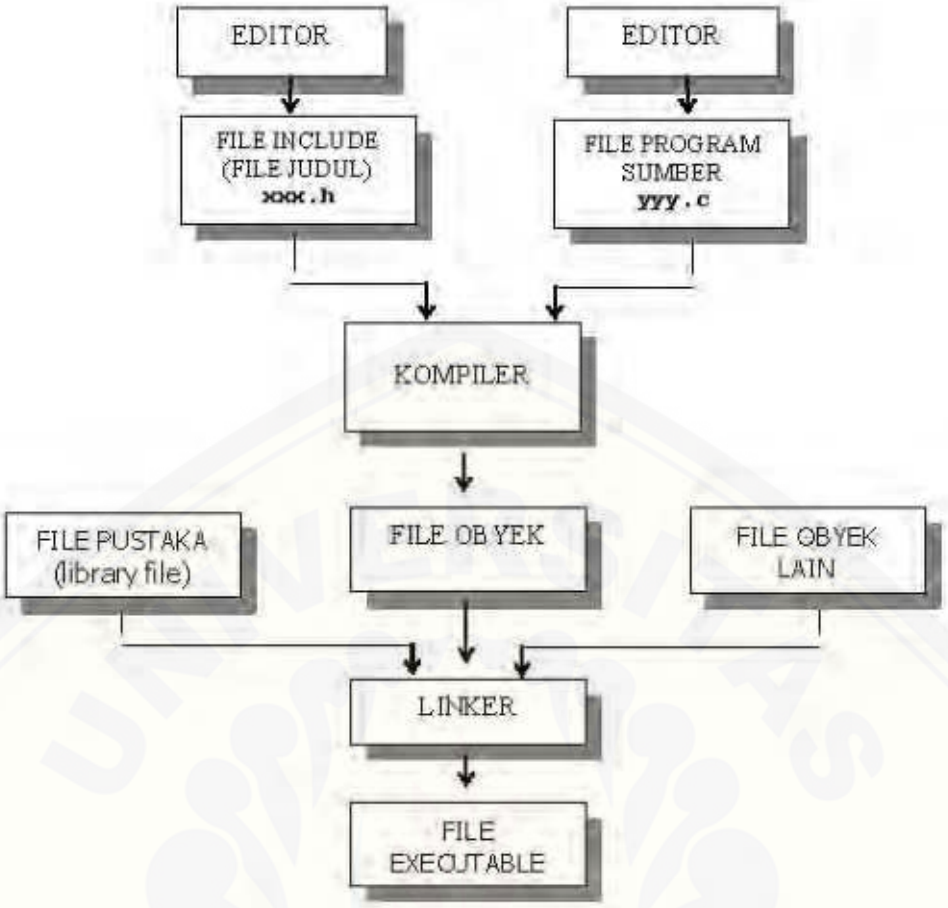
B. DASAR TEORI

Akar dari bahasa C adalah bahasa BCPL yang dikembangkan oleh Martin Richards pada tahun 1967. Bahasa ini memberikan ide kepada Ken Thompson yang kemudian mengembangkan bahasa yang disebut dengan B pada tahun 1970. Perkembangan selanjutnya dari bahasa B adalah bahasa C oleh Dennis Ritchie sekitar tahun 1970-an di Bell Telephone Laboratories Inc. (sekarang adalah AT&T Bell Laboratories). Bahasa C pertama kali digunakan pada komputer Digital Equipment Corporation PDP-11 yang menggunakan sistem operasi UNIX.

Standar bahasa C yang asli adalah standar dari UNIX. Sistem operasi, kompiler C dan seluruh program aplikasi UNIX yang esensial ditulis dalam bahasa C. Kepopuleran bahasa C membuat versi-versi dari bahasa ini banyak dibuat untuk komputer mikro. Untuk membuat versi-versi tersebut menjadi standar, ANSI *(American National Standards Institute)* membentuk suatu komite *(ANSI committee X3J11)* pada tahun 1983 yang kemudian menetapkan standar ANSI untuk bahasa C. Standar ANSI ini didasarkan kepada standar UNIX yang diperluas.

Proses Kompilasi dan Linking Program C

Proses dari bentuk *source program*, yaitu program yang ditulis dalam bahasa C hingga menjadi program yang *executable* ditunjukkan pada Gambar 1 di bawah ini.



Proses Kompilasi-*Linking* dari program C

Struktur Penulisan Program C

Program C pada hakekatnya tersusun atas sejumlah blok fungsi. Sebuah program minimal mengandung sebuah fungsi. Fungsi pertama yang harus ada dalam program C dan sudah ditentukan namanya adalah *main()*. Setiap fungsi terdiri atas satu atau beberapa pernyataan, yang secara keseluruhan dimaksudkan untuk melaksanakan tugas khusus. Bagian pernyataan fungsi (sering disebut tubuh fungsi) diawali dengan tanda kurung kurawal buka ({) dan diakhiri dengan tanda kurung kurawal tutup (}). Di antara kurung kurawal itu dapat dituliskan statemen-statemen program C. Namun pada kenyataannya, suatu fungsi bisa saja tidak mengandung pernyataan sama sekali. Walaupun fungsi tidak memiliki pernyataan, kurung kurawal haruslah tetap ada. Sebab kurung kurawal mengisyaratkan awal dan akhir definisi fungsi. Berikut ini adalah struktur dari program C

```
main()
{
        statemen-statemen;          fungsi utama
}

fungsi_fungsi_lain()
{
        statemen-statemen;          fungsi-fungsi lain yang
}                                   ditulis oleh pemrogram
```

Bahasa C dikatakan sebagai bahasa pemrograman terstruktur karena strukturnya menggunakan fungsi-fungsi sebagai program-program bagiannya *(subroutine)*. Fungsifungsi yang ada selain fungsi utama (*main()*) merupakan program-program bagian. Fungsi-fungsi ini dapat ditulis setelah fungsi utama atau diletakkan di file pustaka *(library)*. Jika fungsi-fungsi diletakkan di file pustaka dan akan dipakai di suatu program, maka nama file judulnya *(header file)* harus dilibatkan dalam program yang menggunakannya dengan *preprocessor directive* berupa *#include*.

Pengenalan Fungsi-Fungsi Dasar

a. Fungsi *main()*

Fungsi *main()* harus ada pada program, sebab fungsi inilah yang menjadi titik awal dan titik akhir eksekusi program. Tanda { di awal fungsi menyatakan awal tubuh fungsi dan sekaligus awal eksekusi program, sedangkan tanda } di akhir fungsi merupakan akhir tubuh fungsi dan sekaligus adalah akhir eksekusi program. Jika program terdiri atas lebih dari satu fungsi, fungsi *main()* biasa ditempatkan pada posisi yang paling atas dalam pendefinisian fungsi. Hal ini hanya merupakan kebiasaan. Tujuannya untuk memudahkan pencarian terhadap program utama bagi pemrogram. Jadi bukanlah merupakan suatu keharusan.

b. Fungsi *printf()*.

Fungsi *printf()* merupakan fungsi yang umum dipakai untuk menampilkan suatu keluaran pada layar peraga. Untuk menampilkan tulisan

```
Selamat belajar bahasa C
```

misalnya, pernyataan yang diperlukan berupa:

```
printf("Selamat belajar bahasa C");
```

Pernyataan di atas berupa pemanggilan fungsi *printf()* dengan argumen atau parameter berupa string. Dalam C suatu konstanta string ditulis dengan diawali dan diakhiri tanda petik-ganda ("). Perlu juga diketahui pernyataan dalam C selalu diakhiri dengan tanda titik koma (;). Tanda titik koma dipakai sebagai tanda pemberhentian sebuah pernyataan dan bukanlah sebagai pemisah antara dua pernyataan.

Tanda \ pada string yang dilewatkan sebagai argumen *printf()* mempunyai makna yang khusus. Tanda ini bisa digunakan untuk menyatakan karakter khusus seperti karakter baris-baru ataupun karakter *backslash* (miring kiri). Jadi karakter seperti \n sebenarnya menyatakan sebuah karakter. Contoh karakter yang ditulis dengan diawali tanda \ adalah:

\"	menyatakan karakter petik-ganda
\\	menyatakan karakter backslash
\t	menyatakan karakter tab

Dalam bentuk yang lebih umum, format *printf()*

```
printf("string kontrol", daftar argumen);
```

dengan string kontrol dapat berupa satu atau sejumlah karakter yang akan ditampilkan ataupun berupa penentu format yang akan mengatur penampilan dari argumen yang terletak pada daftar argumen. Mengenai penentu format di antaranya berupa:

%d	untuk menampilkan bilangan bulat (integer)
%f	untuk menampilkan bilangan titik-mengambang (pecahan)
%c	untuk menampilkan sebuah karakter
%s	untuk menampilkan sebuah string

Contoh:

```
#include <stdio.h>
main()
{
printf("NIP : %d\n", 2020);
printf("Nama : %s\n", "Adam");
printf("Nilai : %f\n",99.9);
printf("Huruf : %c\n",'A');
}
```

Pengenalan Praprosesor #include

#include merupakan salah satu jenis pengarah praprosesor *(preprocessor directive)*. Pengarah praprosesor ini dipakai untuk membaca file yang di antaranya berisi deklarasi fungsi dan definisi konstanta. Beberapa file judul disediakan dalam C. File-file ini mempunyai ciri yaitu namanya diakhiri dengan ekstensi **.h**. Misalnya pada program `#include <stdio.h>` menyatakan pada kompiler agar membaca file bernama *stdio.h* saat pelaksanaan kompilasi.

Bentuk umum #include:

```
#include "namafile"
```

Bentuk pertama (#include <namafile>) mengisyaratkan bahwa pencarian file dilakukan pada direktori khusus, yaitu direktori file *include*. Sedangkan bentuk kedua (#include "namafile") menyatakan bahwa pencarian file dilakukan pertama kali pada direktori aktif tempat program sumber dan seandainya tidak ditemukan pencarian akan dilanjutkan pada direktori lainnya yang sesuai dengan perintah pada sistem operasi.

Kebanyakan program melibatkan file **stdio.h** (file-judul I/O standard, yang disediakan dalam C). Program yang melibatkan file ini yaitu program yang menggunakan pustaka I/O (input-output) standar seperti *printf()*.

Komentar dalam Program

Untuk keperluan dokumentasi dengan maksud agar program mudah dipahami di suatu saat lain, biasanya pada program disertakan komentar atau keterangan mengenai program. Dalam C, suatu komentar ditulis dengan diawali dengan tanda /* dan diakhiri dengan tanda */.

Contoh :

```
#include <stdio.h>
main()
{
/*
ini adalah komentar untuk
multiple lines
*/
printf("NIP : %d\n", 2020); //ini adalah komentar baris 1
printf("Nama : %s\n", "Adam"); //ini adalah komentar baris 2
printf("Nilai : %f\n",99.9); //ini adalah komentar baris 3
printf("Huruf : %c\n",'A'); //ini adalah komentar baris 4
}
```

C. ALAT DAN KOMPONEN

- Komputer / Laptop
- Software program Dev C++

D. TUGAS PENDAHULUAN

Tuliskan desain algoritma dan flowchart untuk soal-soal di bawah ini :

1. Mencetak kalimat "`Hello World!`" dalam satu baris

2. Mencetak kalimat dalam beberapa baris, dengan tampilan sbb :

```
Hello...
..oh my
...when do i stop?
```

```
1
..2
...3
```

3. Menghitung penjumlahan 1024 + 1024 dan mencetak hasilnya

4. Mengisi nilai 2 variabel int, menjumlahkan kedua isi variabel tersebut dan mencetak hasilnya

5. Mengisi nilai sebuah variabel float, mengalikan isi variabel tersebut dengan 0,5 dan mencetak hasilnya

E. PERCOBAAN

Implementasikan semua desain yang telah dibuat dalam tugas pendahuluan menggunakan bahasa pemrograman C

F. DATA HASIL PERCOBAAN

(print screen script beserta hasil compile & running)

Praktikum Bab 3
DASAR PEMROGRAMAN C

A. TUJUAN

- Mahasiswa mampu menjelaskan tentang beberapa tipe data dasar (jenis dan jangkauannya)
- Mahasiswa mampu menjelaskan tentang Variabel
- Mahasiswa mampu menjelaskan tentang konstanta
- Mahasiswa mampu menjelaskan tentang berbagai jenis operator dan pemakaiannya
- Mahasiswa mampu menjelaskan tentang instruksi I/O

B. DASAR TEORI

1. Tipe Data Dasar

Data bisa dinyatakan dalam bentuk konstanta atau variabel.

- Konstanta ⇨ nilainya tetap.
- Variabel ⇨ nilainya dapat diubah-ubah selama eksekusi.

Berdasarkan jenisnya, data dapat dibagi menjadi lima kelompok ⇨ dinamakan tipe data dasar, yaitu:

- Bilangan bulat (integer)
- Bilangan real presisi-tunggal (float)
- Bilangan real presisi-ganda (double)
- Karakter (char)
- Tak-bertipe (*void*) Ukuran Memori untuk tipe data :

Tipe_data	Jumlah bit	Range nilai	Keterangan
char	8	-128 s/d 127	Karakter
int (signed int)	16	-32768 s/d 32767	Bilangan bulat (integer)
short int	16	-32768 s/d 32767	Bilangan bulat.
Unsigned int	16	0 s/d 65535	Bilangan bulat tak bertanda

long int	32	-2147483646 s/d 2147483647	Bilangan bulat
float	32	1.7E-38 s/d 3.4E+38	Bilangan real (single)
double	64	2.2E-308 s/d 1.7E+308	Bilangan real (double)
void	0	-	Tak bertipe

2. Variabel

Aturan penulisan Variabel :

- Nama harus diawali dengan huruf (*A..Z, a..z*) atau karakter garis bawah (_).
- Selanjutnya dapat berupa huruf, digit (0..9) atau karakter garis bawah atau tanda dollar ($).
- Panjang nama variabel boleh lebih dari 31 karakter ☞ hanya 31 karakter pertama yang akan dianggap.
- nama variabel tidak boleh menggunakan nama yang tergolong sebagai katakata cadangan (*reserved words*) seperti *printf, int, if, while* dan sebagainya

Deklarasi Variabel

- Variabel yang akan digunakan dalam program haruslah dideklarasikan terlebih dahulu ☞ pengertian deklarasi di sini berarti memesan memori dan menentukan jenis data yang bisa disimpan di dalamnya.
- Bentuk umum deklarasi variabel:

  tipe_data daftar_nama_variabel;
- Contoh:

```
int var_bulat1;
float var_pecahan1, var_pecahan2;
```

3. Konstanta

Konstanta menyatakan nilai tetap. Tidak perlu dideklarasikan. dan mempunyai tipe data.

Aturan penulisan:

- Konstanta karakter ☞ diawali dan diakhiri dengan tanda petik tunggal, Contoh : 'A' dan '*ii*'

- Konstanta integer ☞ ditulis dengan angka (tanpa tanda petik) tanpa mengandung pemisah ribuan dan tak mengandung bagian pecahan.
  Contoh :
  –1 dan 32767.
- Konstanta real (*float* dan *double*) bisa mengandung pecahan (dengan tanda berupa titik) dan nilainya bisa ditulis dalam bentuk eksponensial (menggunakan tanda e), contohnya : 27.5f (untuk tipe *float*) atau 27.5 (untuk tipe *double*) dan 2.1e+5 (maksudnya 2,1 x 10^5).
- Konstanta string merupakan deretan karakter yang diawali dan diakhiri dengan tanda petik-ganda (“). Contoh: “Program Dasar”.
- Menggunakan keyword #define
  #define <nama_konstanta> <nilai>
  atau
- Menggunakan keyword const
  const <tipe_konstanta> <nama_konstanta> = <nilai>;
  Contoh:
  #define PI 3.14159
  atau : **const** float PI = 3.14159;

4. Operator

Merupakan Simbol atau karakter , digunakan untuk melakukan sesuatu operasi atau manipulasi. Misal: menjumlahkan, mengurangi, membandingkan, memberikan nilai, dll.

Jenis operator:

- Operator Aritmatika
- Operator Increment dan Decrement
- Operator penugasan
- Operator kombinasi

1. Operator Aritmetika Terdiri dari dua jenis:

- **Operator binary**

Operator	Fungsi
*	Perkalian
/	Pembagian
%	Sisa Pembagian
+	Penjumlahan
-	Pengurangan

- **Operator unary**

  Tanda '-' (minus)

  Tanda '+' (plus)

2. Operator Increment dan Decrement
   - **Operator increment: '++'**
   - **Operator decrement: '--'**

operasi	arti
x++/++x	x=x+1
y--/--y	y=y-1

5. Instruksi I/O

Ada beberapa perintah yang digunakan untuk menjalankan instruksi I/O. Perintah perintah tersebut adalah sebagai berikut :

- Fungsi **printf**()
  - digunakan untuk menampilkan data ke layar.
  - Bentuk umum pernyataan *printf()*:
  - **printf("string kontrol", argumen1, argumen2,...);**
  - Format untuk data string dan karakter :

    %c ⇨ untuk menampilkan sebuah karakter

    %s ⇨ untuk menampilkan sebuah string

  - Format data bilangan :

`%u`	untuk menampilkan data bilangan tak bertanda *(unsigned)* dalam bentuk desimal
`%d` `%i`	untuk menampilkan bilangan integer bertanda *(signed)* dalam bentuk desimal
`%o`	untuk menampilkan bilangan bulat tak bertanda dalam bentuk oktal
`%x` `%X`	untuk menampilkan bilangan bulat tak bertanda dalam bentuk heksadesimal (`%x` → notasi yang dipakai : a, b, c, d, e dan f sedangkan `%X` → notasi yang dipakai : A, B, C, D, E dan F)
`%f`	untuk menampilkan bilangan real dalam notasi : dddd.dddddd
`%e` `%E`	untuk menampilkan bilangan real dalam notasi eksponensial
`%g` `%G`	untuk menampilkan bilangan real dalam bentuk notasi seperti `%f`, `%E` atau `%F` bergantung pada kepresisian data (digit `0` yang tak berarti tak akan ditampilkan)
l	merupakan awalan yang digunakan untuk `%d`, `%u`, `%x`, `%X`, `%o` untuk menyatakan long int (misal `%ld`). Jika diterapkan bersama `%e`, `%E`, `%f`, `%F`, `%g` atau `%G` akan menyatakan *double*
L	Merupakan awalan yang digunakan untuk `%f`, `%e`, `%E`, `%g` dan `%G` untuk menyatakan *long double*
h	Merupakan awalan yang digunakan untuk `%d`, `%i`, `%o`, `%u`, `%x`, atau `%X`, untuk menyatakan *short int*.

- Fungsi **scanf()**
  - Digunakan untuk menerima input data dari keyboard.
  - Bentuk *scanf()* menyerupai fungsi *printf()*.
  - Fungsi ini melibatkan penentu format yang pada dasarnya sama digunakan pada *printf()*.
  - Bentuk umum fungsi *scanf()* adalah:

    scanf("string kontrol", daftar_argumen);
  - daftar_argumen dapat berupa satu atau beberapa argumen dan haruslah berupa alamat.
  - Untuk menyatakan alamat dari variabel, di depan variabel dapat ditambahkan tanda & (tanda & dinamakan sebagai operator alamat)

    Contoh :

    scanf ("%f",&radius);

    scanf ("%d %d",&data1, &data2);

- Fungsi **puts()** dan **putchar()**
  - Fungsi **puts()** : menampilkan string

    puts ("Selamat mencoba");

    sama dengan **printf** ("Selamat mencoba\n");
  - Fungsi **putchar()** : menampilkan karakter

    putchar ('F'); sama dengan **printf** ("%c",'F');

- Fungsi **getch()** dan **getchar()**
  - Fungsi **getch()** : membaca karakter dan tidak ditampilkan.
  - Fungsi **getchar()** : membaca karakter dan ditampilkan.

    Contoh : kar = getchar(); scanf ("%c",&kar);

C. ALAT DAN KOMPONEN

- Komputer / Laptop
- Software program Dev C++

D. TUGAS PENDAHULUAN

1. Tipe-tipe data dasar dalam C.

```
/* File program : contoh prakt 3.c */
#include <stdio.h>
```

```
main()
{
int var_bulat = 321123;
float var_pecahan1 = 321.23456781;
double var_pecahan2 = 3.4567e  40;
char var_karakter = 'S';
printf("Variabel bulat    %d\n", var  bulat);
printf("Variabel pecahan1 = %f\n", var_pecahan1);
printf("Variabel pecahan2    %le\n", var  pecahan2);
printf("Variabel karakter = %c\n", var_karakter);
}
```

2. Mengenal beberapa operator dalam C.

```
/* File program : contoh prakt 3operator.c */
#include <stdio.h>
main()
{
int a, b, c, hasil;
printf("Masukkan nilai a = ");
scanf("%d", &a);
printf("Masukkan nilai b    ");
scanf("%d", &b);
printf("Masukkan nilai c    ");
scanf("%d", &c);
printf("\n");
hasil = a - b;
printf("Hasil pengurangan : a - b = %d\n", hasil);
printf("Hasil perkalian : b * c    %d\n", b * c);
hasil    a / c;
printf("Hasil pembagian : a / c    %d\n", hasil);
printf("Hasil operasi : a + b * c    %d\n",
a + b * c);
}
```

3. Operator Modulus

```
/* File program : prakt 3modulus.c */
#include <stdio.h>
main()
{
int a = 14, b = 2, c = 3, d = 4;
printf("a = %d, b = %d, c = %d, d = %d\n\n",
a, b, c, d);
printf("Hasil a %% b = %d\n", a % b);
printf("Hasil a %% c    %d\n", a % c);
printf("Hasil a %% d    %d\n", a % d);
printf("Hasil a / d * d + a %% d = %d\n",
a / d * d + a % d);
}
```

4. Menghitung diskriminan persamaa kuadrat ax^2 + bx + c = 0

```
/* File program : contoh prakt3diskrim.c */
# include <stdio.h>
main()
{
float a,b,c,d    0;
a = 3.0f;
b    4.0f;
c = 7.0f;
d = b*b-4*a*c;
printf("Diskriminan = %f\n",d);
}
```

5. Penggunaan pre & post Increment operator

```
/* File program : contoh prakt3pre  post.C */
#include <stdio.h>
main()
{
int count = 0, loop;
loop    ++count; /* count  count+1; loop  count; */
printf("loop = %d, count = %d\n", loop, count);
loop    count++; /* loop  count; count  count+1; */
printf("loop    %d, count    %d\n", loop, count);
}
```

6. Perbedaan format %g, %e dan %f

```
/* File program : contoh prak3form  efg.c */
#include <stdio.h>
main()
{
float x;
printf("Masukkan nilai pecahan yg akan ditampilkan : ");
scanf("%f", &x);
printf("format e   > %e\n", x);
printf("format f => %f\n", x);
printf("format g   > %g\n", x);
}
```

7. Penggunaan format panjang medan data

```
/* File program : contoh prakt3formatpjg.c */
#include <stdio.h>
main()
{
int nilai1 = 20;
float nilai2    500.0f;
printf("Abad %5d\n", nilai1);
```

```
printf("%10.2f\n", nilai2);
printf("%10s\n", "Bahasa C"); /* String rata kanan */
printf("%-10s\n", "Bahasa C"); /* String rata kiri */
}
```

8. Menghitung keliling dan luas lingkaran

```
/* File program : contoh prakt3lingkaran.c */
#include <stdio.h>
main()
{
float radius, keliling, luas;
printf("Masukkan jari-jari lingkaran : ");
scanf("%f",&radius);
keliling = 2 * 3.14f * radius;
luas = 3.14f * radius * radius;
printf("\nData lingkaran\n");
printf("Jari-jari = %8.2f\n", radius);
printf("Keliling = %8.2f\n", keliling);
printf("Luas = %8.2f\n", luas);
}
```

E. PERCOBAAN

Compile dan Jalankan script pada tugas pendahuluan menggunakan Dev-C selanjutnya lengkapi dengan memberi komentar pada masing-masing baris script, buat pembahasan dan kesimpulan dari masing-masing contoh 1-8 .

F. DATA HASIL PERCOBAAN

(print screen script serta hasil compile & running program disini)

Praktikum Bab 4
PENGAMBILAN KEPUTUSAN

A. TUJUAN

1. Mahasiswa mampu menjelaskan tentang operator kondisi (operator relasi dan logika)
2. Mahasiswa mampu menjelaskan penggunaan pernyataan *if*
3. Mahasiswa mampu menjelaskan penggunaan pernyataan *if-else*
4. Mahasiswa mampu menjelaskan penggunaan pernyataan *if* dalam *if*
5. Mahasiswa mampu menjelaskan penggunaan pernyataan *else-if*
6. Mahasiswa mampu menjelaskan penggunaan pernyataan *switch*

B. DASAR TEORI

Pengambilan keputusan diperlukan jika ada dua atau lebih kondisi yang harus dipilih salah satu. Pernyataan-pernyataan yang dapat digunakan dalam pengambilan keputusan adalah :

- Pernyataan *if*
- Pernyataan *if-else*
- Pernyataan *switch*

Pernyataan-pernyataan diatas memerlukan suatu kondisi yang dibentuk dengan operator relasi dan/atau operator logika.

1. Operator Relasi

Operator relasi menghasilkan kondisi *BENAR* atau *SALAH*.

Operator	Makna
>	Lebih dari
>=	Lebih dari atau sama dengan
<	Kurang dari
<=	Kurang dari atau sama dengan
==	Sama dengan
!=	Tidak sama dengan

Contoh :

Pembandingan		Hasil
`1 > 2`	→	Salah
`1 < 2`	→	Benar
`A == 1`	→	Benar, jika A bernilai 1
		Salah, jika A tidak bernilai 1
`'A' < 'B'`	→	Benar, karena kode ASCII untuk karakter 'A' kurang dari kode ASCII untuk karakter 'B'
`kar == 'Y'`	→	Benar, jika kar berisi 'Y'
		Salah, jika kar tidak berisi 'Y'

2. Operator Logika

Bentuk operator Logika adalah :

Operator	Makna
&&	dan (AND)
||	atau (OR)
!	tidak (NOT)

- Bentuk umum penggunaan operator logika '&&' dan '||':
  - *operand1 operator operand2*
- Bentuk umum penggunaan operator logika '!':
  - *!operand*

    Hasil operasi / bernilai :
    - Benar jika operand bernilai salah
    - Salah jika operand bernilai benar

④ Hubungan antar Operand pada Operator Logika

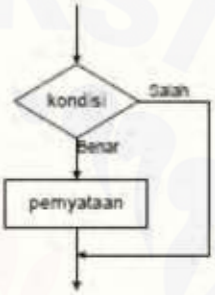
Operand1	Operand2	Hasil	
		||	&&
Salah	Salah	0	0
Salah	Benar	1	0
Benar	Salah	1	0
Benar	Benar	1	1

Prioritas dalam pengambilan keputusan untuk Operator Logika dan Relasi :

Prioritas	Operator
Tertinggi	!
	> >= < <=
	== !=
	&&
Terendah	||

- Pernyataan ***if***
  - Sintak

```
if (kondisi)
        pernyataan;
```



  - Jika kondisi benar maka pernyataan dilakukan
  - Pernyataan bisa satu statement atau beberapa statement
  - Jika pernyataan lebih dari satu gunakan tanda '{' dan '}' untuk mengelompokkan pernyataan-pernyataan itu.
- Pernyataan *if-else*
  - Sintak

```
if (kondisi)
        pernyataan1;
else
        pernyataan2;
```

  - Jika kondisi benar maka pernyataan 1 dilakukan.
  - Jika kondisi salah maka pernyataan 2 dilakukan.
- *Nested –if*
  - Di dalam pernyataan *if* (*if-else*) bisa terdapat pernyataan *if* (atau *if-else*) yang lain.
  - Sintak

```
if (kondisi-1)
        if (kondisi-2)
                .
                .
                if(kondisi-n)
                        pernyataan
                else
                        pernyataan;
                                .
                                .
        else
                pernyataan;
else
        pernyataan;
```

- Pernyataan *Switch-Case*
  - Digunakan sebagai pengganti pernyataan *if* bertingkat (*else-if*)
  - Sintak

```
switch (ekspresi)
{
  case konstanta-1:
                pernyataan-11;
                ......
                break;
  case konstanta-2:
                .
                .
  case konstanta-n:
                pernyataan-n1;
                ......
                break;
  default:
                ......
                break;
}
```

C. ALAT DAN KOMPONEN

- Komputer / Laptop
- Software program Dev C++

D. TUGAS PENDAHULUAN

1. Menghitung nilai discount menggunakan if.

```
/* File program : prakt4 discount.c */
#include <stdio.h>
main()
{
double total_pembelian, discount = 0;
printf("Total pembelian = Rp ");
scanf("%lf", &total_pembelian);
if(total_pembelian >= 100.000)
discount = 0.05 * total_pembelian;
printf("Besarnya discount = Rp %.2lf\n", discount);
}
```

2. Menghitung nilai absolut suatu bilangan.

```
/* File program :prakt4 absolut.c */
#include <stdio.h>
main()
{
int bil, abs;
printf("Masukkan bilangan bulat : ");
scanf("%d", &bil);
if(bil < 0)
abs = -bil;
printf("Nilai absolut dari %d adalah %d\n\n", bil, abs);
}
```

3. Pemakaian if-else untuk menyeleksi bilangan pembagi

```
/* File program :prakt4 bil_pembagi.c */
#include <stdio.h>
main()
{
float a, b;
printf("Masukkan nilai a : ");
scanf("%f", &a);
printf("Masukkan nilai b : ");
scanf("%f", &b);
printf("\n");
if (b == 0)
printf("%g dibagi dengan nol = TAK BERHINGGA\n", a);
else
printf("%g dibagi dengan %g = %g\n", a, b, a/b);
}
```

4. Pemakaian if-else untuk mengecek hasil modulus.

```
/* File program :prakt4 hasil_modulus.c */
#include <stdio.h>
main()
{
int bil1, bil2, sisa;
printf("Masukkan bilangan pertama : ");
scanf("%d", &bil1);
printf("Masukkan bilangan kedua : ");
scanf("%d", &bil2);
sisa = bil1 % bil2;
printf("\n");
if (sisa == 0)
printf("%d habis dibagi dengan %d\n\n", bil1, bil2);
else
printf("%d tidak habis dibagi dengan %d\n\n",
bil1, bil2);
}
```

5. Mengkategorikan karakter masukan

```
/* File program :prakt4 karakter_masukan.c */
#include <stdio.h>
main()
{
char karakter;
printf("Masukkan sebuah karakter : ");
scanf("%c", &karakter);
if ((karakter >= 'a' && karakter <= 'z') | (karakter >=
'A' && karakter <= 'Z'))
printf("%c adalah karakter alphabet\n", karakter);
else if (karakter >= '0' && karakter <= '9')
printf("%c adalah bilangan\n", karakter);
else
printf("%c adalah karakter khusus\n", karakter);
}
```

6. Mengkategorikan bilangan bulat dengan memberinya tanda 1, 0 atau -1.

```
/* File program :prakt4 bulat_ditandai.c */
#include <stdio.h>
main()
{
int bil, tanda;
printf("Masukkan sebuah bilangan : ");
scanf("%d", &bil);
if (bil < 0)
tanda = -1;
else if (bil      0)
tanda = 0;
else
tanda = 1;
printf("Bilangan %d memiliki tanda %d\n", bil, tanda);
}
```

7. Implementasi program kalkulator sederhana menggunakan else-if.

```
/* File program :prakt4 kalkulator.c */
#include <stdio.h>
main()
{
/* valid_operator diinisialisasi dg logika 1 */
int valid_operator = 1;
char operator;
float bil1, bil2, hasil;
printf("Masukkan 2 buah bilangan dan sebuah operator\n");
printf("dengan format : bil1 operator bil2\n\n");
scanf("%f %c %f", &bil1, &operator, &bil2);
if(operator == '*')
hasil = bil1 * bil2;
else if(operator == '/')
```

```
hasil = bil1 / bil2;
else if(operator == '+')
hasil = bil1 + bil2;
else if(operator == '-')
hasil = bil1 - bil2;
else
valid_operator = 0;
if(valid_operator)
printf("%g %c %g adalah %g\n", bil1, operator, bil2,
hasil);
else
printf("Invalid operator!\n");
}
```

8. Implementasi program kalkulator sederhana menggunakan switch - case.

```
/* File program :prak14 switch_case_kalkulator.c */
#include <stdio.h>
main()
{
/* valid_operator diinisialisasi dg logika 1 */
int valid_operator = 1;
char operator;
float bil1, bil2, hasil;
printf("Masukkan 2 buah bilangan dan sebuah operator\n");
printf("dengan format : bil1 operator bil2\n\n");
scanf("%f %c %f", &bil1, &operator, &bil2);
switch(operator) {
case '*' : hasil = bil1 * bil2; break;
case '/' : hasil = bil1 / bil2; break;
case '+' : hasil = bil1 + bil2; break;
case '-' : hasil = bil1 - bil2; break;
default : valid_operator = 0;
}
if(valid_operator)
printf("%g %c %g is %g\n", bil1, operator, bil2,hasil);
else
printf("Invalid operator!\n");
}
```

E. PERCOBAAN

Compile dan Jalankan script pada tugas pendahuluan menggunakan Dev-C selanjutnya lengkapi dengan memberi komentar pada masing-masing baris script, buat pembahasan dan kesimpulan dari masing-masing contoh 1-8 .

F. DATA HASIL PERCOBAAN

(print screen script serta hasil compile & running program disini)

Praktikum Bab 5
PERULANGAN PROSES

A. TUJUAN

1. Mahasiswa mampu menjelaskan proses perulangan menggunakan pernyataan *for*
2. Mahasiswa mampu menjelaskan proses perulangan menggunakan pernyataan *while*
3. Mahasiswa mampu menjelaskan proses perulangan menggunakan pernyataan *do-while*
4. Mahasiswa mampu menjelaskan penggunaan pernyataan *break*
5. Mahasiswa mampu menjelaskan penggunaan pernyataan *continue*
6. Mahasiswa mampu menjelaskan penggunaan pernyataan *goto*
7. Mahasiswa mampu menjelaskan *loop* di dalam *loop* (*nested loop*) dan contoh kasusnya
8. Mahasiswa mampu menjelaskan penggunaan *exit()* untuk menghentikan eksekusi program dan contoh kasusnya

B. DASAR TEORI

1. Pernyataan **for**

Digunakan untuk membuat looping dengan jumlah perulangan yang ditentukan di awal.

- Sintak:

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

Keterangan :

- Ungkapan1: digunakan untuk memberikan inisialisasi terhadap variabel pengendali *loop*.
- Ungkapan2: dipakai sebagai kondisi untuk keluar dari *loop*.
- Ungkapan3: dipakai sebagai pengatur kenaikan nilai variabel pengendali *loop*.

Contoh :

```
for (bil = 1; bil <= 15; bil += 3)
    printf("%d\n", bil);
```

maka akan didapatkan hasil :

1
4
7
10
13

2.Pernyataan **while**

Pada pernyataan ini, pengecekan terhadap loop dilakukan di bagian awal. Pernyataan di dalamnya bisa tidak dikerjakan sama sekali.

- Sintak

```
while(kondisi)
        pernyataan;
```

- Selama kondisi benar maka pernyataan dikerjakan
- Jika kondisi salah ⇒ keluar dari loop

Contoh :

```
bil = 1;
while (bil <= 15)
 {
    printf("%d\n", bil);
    bil = bil + 3;
 }
```

Akan didapatkan hasil :

1
4
7
10
13

3. Pernyataan **do-while**

Pada paernyataan ini, pengecekan terhadap loop dilakukan di bagian akhir. Pernyataan didalamnya pasti dijalankan (minimal 1 kali).

- Sintak :

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

- Mula-mula pernyataan dijalankan, selanjutnya kondisi diuji jika benar dilakukan perulangan, jika salah maka keluar dari loop

Contoh :

```
bil = 1;  do
{
```

```
    printf("%d\n", bil);
    bil = bil + 3;
} while  (bil <= 15);
```

Akan didapatkan hasil :

1
4
7
10
13

4. Pernyataan **break**

Berfungsi untuk keluar dari loop untuk looping dengan **for, while**, dan **do-while**. Juga berfungsi untuk keluar dari struktur **switch.**

- Sintak:

```
break;
```

Contoh :

Pada **loop:**

```
while(kondisi)
{
    break;
}
statement-x;
```

Pada **switch:**

```
switch (ekspresi)  {
. . . . .   case
konstanta-2:
    pernyataan-21;
    break;
. . . . .
}
```

Pernyataan Continue

Pada **loop:**

```
while(kondisi)
{
continue;
}
statemen-
x;
```

5. Pernyataan **goto**

Pernyataan ini berfungsi untuk mengarahkan eksekusi ke pernyataan yang diawali dengan suatu label.

- Sintak :

```
goto nama_label;

label :
```

6. **Nested-loop**

Adalah loop di dalam loop. Kondisi ini hampir sama dengan nested-if.

- Sintak :

```
for(ungkapan1; ungkapan2; ungkapan3)
{
    for(ungkapan4; ungkapan5; ungkapan6)
    {
        for(ungkapan-x; ungkapan-y; ungkapan-z)
        {
        Pernyataan1;
        .
        .
        }
    Pernyataan-n;
    .
    }
Pernyataan-m;
.
}
```

C. ALAT DAN KOMPONEN

- Komputer / Laptop
- Software program Dev C++

D. TUGAS PENDAHULUAN

1. Menghitung jumlah delapan triangular tanpa pernyataan for()

```
/* File program :prakt5 triangular.c */
#include <stdio.h>
main()
{
int jumlah = 0;
jumlah = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8;
printf("Jumlah delapan triangular adalah %d\n",jumlah);
}
```

2. Pemakaian for untuk membentuk deret naik.

```
/* File program :prakt5 deret_naik.c */
#include <stdio.h>
main()
{
int bilangan; for(bilangan = 20; bilangan <=
  100; bilangan =  10)
printf("%d\n", bilangan);
}
```

3. Pemakaian for untuk membentuk deret turun

```
/* File program :prakt5 deret_turun.c */
#include <stdio.h>
main()
{
int bilangan;
for (bilangan = 60; bilangan >= 10; bilangan -= 10)
printf("%d\n", bilangan);
}
```

4. Menghitung jumlah 200 triangular menggunakan for().

```
/* File program :prakt5 for_triangular.c */
#include <stdio.h>
main()
{
int n, jumlah = 0;
for (n=1; n<=200; n++)
jumlah = jumlah + n;
printf("Jumlah 200 triangular adalah %d\n", jumlah);
}
```

5. Membuat tabel dari jumlah triangular yang diinputkan

```
/* File program :prakt5 tabel_triangular.c */
#include <stdio.h>
main()
{
int n, bil, jumlah = 0;
printf("Masukkan bilangan triangular : ");
scanf("%d", &bil);
printf("\nTABEL PENJUMLAHAN TRIANGULAR\n\n");
printf("%3s%10s\n\n","n","Jumlah");
for (n=1; n<=bil; n++)
{
jumlah = jumlah + n;
printf("%3d %7d\n", n, jumlah);
}
}
```

6. Menghitung jumlah kata dan karakter dalam suatu kalimat

```
/* File program :prakt5 jumlah karakter.c */
#include <stdio.h>
main()
{
char kar;
int junkar = 0, junspas = 0;
puts("Masukkan sebuah kalimat dan akhiri dengan
ENTER");
puts("Program akan menghitung jumlah karakter ");
puts("Pada kalimat \n");
while((kar = getchar()) != '\n')
{
junkar++;
if (kar == ' ') junspas++;
}
printf("\nJumlah karakter = %d", junkar);
printf("\nJumlah spasi = %d\n\n", junspas);
}
```

7. Membalik angka menggunakan pernyataan while().

```
/* File program :prakt5 balik angka.c */
#include <stdio.h>
main()
{

int bil, digit_kanan;
printf("Masukkan bilangan yang akan dibalik : ");
scanf("%d", &bil);
printf("Hasil pembalikannya = ");
while(bil != 0)
{
digit_kanan = bil % 10;
printf("%d", digit_kanan);
bil = bil / 10;
}
printf("\n");
}
```

8. Membaca tombol yang ditekan.

```
/*File program :prakt5 membaca_tombol.c */
#include <stdio.h>
main()
{
char pilihan;
int sudah_benar = 0;
printf("Pilihlah Y atau T.\n");

while(!sudah_benar)
{
pilihan = getchar();
sudah_benar = (pilihan == 'Y') || (pilihan == 'y') ||
(pilihan == 'T') || (pilihan == 't');
}

switch(pilihan)
```

```
}
case 'Y':
case 'y':
puts("\nPilihan anda adalah Y");
break;
case 'T':
case 't':
puts("\nPilihan anda adalah T");
}
}
```

9. Pemakaian break untuk keluar dari looping

```
/* File program :prak5 keluar_looping.c */
#include <stdio.h>
main()
{
char kar;
printf("Ketik sembarang kalimat");
printf(" dan akhiri dengan ENTER\n\n");
for ( ; ; )
{
kar = getchar();
if(kar == '\n')
break;
}
printf("Selesai\n");
}
```

10. Loop for bersarang untuk membuat tabel perkalian

```
/* File program :prak5 tabel_kali.c */
#include <stdio.h>
#define MAKS 9
main()
{
int baris, kolom, hasil_kali;
for (baris = 1; baris <= MAKS; baris++)
{
for (kolom = 1; kolom <= MAKS; kolom++)
{
hasil_kali = baris * kolom;
printf ("%4d", hasil_kali);
}
printf("\n");
}
}
```

11. Menampilkan bilangan ganjil antara 7 - < 25 kecuali 15 menggunakan continue.

```
/*prak5 kecuali.c */
#include <stdio.h>
main()
{
```

```
int x = 5;
do
{
x = x + 2;
if (x == 15)
continue;
printf("%d""\n", x);
}
while (x < 25);
printf("\n");
}
```

E. PERCOBAAN

Compile dan Jalankan script pada tugas pendahuluan menggunakan Dev-C selanjutnya lengkapi dengan memberi komentar pada masing-masing baris script, buat pembahasan dan kesimpulan dari masing-masing contoh 1-11 .

F. DATA HASIL PERCOBAAN

(print screen script serta hasil compile & running program disini)

Praktikum Bab 6
PERULANGAN PROSES (lanjutan bab 5)

A. TUJUAN

1. Mahasiswa mampu menjelaskan proses perulangan menggunakan pernyataan *do-while*
2. Mahasiswa mampu menjelaskan penggunaan pernyataan *goto*
3. Mahasiswa mampu menjelaskan *loop* di dalam *loop* (*nested loop*) dan contoh kasusnya
4. Mahasiswa mampu menjelaskan penggunaan *exit()* untuk menghentikan eksekusi program dan contoh kasusnya

B. DASAR TEORI

1. Pernyataan **do-while**

Pada paernyataan ini, pengecekan terhadap loop dilakukan di bagian akhir.

Pernyataan didalamnya pasti dijalankan (minimal 1 kali).

- Sintak :

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

- Mula-mula pernyataan dijalankan, selanjutnya kondisi diuji jika benar dilakukan perulangan, jika salah maka keluar dari loop

Contoh :

```
bil = 1;
do {
    printf("Hallo world\n");
bil++;
 }
while (bil <= 5);
```

Akan menampilkan hasil :

```
Hallo world
Hallo world
Hallo world
Hallo world
Hallo world
```

2. Pernyataan for(; ;)

Terkadang dijumpai adanya pernyataan for yang tidak mengandung bagian ungkapan yang lengkap (beberapa ungkapan dikosongkan). Hal ini disebabkan ungkapan-ungkapan trsebut sudah di-inisialisasi di luar *for* atau dapat dikerjakan di dalam loop itu sendiri.

Contoh :

```
for (bil=10;bil<=60;bil++)
```

dapat diganti menjadi :

```
bil=10;
for( ; ; )
{
  bil++;
  if(bil==60)
  break;
}
```

3. Pernyataan exit()

Pernyataan ini digunakan untuk keluar dari program. Biasa disertakan pada program yang menggunakan looping dengan *for, while* atau *do-while*

Didefinisikan di **stdlib.h**

Sintak:

```
exit();
```

Contoh:

```
   /*tekan ESC untuk menghentikan program*/
#include<stdio.h>
#include<stdlib.h>
main()
    {
  puts("Tekan  ESC  untuk  menghentikan  program.");
  for( ; ; )
      if(getch()==27)
            exit(0);
    }
```

4. Pernyataan **goto**

Pernyataan ini berfungsi untuk mengarahkan eksekusi ke pernyataan yang diawali dengan suatu label. Label sendiri berupa suatu pengenal (*identifier*) yang diikuti dengan tanda titik dua (:).

- Sintak :

```
goto nama_label;

label :
```

Contoh :

```
bil++;              //naikkan nilai bil sebesar 1
if( bil<= 10)       //jika bil kurang atau sama dengan 10
     goto label_cetak;      //eksekusi menuju label cetak
label_cetak:
```

5. Nested-loop

Adalah loop di dalam loop. Kondisi ini hampir sama dengan nested-if.

- Sintak :

```
for(ungkapan1; ungkapan2; ungkapan3)
{
     for(ungkapan4; ungkapan5; ungkapan6)
     {
          for(ungkapan-x; ungkapan-y; ungkapan-z)
          {
          Pernyataan1;
          .
          .
          }
     Pernyataan-n;
     .
     }
Pernyataan-m;
.
}
```

C. ALAT DAN KOMPONEN

- Komputer / Laptop
- Software program Dev C++

D. TUGAS PENDAHULUAN

1. Membuat Tabel Faktorial menggunakan do-while

```
//menghitung nilai faktorial menggunakan do-while
#include<stdio.h>
main()
{
int k,i=1,fak=1;
printf("Masukkan nilai yang akan difaktorialkan = ");
scanf("%d",&k);
printf("\n\nTabel faktorial \n ");
printf("%2s %5s","k","faktorial\n");
do
{
fak=fak*i;
printf("%3d %5d \n",i,fak);
i++;
}while(i<=k);
}
```

2. Menentukan bilangan terbesar dari 5 buah data

```
/*prakt6 cari_bilangan_terbesar.c*/
#include<stdio.h>
main()
{
int i,j,k;
k=0;
printf("Masukkan sembarang bilangan bulat \n");
for(i=1;i<=5;i++)
{
printf("\nData ke- %d adalah - ",i);
scanf("%d",&j);
if(j>k)
k=j;
}

printf("Bilangan terbesar adalah : %d\n",k);
}
```

3. Menjumlahkan bilangan dengan hasil < 25

```
//prakt6 goto.c
#include<stdio.h>
main()
{
int nil,i,jumlah;
printf("Masukkan nilai maksimal yang dijumlah : ");
scanf("%d",&nil);
jumlah=0;
for(i=0;i<=nil;i++)
{
jumlah=jumlah+i;
if(jumlah>=25)
goto selesai;
}
printf("Jumlah = %d\n",jumlah);
selesai:
printf("selesai\n");
}
```

4. Menampilkan bilangan prima dari 3 s/d bil tertentu

```
//prakt6 bil_prima.c
#include<stdio.h>
main()
{
int n,i,j;
printf("Masukkan nilai terakhir yang diinginkan - ");
scanf("%d",&j);
for(n=2;n<=j;n++)
{
for(i=2;i<=n/2;i++)
{
if(n%i == 0)
break;
}
if(n%i != 0)
printf("%2d\n",n);
}
```

5. Menghitung harga total pembelian minuman

```
//prakt6 total_pembelian.c
#include<stdio.h>
main()
{
int jumlah,kode;
float harga,total;
double bayar;
char kar;
printf("kode Jenis Harga\n");
printf("1 Milo Rp 10.000\n");
printf("2 Kopi Rp. 5000\n");
printf("3 Coca Cola Rp 2500\n");
printf("4 Orange Juice Rp 2000\n");
total=0;
do
{
printf("Masukkan kode minuman : \n");
scanf("%d",&kode);
printf("Masukkan jumlah pesanan ");
scanf("%d",&jumlah);
if(kode==1)
harga = (float) 10000*jumlah;
else if(kode==2)
harga = (float)5000*jumlah;
else if(kode==3)
harga = (float)2500*jumlah;
else if(kode==4)
harga = (float)2000*jumlah;
total=total+harga;
printf("Mau menambah pesanan ? (Y/T)\n");
scanf("%s",&kar);
} while(kar=='Y');
if(total > 100000)
bayar=total-(0.15*total);
else
bayar=total;
printf("\nHarga yang harus dibayar = %10.2f\n",bayar);
}
```

E. PERCOBAAN

Compile dan Jalankan script pada tugas pendahuluan menggunakan Dev-C selanjutnya lengkapi dengan memberi komentar pada masing-masing baris script, buat pembahasan dan kesimpulan dari masing-masing contoh 1-5.

F. DATA HASIL PERCOBAAN

(print screen script serta hasil compile & running program disini)

Praktikum Bab 7

FUNGSI

A. TUJUAN

1. Mahasiswa mampu menjelaskan tentang prinsip dasar fungsi.
2. Mahasiswa mampu menjelaskan tentang parameter formal dan parameter actual.

B. DASAR TEORI

Fungsi adalah suatu bagian dari program yang dirancang untuk melaksanakan tugas tertentu dan letaknya dipisahkan dari program yang menggunakannya. Elemen utama dari program bahasa C berupa fungsi-fungsi, dalam hal ini program dari bahasa C dibentuk dari kumpulan fungsi pustaka (standar) dan fungsi yang dibuat sendiri oleh pemrogram. Fungsi banyak digunakan pada program C dengan tujuan :

a. Program menjadi terstruktur, sehingga mudah dipahami dan mudah dikembangkan. Dengan memisahkan langkah-langkah detail ke satu atau lebih fungsi-fungsi, maka fungsi utama (*main()*) menjadi lebih pendek, jelas dan mudah dimengerti.
b. dapat mengurangi pengulangan (duplikasi) kode. Langkah-langkah program yang sama dan dipakai berulang-ulang di program dapat dituliskan sekali saja secara terpisah dalam bentuk fungsi-fungsi. Selanjutnya bagian program yang membutuhkan langkah-langkah ini tidak perlu selalu menuliskannya, tetapi cukup memanggil fungsi-fungsi tersebut.

Dasar Fungsi

Fungsi standar C yang mengemban tugas khusus contohnya adalah ;

- *printf()* , yaitu untuk menampilkan informasi atau data ke layar.
- *scanf()* , yaitu untuk membaca kode tombol yang diinputkan.

Pada umumnya fungsi memerlukan nilai masukan atau parameter yang disebut sebagai argumen. Nilai masukan ini akan diolah oleh fungsi. Hasil akhir fungsi berupa sebuah nilai (disebut sebagai *return value* atau nilai keluaran fungsi). Oleh karena itu fungsi sering digambarkan sebagai "kotak gelap" seperti ditunjukkan pada gambar di bawah ini.

Penggambaran sebagai kotak gelap di antaranya menjelaskan bahwa bagian dalam fungsi bersifat pribadi bagi fungsi. Tak ada suatu pernyataan di luar fungsi yang bisa mengakses bagian dalam fungsi, selain melalui parameter (atau variabel eksternal yang akan dibahas belakangan). Misalnya melakukan *goto* dari pernyataan di luar fungsi ke pernyataan dalam fungsi adalah tidak diperkenankan.

Bentuk umum dari definisi sebuah fungsi adalah sebagai berikut ;

```
tipe-keluaran-fungsi  nama-fungsi (deklarasi argumen)
{
    tubuh fungsi
}
```

Keterangan :

- **tipe-keluaran-fungsi**, dapat berupa salah satu tipe data C, misalnya *char* atau *int* . Kalau penentu tipe tidak disebutkan maka dianggap bertipe *int* (secara *default*).
- **tubuh fungsi** berisi deklarasi variabel (kalau ada) dan statemen-statemen yang akan melakukan tugas yang akan diberikan kepada fungsi yang bersangkutan. Tubuh fungsi ini ditulis di dalam tanda kurung kurawal buka dan kurung kurawal tutup.

Sebuah fungsi yang sederhana bisa saja tidak mengandung parameter sama sekali dan tentu saja untuk keadaan ini deklarasi parameter juga tidak ada.

Contoh :

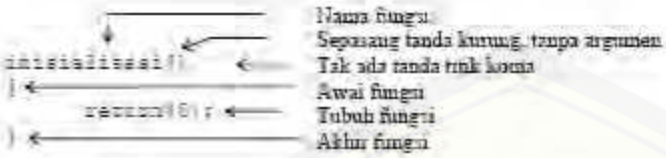
```
int inisialisasi()
{
    return(0);
}

inisialisasi()
{
    return(0);
}
```

Pada fungsi di atas :

- tipe keluaran fungsi tidak disebutkan, berarti keluaran fungsi ber tipe *int*.

- `inisialisasi` adalah nama fungsi
- Tanda `()` sesudah nama fungsi menyatakan bahwa fungsi tak memiliki parameter.
- Tanda `{` dan `}` adalah awal dan akhir fungsi
- `return(0)` merupakan sebuah pernyataan dalam tubuh fungsi.

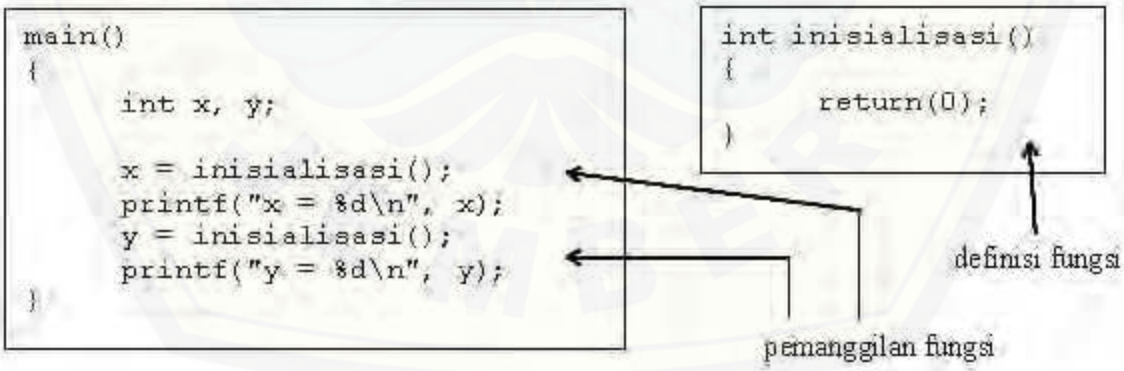


Memberikan Nilai Keluaran Fungsi

Suatu fungsi dibuat untuk maksud menyelesaikan tugas tertentu. Suatu fungsi dapat hanya melakukan suatu tugas saja tanpa memberikan suatu hasil keluaran atau melakukan suatu tugas dan kemudian memberikan hasil keluaran. Fungsi yang hanya melakukan suatu tugas saja tanpa memberikan hasil keluaran misalnya adalah fungsi untuk menampilkan hasil di layar.

Dalam tubuh fungsi, pernyataan yang digunakan untuk memberikan nilai keluaran fungsi berupa *return*. Sebagai contoh, pada fungsi **inisialisasi()** di atas terdapat pernyataan

`return(0);` merupakan pernyataan untuk memberikan nilai keluaran fungsi berupa nol.



Program di atas sekaligus menjelaskan bahwa suatu fungsi cukup didefinisikan satu kali tetapi bisa digunakan beberapa kali. Pada keadaan semacam ini seandainya tubuh fungsi mengandung banyak pernyataan, maka pemakaian fungsi dapat menghindari duplikasi kode dan tentu saja menghemat penulisan program maupun kode dalam memori.
Misalnya pada saat pernyataan

x= inisialisasi(); dijalankan, mula-mula eksekusi akan diarahkan ke fungsi **inisialisasi()**, selanjutnya suatu nilai keluaran (hasil fungsi) akhir fungsi diberikan ke **x**. Proses yang serupa, dilakukan untuk pernyataan

```
y= [illegible]();
```

Bagi suatu fungsi, jika suatu pernyataan *return* dieksekusi, maka eksekusi terhadap fungsi akan berakhir dan nilai pada parameter *return* akan menjadi keluaran fungsi. Untuk fungsi yang tidak memiliki pernyataan *return*, tanda } pada bagian akhir fungsi akan menyatakan akhir eksekusi fungsi.

Di bawah ini diberikan contoh sebuah fungsi yang mengandung dua buah pernyataan *return*. Fungsi digunakan untuk memperoleh nilai minimum di antara 2 buah nilai yang menjadi parameternya.

```
Int minimum(int x, int y)
{
    if (x < y)
     return(x);
    else
          return(y);
}
```

Pada fungsi di atas terdapat dua buah parameter berupa **x** dan **y**. Oleh karena itu fungsi juga mengandung bagian untuk mendeklarasikan parameter, yang menyatakan **x** dan **y** bertipe *int*. Adapun penentuan nilai keluaran fungsi dilakukan pada tubuh fungsi, berupa pernyataan

```
if (x < y)
 return(x);     else

     return(y);
```

yang menyatakan :

- jika x < y maka nilai keluaran fungsi adalah sebesar nilai x.
- untuk keadaan lainnya (x >= y) maka keluaran fungsi adalah sebesar y.

Fungsi Dengan Keluaran Bukan Integer

Untuk fungsi yang mempunyai keluaran bertipe bukan integer, maka fungsi haruslah didefiniskan dengan diawali tipe keluaran fungsinya (ditulis di depan nama fungsi). Sebagai contoh untuk menghasilkan nilai terkecil di antara dua buah nilai real, maka definisinya berupa :

```
Ilea. minimum(Ilea. x, Ilea. y)
```

```
{
    if (x < y)
    return(x);
    else
          return(y);
}
```

Perhatikan, di depan nama **minimum** diberikan tipe keluaran fungsi berupa *float*. Seluruh parameter sendiri juga didefinisikan dengan tipe *float*.

Khusus untuk fungsi yang dirancang tanpa memberikan nilai keluaran (melainkan hanya menjalankan suatu tugas khusus) biasa didefinisikan dengan diawali kata kunci *void* (di depan nama fungsi).

Prototipe Fungsi

Prototipe fungsi digunakan untuk menjelaskan kepada kompiler mengenai:

- tipe keluaran fungsi
- jumlah parameter
- tipe dari masing-masing parameter.

Bagi kompiler, informasi dalam prototipe akan dipakai untuk memeriksa keabsahan (validitas) parameter dalam pemanggilan fungsi. Salah satu keuntungannya adalah, kompiler akan melakukan konversi seandainya antara tipe parameter dalam fungsi dan parameter saat pemanggilan fungsi tidak sama, atau akan menunjukan kesalahan bila jumlah parameter dalam definisi dan saat pemanggilan berbeda.

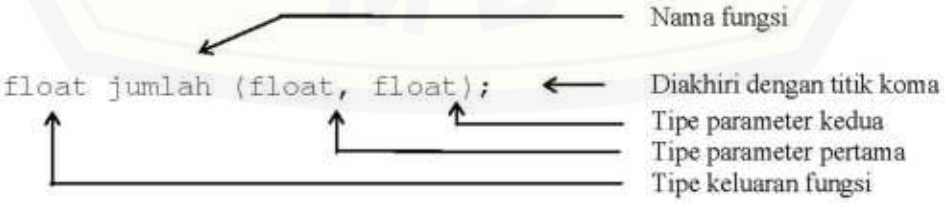
Contoh prototipe fungsi;

```
float jumlah (float x, float y);
```

atau

```
float jumlah (float, float);
```

Penjelasannya adalah sbb :



Untuk fungsi yang tidak memiliki argumen (contoh program **void.c**), maka deklarasinya adalah

```
void info_program(void);
```

menyatakan bahwa
info_program()
tidak memiliki parameter

Catatan :

- Untuk fungsi-fungsi pustaka, prototipe dari fungsi-fungsi berada di file-file judulnya *(header file)*. Misalnya fungsi pustaka *printf()* dan *scanf()* prototipenya berada pada file dengan nama **stdio.h**
- Untuk fungsi pustaka pencantuman pada prototipe fungsi dapat dilakukan dengan menggunakan *preprocessor directive* **#include**.

Parameter Formal dan Parameter Aktual

Parameter formal adalah variabel yang ada pada daftar parameter dalam definisi fungsi. Pada contoh program di atas misalnya, maka dalam fungsi **jumlah()** variabel **x** dan **y** dinamakan sebagai parameter formal. Adapun parameter aktual adalah parameter (tidak selalu berupa variabel) yang dipakai dalam pemanggilan fungsi.

```
main()
{
      ...
      c = jumlah(a, b);
      ...
}
```

parameter aktual

```
float jumlah(float x, float y)
{
     return(x + y);
}
```

parameter formal

Pada pernyataan :

```
x = jumlah(a, b);
y = jumlah(20.1, 45.6);
```

a dan **b** merupakan parameter aktual dari fungsi **jumlah()** dalam hal ini parameter berupa variabel. Demikian juga **20.1** dan **45.6** adalah parameter aktual, dalam hal ini berupa konstanta. Bahkan bisa juga parameter aktual berupa ungkapan yang melibatkan operator, misalnya :

```
printf("%g\n", jumlah(2+3, 3+6));
```

ungkapan

C. ALAT DAN KOMPONEN

- Komputer / Laptop
- Software program Dev C++

D. TUGAS PENDAHULUAN

1. Fungsi dengan nilai akhir

```
//prakt7 fungsi_dengan_nilai_akhir.c
#include<stdio.h>
main()
{
int hasil;
int a=20;
int b=44;
hasil=minimum(a,b);
printf("Nilai minimum= %d\n",hasil);
printf("Nilai minimum= %d\n",minimum(12,15));
}
int minimum(int x,int y)
{
if(x<y)
return(x);
else return(y);
}
```

2. Fungsi bertipe non integer (float)

```
//prakt7 fungsi_tipe_noninteger_float.c
#include<stdio.h>
float minimum(float x,float y);
main()
{
float hasil; float a=20.5,b=44.5;
hasil=minimum(a,b);
printf("Nilaiminimum= %g\n",hasil);
printf("Nilaiminimum= %g\n",minimum(3.5,2.5));
}
float minimum(float x,float y)
{
if(x<y)
return(x);
else return(y);
}
```

3. Fungsi bertipe void (tanpa nilai keluaran)

```
//prakt7 fungsi_tipe_void_non_nilai_keluaran.c
#include<stdio.h>
void sekilas_info();
main()
{
printf("Panggil sekilas info...\n");
```

```
sesilas_info();
}
void sesilas_info()

printf(" ini adalah program percobaan\n");
printf("Membuat fungsi tanpa nilai keluaran bertipe void\n");
}
```

4. Pemakaian prototype fungsi

```
//pasti prototipe fungsi.c
#include<stdio.h>
float jumlah(float,float);
void(){

int a=8;
int b=3;
float c;
c=jumlah(a,b);
printf("c=a+b= %f\n",c);
printf("Hasil penjumlahan= %g\n",jumlah(20.1,3.9));
}
float jumlah(float x,float y)

return(x+y);
}
```

5. Pemakaian parameter formal dan aktual untuk menentukan bilangan input ganjil atau genap

```
/*pasti parameter_formal_aktual.c
menentukan apakah bilangan yang diinputkan ganjil atau genap*/
#include<stdio.h>
int ganjil(int);
main()
{
int a,hasil;
printf("Masukkan sembarang bilangan bulat: ");
scanf("%d",&a);
hasil=ganjil(a);
if(hasil==0)
printf("Bilangan tersebut tidak ganjil\n");
else if(hasil==1)
printf("Bilangan tersebut ganjil\n");
}
int ganjil(int x)

if (x%2 == 0)
return(0);
else
return(1);
}
```

E. PERCOBAAN

Compile dan Jalankan script pada tugas pendahuluan menggunakan Dev-C selanjutnya lengkapi dengan memberi komentar pada masing-masing baris script, buat pembahasan dan kesimpulan dari masing-masing contoh 1-5.

F. DATA HASIL PERCOBAAN

(print screen script serta hasil compile & running program disini)

Praktikum Bab 8

FUNGSI (lanjutan bab 7)

A. TUJUAN

1. Mahasiswa mampu menjelaskan cara pemanggilan fungsi
2. Mahasiswa mampu menjelaskan jenis variabel fungsi berdasarkan kelas penyimpanan
3. Mahasiswa mampu menjelaskan cara membuat beberapa fungsi dalam sebuah program

B. DASAR TEORI

1. Pemanggilan dengan Nilai dan Pemanggilan dengan Referensi

Pemanggilan dengan nilai (*call by value*) merupakan cara yang dipakai untuk seluruh fungsi buatan yang telah dibahas pada praktikum sebelumnya. Pada pemanggilan dengan nilai, nilai dari parameter aktual akan disalin ke parameter formal. Dengan cara ini nilai parameter aktual tidak bisa dirubah sekalipun nilai parameter formal berubah. Untuk lebih jelasnya lihat pada fungsi **tukar()** pada contoh berikut ini.

```
/* prakt8 tukar_data.c
Untuk melihat pengaruh pemanggilan nilai pada fungsi untuk
penukaran dua data */
#include <stdio.h>
void tukar (int, int);
main()
{
int a = 88, b = 77;
printf("Nilai sebelum pemanggilan fungsi\n");
printf("a = %d b = %d\n", a, b);
tukar(a,b);
printf("\nNilai setelah pemanggilan fungsi\n");
printf("a = %d b = %d\n", a, b);
}
void tukar(int x, int y)
{
int z;
z = x;
x = y;
y = z;
```

```
printf("\nNilai di akhir fungsi tukar():\n");
printf("x = %d y = %d\n", x, y);
}
```

Tampak bahwa setelah keluarnya dari pemanggilan fungsi **tukar()**, variabel **a** dan **b** (yang dilewatkan ke fungsi **tukar()** tidak berubah, walaupun pada fungsi **tukar()** telah terjadi penukaran antara parameter **x** dan **y** . Mengapa hal ini bisa terjadi? Sebab **x** hanyalah salinan dari **a** dan **y** adalah salinan dari **b**. Pada saat pemanggilan fungsi, maka :

- **x** bernilai 88 (nilai **a**)
- **y** bernilai 77 (nilai **b**)

Pemanggilan dengan referensi *(call by reference)* merupakan upaya untuk melewatkan alamat dari suatu variabel ke dalam fungsi. Cara ini dapat dipakai untuk mengubah isi suatu variabel di luar fungsi dengan pelaksanaan pengubahan dilakukan di dalam fungsi. Sebagai contoh perhatikan program **tukar_2data.c** yang merupakan modifikasi dari **tukar_data.c**. Perubahan yang pertama terletak dalam definisi fungsi, yang kini berupa

```
void tukar(int *px, int *py)
{
    int z;

    z = *px;
    *px = *py;
    *py = z;

    printf("\nNilai di akhir fungsi tukar():\n");
    printf("x = %d    y = %d\n", *px, *py);
}
```

Adapun perubahan dalam parameter aktualnya menjadi :

```
tukar(&a,&b);                    //alamat a dan alamat b
```

Dalam deklarasi parameter

```
int *px, int *py
```

menyatakan bahwa **px** dan **py** adalah suatu variabel pointer. Yang dimaksudkan sebagai variabel pointer adalah suatu variabel yang menunjuk ke variabel lain. Lebih jelasnya, variabel pointer berisi alamat dari variabel lain.

Adapun pada pemanggilan fungsi, **&a** dan **&b** masing-masing berarti "alamat a" dan "alamat b". Dengan pemanggilan seperti ini, hubungan antara variabel pointer **px** dan **py**

dengan variabel **a** dan **b** adalah seperti ditunjukkan pada gambar di bawah ini. Dalam hal ini, **px** dikatakan menunjuk variabel **a** dan **py** menunjuk variabel **b**.

```
/* prakt0 tukar_2data.c
Untuk melihat pengaruh pemanggilan nilai pada fungsi untuk
penukaran dua data */
#include <stdio.h>
void tukar (int *px, int *py); //prototype fungsi
main()
{
int a= 88,b = 77;
printf("Nilai sebelum pemanggilan fungsi\n");
printf("a = %d b = %d\n", a, b);
tukar(&a,&b); //alamat a dan alamat b
printf("\nNilai setelah pemanggilan fungsi\n");
printf("a = %d b = %d\n", a, b);
}
void tukar(int *px, int *py)
{
int z;
z = *px;
*px = *py;
*py = z;
printf("\nNilai di akhir fungsi tukar()\n");
printf("x = %d y = %d\n", *px, *py);
}
```

2. Jenis Variabel berdasarkan Kelas Penyimpanan

Suatu variabel, di samping dapat digolongkan berdasarkan jenis/tipe data juga dapat diklasifikasikan berdasarkan kelas penyimpanan *(storage class)*. Penggolongan berdasarkan kelas penyimpanan berupa :

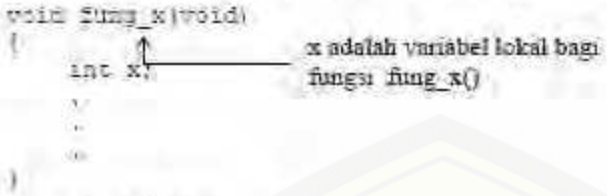
- variabel lokal
- variabel eksternal
- variabel statis
- variabel register

Variabel Lokal

Variabel lokal adalah variabel yang dideklarasikan dalam fungsi, dengan sifat :

- secara otomatis diciptakan ketika fungsi dipanggil dan akan sirna (lenyap) ketika eksekusi terhadap fungsi berakhir.
- Hanya dikenal oleh fungsi tempat variabel tersebut dideklarasikan
- Tidak ada inisialisasi secara otomatis (saat variabel diciptakan, nilainya tak menentu).

Dalam beberapa literatur, variabel lokal disebut juga dengan variabel otomatis. Variabel yang termasuk dalam golongan ini bisa dideklarasikan dengan menambahkan kata kuci *auto* di depan tipe-data variabel. Kata kunci ini bersifat opsional, biasanya disertakan sebagai penjelas saja. Contoh variabel lokal ditunjukkan di bawah



Pada **fung_x()**, deklarasi

```
int x;
```

dapat ditulis menjadi

```
auto int x;
```

Penerapan variabel lokal yaitu bila variabel hanya dipakai oleh suatu fungsi (tidak dimaksudkan untuk dipakai oleh fungsi yang lain). Pada contoh berikut, antara variabel **i** dalam fungsi **main()** dan **fung_1()** tidak ada kaitannya, sebab masing-masing merupakan variabel lokal.

Variabel Eksternal

Variabel eksternal merupakan variabel yang dideklarasikan di luar fungsi, dengan sifat :

- dapat diakses oleh semua fungsi
- kalau tak diberi nilai, secara otomatis diinisialisasi dengan nilai sama dengan nol.

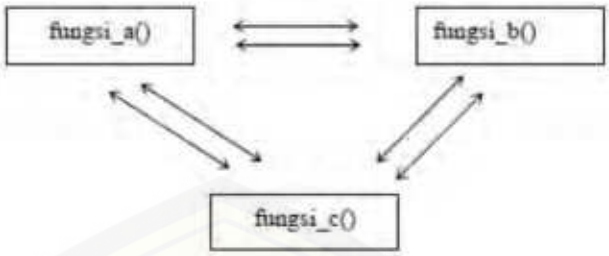
Variabel eksternal haruslah dideklarasikan sebelum definisi fungsi yang akan mempergunakannya. Untuk memperjelas bahwa suatu variabel dalam fungsi merupakan variabel eksternal, di dalam fungsi yang menggunakannya dapat mendeklarasikan variabel itu kembali dengan menambahkan kata kunci *extern* di depan tipe data variabel.

Kalau dalam suatu program terdapat suatu variabel eksternal, suatu fungsi bisa saja menggunakan nama variabel yang sama dengan variabel eksternal, namun diperlakukan sebagai variabel lokal.

3. Menciptakan Sejumlah Fungsi

Pada C, semua fungsi bersifat sederajat. Suatu fungsi tidak dapat didefinisikan di dalam fungsi yang lain. Akan tetapi suatu fungsi diperbolehkan memanggil fungsi yang lain, dan tidak tergantung kepada peletakan definisi fungsi pada program. Komunikasi antara fungsi dalam C ditunjukkan dalam gambar fungsi. Gambar fungsi tersebut menjelaskan kalau suatu fungsi katakanlah **fungsi_a()** memanggil **fungsi_b()**, maka bisa saja **fungsi_b()** memanggil **fungsi_a()**. Contoh program yang melibatkan fungsi yang memanggil fungsi yang lain ada pada program

kom_fung.c, yaitu **fungsi_1()** dipanggil dalam **main()**, sedangkan **fungsi_2()** dipanggil oleh **fungsi_1()**.



Gambar Fungsi

C. ALAT DAN KOMPONEN

- Komputer / Laptop
- Software program Dev C++

D. TUGAS PENDAHULUAN

1. Melewatkan parameter ke dan dari fungsi dengan cara pemanggilan dengan nilai (*call by value*)

```
//prakt8 pemanggilan_dengan_nilai.c
#include <stdio.h>
void tukar(int,int);
main()
{
int a,b;
a=88;
b=77;
printf("Nilai sebelum pemanggilan fungsi\n");
printf("a= %d b= %d\n\n", a,b);
tukar(a,b);
printf("Nilai sesudah pemanggilan fungsi\n");
printf("a= %d b=%d\n\n",a,b);
}
void tukar(int x, int y)
{
int z; //variabel sementara
z=x;
x=y;
y=z;
printf("Nilai diakhir fungsi tukar\n");
printf("x= %d y= %d\n\n",x,y);
}
```

2. Melewatkan parameter ke dan dari fungsi dengan cara pemanggilan dengan referensi (*call by reference*)

```
//prakt8 pemanggilan_dengan_referensi.c
```

```
#include <stdio.h>
void tukar(int*px,int*py);
main()
{
int a,b;
a=38;
b=7;
printf("Nilai sebelum memanggil fungsi\n");
printf("a= %d b= %d\n\n", a,b);
tukar(&a,&b);
printf("Nilai sesudah memanggil fungsi\n");
printf("a= %d b=%d\n\n",a,b);
}
void tukar(int*px, int*py)
{
int t; //variabel sementara
t=*px;
*px=*py;
*py=t;
printf("Nilai dikbir fungsi tukar\n");
printf("x= %d y= %d\n\n",*px,*py);
}
```

3. Jenis variabel berdasarkan kelas penyimpanan (*storage class*)

a. Variabel lokal

```
/* prakt3 variabel lokal.c*/
#include <stdio.h>
void fung_1(void);
main()
{
int i = 20;
fung_1();
printf("nilai i di dalam main() = %d\n", i);
}
void fung_1(void)
{
int i = 11;
printf("nilai i didalam fung_1() = %d\n", i);
}
```

b. Variabel eksternal

```
#include <stdio.h>
int i = 273;/* prakt3 variabel external.c*/
void tambah(void);
main()
{

printf("Nilai awal i = %d\n", i);
i = 7;
printf("Nilai i sini= %d\n", i);
tambah();
printf("Nilai i sini= %d\n", i);
tambah();
printf("Nilaii kini= %d\n", i);
}
void tambah(void)
{

i++;
```

```
}
```

c. Kombinasi variabel eksternal dan variabel lokal

```
/*prakt8 variabel eksternal lokal.c*/
#include <stdio.h>
int i = 275;/* variabel eksternal*/
void tambah(void);
main()
{
extern int i;/* variabel eksternal*/
printf("Nilai awal i = %d\n", i);
i --;
printf("Nilai i sesudah= %d\n", i);
tambah();
printf("Nilai i sekarang= %d\n", i);
tambah(); printf("Nilai i sekarang= %d\n", i);
}
void tambah(void)
{
int i; /* variabel lokal*/
i++;
}
```

4. Menciptakan sejumlah fungsi

```
//prakt8 beberapa_fungsi.c
#include <stdio.h>
void fungsi_1(void);
void fungsi_2(void);
main()
{
fungsi_1();
}
void fungsi_1(void)
{
printf(" fungsi 1 dipanggil\n");
fungsi_2();
}
void fungsi_2(void)
{
printf(" fungsi 2 dipanggil\n");
}
```

5. Fungsi dipakai secara rekursi (dapat memanggil dirinya sendiri), untuk mencari faktorial.

```
//prakt8 rekursi untuk mencari faktorial.c
#include<stdio.h>
int faktorial(int);
main()
{
int x;
printf("MENCARI FAKTORIAL DARI X\n");
printf("Masukkan nilai x : ");
scanf("%d",&x);
printf("Nilai faktorial dari x = %d adalah
%d\n",x,faktorial(x));
}
```

```
int faktorial(int m)
{
if(m==1)
return(1);
else return(m=m*faktorial(m-1));
}
```

E. PERCOBAAN

Compile dan Jalankan script pada tugas pendahuluan menggunakan Dev-C selanjutnya lengkapi dengan memberi komentar pada masing-masing baris script, buat pembahasan dan kesimpulan dari masing-masing contoh 1-5.

F. DATA HASIL PERCOBAAN

(print screen script serta hasil compile & running program disini)

Praktikum Bab 9
ARRAY

A. TUJUAN

1. Mahasiswa mampu menjelaskan tentang array berdimensi satu
2. Mahasiswa mampu menjelaskan tentang array berdimensi dua
3. Mahasiswa mampu menjelaskan tentang array berdimensi banyak
4. Mahasiswa mampu menjelaskan tentang inisialisasi array tak berukuran.
5. Mahasiswa mampu menjelaskan array sebagai parameter fungsi

B. DASAR TEORI

Array merupakan kumpulan dari nilai-nilai data yang bertipe sama dalam urutan tertentu yang menggunakan nama yang sama. Letak atau posisi dari elemen array ditunjukkan oleh suatu index. Dilihat dari dimensinya array dapat dibagi menjadi Array dimensi satu, array dimensi dua dan array multi-dimensi.

1. ARRAY DIMENSI SATU

- Setiap elemen array dapat diakses melalui indeks.
- Indeks array secara default dimulai dari 0.
- Deklarasi Array

```
tipe nama_var[ukuran];
```

dimana :

tipe = menyatakan jenis elemen array (ex. char, int)

ukuran = menyatakan jumlah maksimal elemen array

Contoh :

```
int Nilai [5];
```

Nilai[0]	Nilai[1]	Nilai[2]	Nilai[3]	Nilai[4]
70	80	82	60	75

2. ARRAY DIMENSI DUA

- Array dua dimensi merupakan array yang terdiri dari m buah baris dan n buah kolom.
- Bentuknya dapat berupa matriks atau tabel.

• Deklarasi array :

```
tipe_array nama_array[baris][kolom];
```

Contoh :

```
int X[3][4];
```

X[0][0]	X[0][1]	X[0][2]	X[0][3]
X[1][0]	X[1][1]	X[1][2]	X[1][3]
X[2][0]	X[2][1]	X[2][2]	X[2][3]

3. ARRAY DIMENSI BANYAK

• Bentuk umum deklarasi array dimensi banyak :

```
tipe_data       nama_var[uk_1][uk_2]..[uk_n];
```

dimana : uk_1, uk_2, uk_n adalah ukuran dari array.

Contoh deklarasi:

```
int nilai[4][2][7];
```

4. ARRAY TAK BERUKURAN

• Array dapat dideklarasikan tanpa memberi ukuran (jumlah data dalam array).

Dengan syarat:

- Harus langsung diinisialisasi.
- Hanya elemen pertama yang boleh tidak berukuran.

Contoh:

```
int nilai[] = {32, 45, 67,21};
int nilai[][2]={{9,7},{4,2},{8,3}};
```

5. MELEWATKAN ARRAY SEBAGAI PARAMETER

Untuk melewatkan array sebagai parameter ke suatu fungsi, untuk memanggilnya cukup dipanggil nama fungsinya.

Contoh :

Prototype fungsi :

```
void tambah (int data[]);
```

Pemanggilan fungsi, cukup dikirim nama array :

```
tambah (data);
```

C. ALAT DAN KOMPONEN

- Komputer / Laptop
- Software program Dev C++

D. TUGAS PENDAHULUAN

1. Mengisi dan menampilkan beberapa variabel array

```
 //prakt9 variabel_array.c
#include<stdio.h>
main()
{
int nilai[10];
int index;
printf("Masukkan nilai yang akan diinputkan \n");
for(index=0;index<10;index++)
scanf("%d",&nilai[index]);
printf("Data yang di-outputkan adalah \n");
for(index=0;index<10;index++)
printf("nilai[%d] = %d\n",index,nilai[index]);
}
```

2. Menginisialisasi elemen array bertipe integer

```
//prakt9 elemen_array_integer.c
#include <stdio.h>
main()
{
int i;
int nilai[10] = {0, 1, 4, 9, 16};
for(i=5; i<10; i++)
nilai[i] = i * i;
for(i=0; i<10; i++)
printf("nilai[%d] = %d\n", i, nilai[i]);
}
```

3. Menjumlahkan data-data yang terdapat dalam urutan array

```
//prakt9 jml_data_urutan_array.c
#include<stdio.h>
void tambah(int [], int);
main()
{
int a,dataku[5]={1,3,4,7,10};
for(a=0;a<5;a++)
printf("data ke - %d : %d\n",a,dataku[a]);
```

```
printf("Kirimkan nilai ini ke fungsi \n");
tambah(dataku,5);
}
void tambah(int dataku[],int juml_data)
{
int i,total=0;
for(i=0;i<juml_data;i++)
total=total+dataku[i];
printf("Hasil penjumlahan total adalah : %d\n",total);
}
```

4. Menentukan nilai maksimum dari sederetan nilai yang sudah diinisialisasi dan disimpan dalam array; mengirim array sebagai parameter sebuah fungsi.

```
//praktik3 array_sbg_parameter_fungsi.c
#include<stdio.h>
int maks(int [],int);
main()
{
int i;
int data1[]={5,34,56,-12,3,19};
int data2[]={1,-2,57,787,98,-13};
for(i=0;i<6;i++)
printf("data1[%d]= %d\n",i,data1[i]);
for(i=0;i<6;i++)
printf("data2[%d]= %d\n",i,data2[i]);
printf("nilai maksimum dari data1 adalah %d\n",
maks(data1,6));
printf("nilai maksimum dari data2 adalah %d\n",
maks(data2,6));
}
int maks(int nilai[],int jumlah_data)
{
int terbesar,i;
terbesar=nilai[0];
for(i=1;i<jumlah_data;i++)
if(nilai[i] > terbesar)
terbesar=nilai[i];
return terbesar;
}
```

5. Menjumlahkan data-data yang terdapat dalam dua buah array dan mengirim array sebagai parameter sebuah fungsi.

```
//praktik4 jml_data_array_sbg_parameter_fungsi.c
#include<stdio.h>
void hsl_jumlah(int dat1[2][3] , int dat2[2][3]);
main()
```

```
{
int grup1[2][3]={2,3,4,6,11,15};
int grup2[2][3]={3,4,1,2,-10,3};
printf("Pengiriman nilai ke fungsi \n");
hsl_jumlah(grup1,grup2);
}
void hsl_jumlah(int dat1[2][3], int dat2[2][3])
{
int i,j,jml[2][3];
for(i=0;i<2;i++)
for(j=0;j<3;j++)
{
jml[i][j]=dat1[i][j]+dat2[i][j];
printf("%5d",jml[i][j]);
}
printf("\n");
}
```

E. PERCOBAAN

Compile dan Jalankan script pada tugas pendahuluan menggunakan Dev-C selanjutnya lengkapi dengan memberi komentar pada masing-masing baris script, buat pembahasan dan kesimpulan dari masing-masing contoh 1-5.

F. DATA HASIL PERCOBAAN

(print screen script serta hasil compile & running program disini)

Praktikum Bab 10
STRING

A. TUJUAN

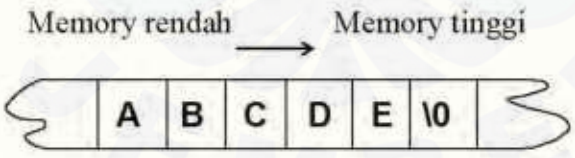
1. Mahasiswa mamapu menjelaskan tentang konsep string
2. Mahasiswa mampu menjelaskan operasi I/O pada string.
3. Mahasiswa mampu menjelaskan cara mengakses elemen string
4. Mahsiswa mampu menjelaskan berbagai fungsi mengenai string

B. DASAR TEORI

String merupakan bentuk data yang biasa dipakai dalam bahasa pemrograman untuk keperluan menampung dan memanipulasi data teks. Misalnya untuk menampung (menyimpan) suatu kalimat. Pada bahasa pemrograman C, string bukanlah sebagai tipe data tersendiri, melainkan hanya jenis khusus dari array. String ditulis dengan diawali dan diakhiri tanda petik ganda dan nilai string "selalu" diakhiri dengan tanda NULL (\0) Misal : "ABCDE"

1. Konstanta String

Kostanta "ABCDE" disimpan dalam memory secara berurutan, dengan komposisi sebagai berikut :



Setiap karakter akan menempati memory sebesar 1 byte. Byte terakhir secara otomatis akan berisi karakter NULL (\0).

2. Variabel String

Variabel string adalah variable yang dpakai untuk menyimpan string.

Misal :

```
char nama[15];
```

merupakan instruksi untuk mendeklarasikan variable string dengan panjang maksimal 15 karakter (termasuk karakter Null.

3. Input Data String dari Keyboard.

Untuk memasukkan data string ke dalam suatu variabel dapat dilakukan dengan 2 jenis perintah / fungsi:

a. gets(nama_array);

b. scanf("%s",&nama_array); Perhatikan :

- Nama_array adalah variable bertipe array_of_char
- Pada instruksi scanf(), di depan nama_array boleh diberi operator &, boleh tidak.
- Prototype gets() ada pada file **stdio.h**
- Gets() akan membaca seluruh karakter yang diketik sampai penekanan ENTER. Tidak ada pengecekan terhadap batasan dari array. Jika string yang dimasukkan melebihi ukuran array, sisa string berikutnya akan ditempatkan sesudah bagian terakhir dari array. Hal ini akan menimbulkan kejadian yang tidak diinginkan, seperti berubahnya isi variable.

4. Insialisasi String

Cara pengisian variable string dapat dilakukan dengan dua cara :

```
char kompiler c[]    {'T','U','R','B','O',' ','C','\0'}; atau
char kompiler_c[]="TURBO C";
```

5. Output Data String ke Layar

Untuk menampilkan isi variable string ke layar, dapat dilakukan dengan 2 jenis perintah / fungsi :

a. puts(var_string);

b. printf("%s", var_string);

Perhatikan :

- **var_string** adalah berupa *array of char*
- *puts()* secara otomatis menambahkan karakter '\n' di akhir string

6. Fungsi-fungsi String

Merupakan kumpulan fungsi-fungsi pustaka string yang prototype-nya berada pada file **string.h**

a. Fungsi *strcpy()* untuk menyalin nilai string Bentuk penulisan :

```
#include<string.h>
strcpy(tujuan, asal);
```

b. Fungsi *strlen()* untuk mengetahui panjang nilai string

Bentuk penulisan :

```
#include<string.h>;
strlen(var_string);
```

c. Fungsi *strcat()* untuk menggabungkan nilai string

Bentuk penulisan :

```
#include<string.h>;
strcat(tujuan, sumber);
```

d. Fungsi *strcmp()* untuk membandingkan dua nilai string

Bentuk penulisan :

```
#include<string.h>;
strcmp(str1, str2);
```

e. Fungsi *strchr()* untuk mencari nilai karakter dalam string Bentuk penulisan :

```
#include<string.h>;
strchr(var_string, kar);
```

C. ALAT DAN KOMPONEN

- Komputer / Laptop
- Software program Dev C++

D. TUGAS PENDAHULUAN

1. Memasukkan data string dari keyboard

```
//prakt10 string_keyboard.c
#include <stdio.h>
#define MAKS 15
main()
{
char nama[MAKS];
printf("Masukkan nama Anda : ");
gets(nama);
printf("\nHalo,    %s.    Selamat
belajar string.\n", nama);
}
```

2. *Menghitung banyaknya karakter dari suatu string yang dimasukkan melalui keyboard.*

```
/* File program :prakt10 karakter.c */
#include <stdio.h>
#define MAKS 256
main()
{
int i, jumkar = 0;
char teks[MAKS];
puts("Masukkan suatu kalimat (maks
256 karakter).");
//masukan dari keyboard
gets(teks);
for(i=0; teks[i]; i++)
jumkar++;
printf("\nJumlah karakter = %d\n",
jumkar);
}
```

3. *Menyalin sebuah string masukan.*

```
/* File program : prakt10 salinan.c */
#include<stdio.h>
#define MAKS 30
main()
{
int i;
char asal[] = "Sedang belajar bahasa C";
char salinan[MAKS];
i=0;
while(asal[i] != '\0')
{
salinan[i] asal[i];
i++;
}
salinan[i]='\0';
printf("Isi salinan adalah : %s\n",salinan);
}
```

4. *Menyalin isi string2 ke string1 menggunakan fungsi* ***strcpy()****.*

```
/* File program :prakt10 salinan2.c */
#include<stdio.h>
#include<string.h>
main()
{
char alt1[50];
char alt2[] "ABCDE";
strcpy(alt1,alt2);
printf("Isi string 1 adalah : %s\n",alt1);
```

```
printf("Isi string 2 adalah : %s\n",str2);
}
```

5. *Menghitung jumlah karakter dari suatu string masukan menggunakan fungsi* ***strlen()***.

```
/*prakt10 jml_karakter_strlen */
#include <stdio.h>
#include <string.h>
#define MAKS 256
main()
{
char kal[MAKS];
printf("Masukkan kalimat yang akan dihitung panjangnya :\n");
gets(kal);
printf("\nPanjang string pada kalimat = %d karakter\n",strlen(kal));
}
```

6. *Menggabungkan isi string1 dengan string2 menggunakan fungsi* ***strcat()***.

```
/*prakt10 gabung_string_strcat.c*/
#include<stdio.h>
#include<string.h>
#define PJG 15
main()
{
char str1[PJG], str2[PJG];
strcpy(str1,"sala"); //str1 diisi "sala"
strcpy(str2,"tiga"); //str2 diisi "tiga"
strcat(str1,str2); //tambahkan isi str2 di akhir str1
printf("str1 --> %s str2 --> %s\n",str1,str2);
}
```

7. *Membandingkan isi string1 dengan string2 menggunakan fungsi* ***strcmp()***.

```
/*prakt10 bandingkan string strcmp.c*/
#include<stdio.h>
#include<string.h>
main()
{
char str1[]="ADode";
char str2[]="ADODE";
int hasil;
hasil=strcmp(str1,str2);
if(hasil==0)
printf("String 1 sama dengan String 2\n");
else if(hasil < 0)
printf("String 1 lebih kecil dari String 2\n");
else
```

```
printf("String 1 lebih besar dari String 2\n");
}
```

E. PERCOBAAN

Compile dan Jalankan script pada tugas pendahuluan menggunakan Dev-C selanjutnya lengkapi dengan memberi komentar pada masing-masing baris script, buat pembahasan dan kesimpulan dari masing-masing contoh 1-7.

F. DATA HASIL PERCOBAAN

(print screen script serta hasil compile & running program disini)

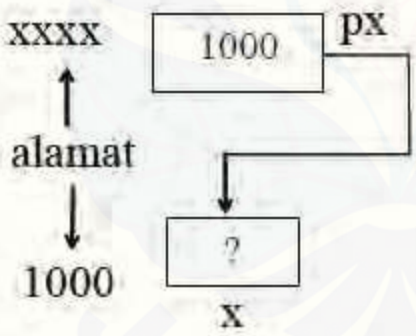
Praktikum Bab 11
POINTER

A. TUJUAN

1. Mahasiswa mampu menjelaskan tentang konsep dari variabel pointer
2. Mahasiswa mampu menjelaskan tentang pointer array
3. Mahasiswa mampu menjelaskan tentang pointer string
4. Mahasiswa mampu menjelaskan tentang array pointer
5. Mahasiswa mampu menjelaskan tentang pointer dalam fungsi

B. DASAR TEORI

Variabel pointer sering dikatakan sebagai vaariabel yang menunjuk obyek lain. Pada kenyataan, variabel pointer (atau disingkat pointer) berisi alamat dari obyek lain (yaitu obyek yang dikatakan ditunjuk oleh pointer). Perhatikan ilistrasi di bawah :



Pointer px menunjuk alamat variable x. Jika alamat x adalah 1000, maka pointer px bernilai 1000

1. Deklarasi Variabel Pointer

Suatu variable pointer dideklarasikan sebagai berikut

```
tipe_data *nama_variabel;
```

Nama variable pointer

tipe_data yang ditunjuk oleh pointer

Agar suatu pointer menunjuk ke variable lain, mula-mula pointer harus diisi dengan Alamat dari variabel yang akan ditunjuk. Untuk menyatakan alamat dari suatu variabel, digunakan operator & di depan nama variabel. Contoh, jika x adalah variabel bertipe int, maka pointer px yang berisi alamat dari variabel x ditulis sebagai :

```
px = &x;
```

Untuk mengakses isi dari suatu variabel melaui pointer, perlu dinyatakan dulu penunjukan alamat dari variabel tersebut. Dua pernyataan di bawah ini menunjukkan bahwa nilai y adalah sama dengan nilai x.

```
px = &x;    ← pointer px berisi alamat dari variabel x
y = *px;    ← variabel y berisi nilai yang ditunjuk oleh pointer px.
```

Antara tipe pointer (sesuai dengan pendeklarasian pointer) dan tipe obyek yang akan ditunjuk oleh pointer haruslah sejenis. Jika tidak sejenis akan menimbulkan kesalahan. Contoh :

```
int x;      ← x adalah obyek
int *px;    ← px adalah pointer yang menunjuk obyek x
```

2. Pointer dan Array

Pointer dan array dalam C mempunyai hubungan yang sangat erat. Sebab sesungguhnya array secara internal akan diterjemahkan ke dalam bentuk pointer.
Perhatikan:

```
static int tgl_lahir[3] = {16,10,1993};   ← array tgl_lahir dengan panjang 3
```

dan: `int *ptgl;` ← pointer ptgl

jika dilanjutkan dengan instruksi : `ptgl = &tgl_lahir[0];`

maka pointer **ptgl** berisi alamat dari array **tgl_lahir** yang ke-0 (=alamat pertama dari array tgl_lahir).

Pernyataan di atas bisa ditulis : `ptgl = tgl_lahir` ← yang otomatis akan menyatakan alamat awal dari array tgl_lahir

Kalau ingin menampilkan seluruh elemen array tgl_lahir, instruksi yang biasa dilakukan berupa :

```
for(i=0;i<3;i++)
  printf("%d",tgl_lahir[i]);
```

Instruksi di atas, jika menggunakan pointer dapat dirubah menjadi :

```
ptgl=tgl_lahir;
for(i=0;i<3;i++)
printf("%d",*(ptgl+i));
```

Dimana : `tgl_lahir[i]` dapat diganti dengan variable pointer `*(ptgl+i)`

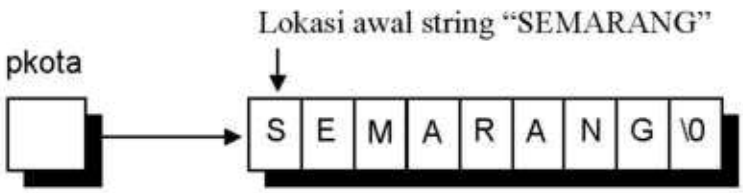
3. Pointer dan String

Seperti halnya array, hubungan pointer dengan string juga dekat. Jika sebuah instruksi dinyatakan sebagai berikut :

```
char  *pkota = "SEMARANG";
```

Instruksi di atas diartikan sebagai :

- Kompiler akan mengalokasikan variabel **pkota** sebagai variabel pointer yang akan menunjuk ke obyek bertipe **char** dan menempatkan konstanta "SEMARANG" dalam suatu memori.
- Kemudian pointer pkota akan menujuk ke lokasi string "SEMARANG"

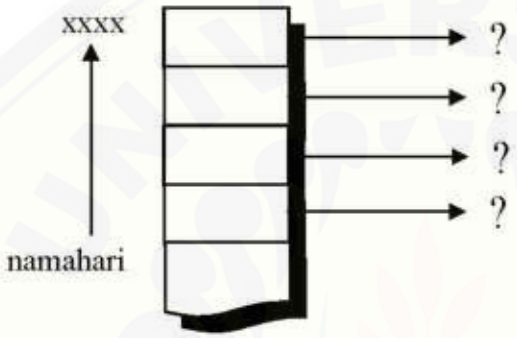


4. Array dari Pointer

Suatu array bisa digunakan untuk menyimpan sejumlah pointer. Sebagai contoh :

```
char *namahari[10];
```

Instruksi di atas menyatakan deklarasi array pointer. Array **namahari** terdiri dari 10 elemen berupa pointer yang menunjuk ke data bertipe **char**.



Inisialisasi array pointer dilakukan saat pendeklarasian. Sebagai contoh :

```
static char *namahari[] =
{"Senin","Selasa","Rabu","Kamis","Jumat","Sabtu","Minggu"};
```

Yang artinya :

`namahari[0]` menunjuk ke string "Senin"
`namahari[1]` menunjuk ke string "Selasa" dsb.

5. Pointer dan Fungsi

5.1. Pointer Sebagai Parameter Fungsi

Penerapan pointer sebagai parameter yaitu jika diinginkan agar nilai suatu variabel internal dapat diubah oleh fungsi yang dipanggil. Sebagai contoh :

```
void naikkan_nilai(int *x, int *y)
{
    *x = *x+2;
    *y= *y+2;
}
```

Fungsi di atas dimaksudkan agar jika dipanggil, variabel yang berkaitan dengan parameter aktual dapat diubah nilainya. Pemanggilannya dapat berupa :

```
naikkan_nilai(&a, &b);
```

Variabel a dan b harus diawali dengan operator & yang menyatakan sebagai alamat variabel, karena parameter fungsinya berupa pointer.

5.2. Pointer Sebagai Keluaran Fungsi

Suatu fungsi dapat dibuat agar keluarannya berupa pointer. Misalkan sebuah fungsi yang menghasilkan keluaran berupa pointer yang menunjuk ke string nama buah, seperti berikut:

```
char *nama_buah(int n)
{
    static char *buah[] =
  {"Kode buah salah", "Jambu","Tomat", "Durian",
   "Semangka","Jeruk"};
   return ((n<1 || n>5) ? buah[0] : buah[n]);
}
```

Pada bagian akhir dari fungsi diatas menyatakan, jika masukan fungsi n<1 atau n>5 maka akan dikeluarkan string "Kode buah salah", jika 1<n<5, maka string nama buah yang lain yang dikelaurkan.

C. ALAT DAN KOMPONEN

- Komputer / Laptop
- Software program Dev C++

D. TUGAS PENDAHULUAN

1. Mengakses isi suatu variabel melalui pointer.

```
/*prakt11 akses isi variable_pointer.c*/
#include <stdio.h>
main()
{
int y, x = 87;
int *px;
px = &x;
y = *px; printf("Alamat x = %p\n", &x);
printf("Isi px = %p\n", px);
printf("Isi x = %d\n", x);
printf("Nilai yang ditunjuk oleh px = %d\n", *px);
printf("Nilai y = %d\n", y);
}
```

2. Mengubah isi suatu variabel melalui pointer.

```
/*prakt11 ubah_variabel_pointer.c*/
#include <stdio.h>
main()
{
float d, *pd;
d=54.5;
printf("Isi d mula-mula = %g\n", d);
pd = &d;
*pd = *pd+10;
printf("Isi d sekarang = %g\n", d);
}
```

3. *Pointer dan array. Suatu nama array yang ditulis tanpa indeks menunjukkan alamat elemen pertama dari array.*

```
/*praktl1 alamat_elemen_array.c*/
#include<stdio.h>
main()
{
static int tgl_lahir[]={17,8,1945};
int *ptgl;
ptgl=tgl_lahir; /*berisi alamat array */
printf("Nilai yang ditunjuk oleh ptgl = %d\n",*ptgl);
printf("Nilai dari tgl lahir[0] = %d\n",tgl_lahir[0]);
}
```

4. *Pointer dan array. Menampilkan seluruh elemen array.*

```
/*praktl1 pointer_array_elemen_array.c*/
 #include<stdio.h>
 main()
 {
 static int tgl_lahir[] ={16,10,1993};
 int *ptgl,i;
 ptgl =tgl_lahir;
 for(i=0;i<2;i++)
 printf("isi array ke- %d   %d\n",i,*(ptgl+i));
 }
```

5. *Pointer dan String. Menukarkan isi 2 string dengan fasilitas pointer.*

```
/*praktl1 tukar_string_hasil_tes_pointer.c*/
#include<stdio.h>
main()
{
char *nama1="HERCULES";
char *nama2="SUPERMAN";
char *namax;
puts("MULA_MULA");
printf("nama1 --> %s\n",nama1);
printf("nama2 --> %s\n",nama2);
namax=nama1;
nama1=nama2;
nama2=namax;
puts("SEKARANG");
printf("nama1 --> %s\n",nama1);
printf("nama2 --> %s\n",nama2);
}
```

6. *Pointer sebagai Parameter Fungsi. Menaikkan nilai suatu variabel*

```
/*praktl1 pointer parameter fungsi.c*/
#include<stdio.h>
```

```
void naikkan_nilai(int *x, int *y);
main()
{
int a=3, b=7;
printf("Awal a= %d, b= %d\n",a,b);
naikkan_nilai(&a,&b);
printf("Akhir a= %d, b= %d\n",a,b);
}
void naikkan_nilai(int *x, int *y)
{
*x = *x-2;
*y   *y  2;
}
```

7. *Pointer sebagai Keluaran Fungsi.*

```
/*prakt11 pointer_keluaran_fungsi.c*/
#include<stdio.h>
char *nama_bulan(int n);
main()
{
int bln;
printf("Bulan ke(1..12) : ");
scanf("%d",&bln);
printf("%s\n",nama_bulan(bln));
}
char *nama_bulan(int n)
{
char *bulan[]=
{
"Kode bulan salah",
"Januari",
"Februari",
"Maret",
"April",
"Mei",
"Juni",
"Juli",
"Agustus",
"September",
"Oktober",
"Nopember",
"Desember"
};
return((n<1 | n>12) ? bulan[0] : bulan[n]);
}
```

E. PERCOBAAN

Compile dan Jalankan script pada tugas pendahuluan menggunakan Dev-C selanjutnya lengkapi dengan memberi komentar pada masing-masing baris script, buat pembahasan dan kesimpulan dari masing-masing contoh 1-7.

F. DATA HASIL PERCOBAAN

(print screen script serta hasil compile & running program disini)

Daftar pustaka

http://kangedi.lecturer.pens.ac.id/materi%20kuliah/pemrograman/Prakt%201%20Pengantar%20Konsep%20Pemrograman.pdf [online, diakses 2015]

Kadir, Abdul. 2012. *Algoritma & Pemrograman menggunakan C&C++*. Yogyakarta:PENERBIT ANDI.