Mengenal Subroutine pada Pemrograman "C"

Dian Wirdasari

Abstrak

Subroutine merupakan suatu bagian dari program yang dimaksudkan untuk mengerjakan suatu tugas tertentu dan letaknya terpisah dari program yang memanggilnya. Dalam bahasa C sebuah subroutine adalah sebuah function. Function atau fungsi Fungsi merupakan elemen utama dalam bahasa C karena bahasa C sendiri terbentuk dari kumpulan fungsi-fungsi. Dalam setiap program bahasa C, minimal terdapat satu fungsi yaitu fungsi main(). Dalam bahasa C fungsi dapat dibagi menjadi dua, yaitu fungsi pustaka atau fungsi yang telah tersedia dalam Turbo C dan fungsi yang didefinisikan atau dibuat oleh programmer.

Kata Kunci: function, fungsi, program, bahasa C

A. PENDAHULUAN

Sebuah subroutine dalam bahasa C adalah merupakan sebuah function atau fungsi. Fungsi merupakan suatu bagian dari program yang dimaksudkan untuk mengerjakan suatu tugas tertentu dan letaknya terpisah dari program yang memanggilnya. Fungsi merupakan elemen utama dalam bahasa C karena bahasa C sendiri terbentuk dari kumpulan fungsi-fungsi. Dalam setiap program bahasa C, minimal terdapat satu fungsi yaitu fungsi main().

Fungsi banyak diterapkan dalam program-program C yang terstruktur. Keuntungan penggunaan fungsi dalam program yaitu program akan memiliki struktur yang jelas (mempunyai readability yang tinggi) dan juga akan menghindari penulisan bagian program yang sama.

Dalam bahasa C fungsi dapat dibagi menjadi dua, yaitu fungsi pustaka atau fungsi yang telah tersedia dalam Turbo C dan fungsi yang didefinisikan atau dibuat oleh programmer.

B. FUNGSI MATEMATIKA

Fungsi-fungsi pustaka untuk operasi matematika tersimpan dalam header file **math.h** dan **stdlib.h**, fungsi-fungsi sqrt, pow, sin, cos, tan, atof, atoi pakai **math.h**. sedangkan fungsi-fungsi **max,min, div** pakai **stdlib.h.**

1. **sqrt()**
   - Digunakan untuk menghitung akar dari sebuah bilangan.
   - Bentuk umum : **sqrt(bilangan);**
2. **pow()**
   - Digunakan untuk menghitung pemangkatan suatu bilangan.
   - Bentuk umum : **pow(bilangan, pangkat);**

Contoh 1

```
#include "stdio.h"
#include "conio.h"
#include "math.h"
void main()
```

```
{
int x, y;
float z;
clrscr();
printf("Menghitung x pangkat y\n");
printf("x = "); scanf("%i", &x);
printf("y = "); scanf("%i", &y);
printf("%i dipangkatkan dengan %i = %7.2lf", x, y, pow(x, y));
getch();
clrscr();
printf("Menghitung akar suatu bilangan z\n");
printf("z = "); scanf("%f", &z);
printf("Akar dari %f adalah %7.2lf", z, sqrt(z));
getch();
}
```

Jika program Contoh 1 dijalankan, contoh hasilnya adalah sebagai berikut:

```
Menghitung x pangkat y
x = 5
y = 3
5 dipangkatkan dengan 3 =  125.00

Menghitung akar suatu bilangan z
z = 75
Akar dari 75.000000 adalah    8.66
```

3. sin(), cos(), tan()

- Masing-masing digunakan untuk menghitung nilai sinus, cosinus dan tangens dari suatu sudut.
- Bentuk umum :
  - **sin(sudut);**
  - **cos(sudut);**
  - **tan(sudut);**

Contoh 2

```
#include "stdio.h"
#include "conio.h"
#include "math.h"
void main()
{
float sudut;
clrscr();
printf("Menghitung nilai sinus, cosinus dan tangens\n");
printf("Masukkan sudut : "); scanf("%f", &sudut);
printf("Nilai sinus %.2f derajat = %.3f\n", sudut, sin(sudut));
printf("Nilai cosinus %.2f derajat = %.3f\n", sudut, cos(sudut));
printf("Nilai tangens %.2f derajat = %.3f\n", sudut, tan(sudut));
getch();
}
```

Jika program Contoh 2 dijalankan, contoh hasilnya adalah sebagai berikut:

```
Menghitung nilai sinus, cosinus dan tangens
Masukkan sudut : 60
Nilai sinus 60.00 derajat = -0.305
Nilai cosinus 60.00 derajat = -0.952
Nilai tangens 60.00 derajat = 0.320
```

4. atof()

- Digunakan untuk mengkonversi nilai string menjadi bilangan bertipe double.
- Bentuk umum : **atof(char x);**

5. atoi()

- Digunakan untuk mengkonversi nilai string menjadi bilangan bertipe integer.
- Bentuk umum : **atoi(char x);**

Contoh 3

```
#include "stdio.h"
#include "conio.h"
#include "math.h"
void main()
{ char x[4] = "100", y[5] ="10.3";
int a;
float b;
clrscr();
a = atoi(x); b = atof(y);
printf("Semula A = %s B = %s\n", x,y);
printf("Setelah dikonversi A = %i B = %.2f", a,b);
getch();
}
```

Jika program Contoh 3 dijalankan, hasilnya adalah sebagai berikut:

```
Semula A = 100 B = 10.3
Setelah dikonversi A = 100 B = 10.30
```

6. div()

- Digunakan untuk menghitung hasil pembagian dan sisa pembagian.
- Bentuk umum : **div_t div(int x, int y)**
- Strukturnya :

```
typedef struct
{ int qout; /*hasil pembagian*/
int rem /* sisa pembagian */
} div_t;
```

Contoh 4

```
#include "stdio.h"
#include "conio.h"
#include "stdlib.h"
```

```
void main()
{
int x, y;
div_t hasil;
clrscr();
printf("Menghitung sisa dan hasil pembagian x dengan y\n");
printf("x = "); scanf("%i", &x);
printf("y = "); scanf("%i", &y);
hasil = div(x,y);
printf("\n\n");
printf("%3i div %3i = %3i sisa %3i", x, y,hasil.quot,hasil.rem);
getch();
}
```

Jika program Contoh 4 dijalankan, contoh hasilnya adalah sebagai berikut:

```
Menghitung sisa dan hasil pembagian x dengan y
x = 9
y = 4

 9 div   4 =   2 sisa   1
```

7. max()

- Digunakan untuk menentukan nilai maksimal dari dua buah bilangan.
- Bentuk umum : **max(bilangan1, bilangan2);**

8. min()

- Digunakan untuk menentukan bilangan terkecil dari dua buah bilangan.
- Bentuk umum : **min(bilangan1, bilangan2);**

Contoh 5

```
#include "stdio.h"
#include "conio.h"
#include "stdlib.h"
void main()
{
int x, y, z;
clrscr();
printf("Menentukan bilangan terbesar dan terkecil\n");
printf("X = "); scanf("%i", &x);
printf("Y = "); scanf("%i", &y);
printf("Z = "); scanf("%i", &z);
printf("\nBilangan terbesar : %i", max(max(x, y), z));
printf("\nBilangan terkecil : %i", min(min(x, y), z));
getch();
}
```

Jika program Contoh 5 dijalankan, contoh hasilnya adalah sebagai berikut:

```
Menentukan bilangan terbesar dan terkecil
X = 20
Y = 9
Z = -4

Bilangan terbesar : 20
Bilangan terkecil : -4
```

C. MEMBUAT FUNGSI SENDIRI

1. Deklarasi Fungsi

Sebelum digunakan (dipanggil), suatu fungsi harus dideklarasikan dan didefinisikan terlebih dahulu. Bentuk umum pendeklarasian fungsi adalah :

tipe_fungsi nama_fungsi(parameter_fungsi);

Sedangkan bentuk umum pendefinisian fungsi adalah :

```
tipe_fungsi nama_fungsi(parameter_fungsi)
{ statement
  statement
  ..........
}
```

2. Hal-hal yang perlu diperhatikan

Hal-hal yang perlu diperhatikan dalam penggunaan fungsi adalah:

1. Kalau tipe fungsi tidak disebutkan, maka akan dianggap sebagai fungsi dengan nilai keluaran bertipe integer.
2. Untuk fungsi yang memiliki keluaran bertipe bukan integer, maka diperlukan pendefinisian penentu tipe fungsi.
3. Untuk fungsi yang tidak mempunyai nilai keluaran maka dimasukkan ke dalam tipe void
4. Pernyataan yang diberikan untuk memberikan nilai akhir fungsi berupa pernyataan return.
5. Suatu fungsi dapat menghasilkan nilai balik bagi fungsi pemanggilnya.

Contoh 6

```
#include<stdio.h>
#include<conio.h>
void info_program(); /*Prototype Fungsi*/
main()
{
   clrscr();
   printf("\nInfo Pembuat Program \n");
   info_program();
   getch();
   info_program();
}
void info_program() /*Definisi Fungsi*/
{
   printf("Designed Program by \n");
```

```
   printf("Lab. Kom. Digital \n");
   printf("STMIK Triguna Dharma \n");
}
```

Jika program Contoh 6 dijalankan, hasilnya adalah sebagai berikut:

```
Info Pembuat Program
Designed Program by
Lab. Kom. Digital
STMIK Triguna Dharma
```

3. Prototype Fungsi

Digunakan untuk menjelaskan kepada kompiler mengenai :

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

Contoh 7

```
/* Program penjumlahan dua bilangan dengan fungsi */
#include "stdio.h"
#include "conio.h"
float tambah(float x, float y); /* prototype fungsi tambah() */
void main(){    /*fungsi utama main */
float a, b, c;
clrscr();
printf("Program penjumlahan dua bilangan dengan fungsi\n");
printf("A = "); scanf("%f", &a);
printf("B = "); scanf("%f", &b);
c = tambah(a, b); /* pemanggilan fungsi tambah() */
printf("A + B = %.2f", c);
getch();
}
float tambah(float x, float y) /* Definisi fungsi , tanpa titik koma */
{
return (x+y); /* Nilai balik fungsi */
}
```

Jika program Contoh 7 dijalankan, contoh hasilnya adalah sebagai berikut:

```
Program penjumlahan dua bilangan dengan fungsi
A = 80
B = 50
A + B = 130.00
```

Contoh 8

```
/* Program menghitung nilai faktorial */
#include<stdio.h>
#include<conio.h>
long int Faktorial(int N);
main() {
  int N;  long int Fak;
  clrscr();
  printf("Berapa Faktorial ? "); scanf("%d",&N);
  Fak = Faktorial(N);
  printf ("%d faktorial = %ld \n", N, Fak);
  getch();
  return(0);
}
long int Faktorial(int K) {
 int I;
 long int F =1;
 if (K<=0) return(0);
 for(I=2;I<=K;I++)
   {
    F= F *I; /* sama dengan F *= I*/
   }
 return(F);
 }
```

Jika program Contoh 8 dijalankan, contoh hasilnya adalah sebagai berikut:

```
Berapa Faktorial ? 6
6 faktorial = 720
```

4. Parameter Formal dan Parameter Aktual

1. **Parameter Formal** adalah variabel yang ada pada daftar parameter dalam definisi fungsi.
2. **Parameter Aktual** adalah variabel (parameter) yang dipakai dalam pemanggilan fungsi.

Dalam contoh program pertambahan (Contoh 7) di atas parameter formal terdapat pada pendefinisian fungsi :

```
float tambah(float x, float y) //parameter formal
{ return (a+b);
}
```

Sedangkan parameter aktual terdapat pada pemanggilan fungsi :

```
void main(){
........
c = tambah(a, b); //parameter aktual
........
}
```

5. Cara Melewatkan Parameter

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

1. Pemanggilan Secara Nilai (*Call by Value*)
   - Call by value akan menyalin nilai dari parameter aktual ke parameter formal.

- Yang dikirimkan ke fungsi adalah nilai dari datanya, bukan alamat memori letak dari datanya.
- Fungsi yang menerima kiriman nilai akan menyimpannya di alamat terpisah dari nilai aslinya yang digunakan oleh bagian program yang memanggil fungsi.
- Perubahan nilai di fungsi (parameter formal) tidak akan merubah nilai asli di bagian program yang memanggilnya.
- Pengiriman parameter secara nilai adalah pengiriman searah, yaitu dari bagian program yang memanggil fungsi ke fungsi yang dipanggil.
- Pengiriman suatu nilai dapat dilakukan untuk suatu ungkapan, tidak hanya untuk sebuah variabel, elemen array atau konstanta saja.

Contoh 9

```
/*Contoh fungsi by value*/
#include <stdio.h>
void fungsi_nilai (int ); /*Pendeklarasian Fungsi*/
main()
{
int a;
a = 10;
printf("nilai a sebelum fungsi = %d\n", a);
fungsi_nilai (a); /*Pemanggilan Fungsi*/
printf("nilai a setelah fungsi = %d\n", a);
}
void fungsi_nilai (int b) /*Pendefinisian Fungsi*/
{
b = b + 5;
printf ("nilai a di fungsi = %d\n",b);
}
```

Jika program contoh 9 dijalankan, hasil tampilannya di layar adalah sebagai berikut:

```
nilai a sebelum fungsi = 10
nilai a di fungsi = 15
nilai a setelah fungsi = 10
```

Contoh 10

```
#include "stdio.h"
#include "conio.h"
void tukar(int x, int y); /* pendeklarasian fungsi */
void main(){
int a,b;
clrscr();
a = 15;
b = 10;
printf("Nilai sebelum pemanggilan fungsi\n");
printf("a = %i b = %i\n\n", a, b); /* a dan b sebelum pemanggilan
fungsi*/
tukar(a,b); /* pemanggilan fungsi tukar() */
printf("Nilai setelah pemanggilan fungsi\n");
```

```
printf("a = %i b = %i\n\n", a, b); /* a dan b setelah pemanggilan
fungsi*/
getch();
}
void tukar(int x, int y) /* Pendefinisian fungsi tukar() */
{ int z; /* variabel sementara */
z = x;
x = y;
y = z;
printf("Nilai di akhir fungsi tukar()\n");
printf("x = %i y = %i\n\n", x, y);
}
```

Jika program contoh 10 dijalankan, hasil tampilannya di layar adalah sebagai berikut:

```
Nilai sebelum pemanggilan fungsi
a = 15 b = 10

Nilai di akhir fungsi tukar()
x = 10 y = 15

Nilai setelah pemanggilan fungsi
a = 15 b = 10
```

2. Pemanggilan Secara Referensi (*Call by Reference*)
   - Pemanggilan secara Referensi merupakan upaya untuk melewatkan alamat dari suatu variabel ke dalam fungsi.
   - Yang dikirimkan ke fungsi adalah alamat letak dari nilai datanya, bukan nilai datanya.
   - Fungsi yang menerima kiriman alamat ini makan menggunakan alamat yang sama untuk mendapatkan nilai datanya.
   - Perubahan nilai di fungsi akan merubah nilai asli di bagian program yang memanggil fungsi.
   - Pengiriman parameter secara referensi adalah pengiriman dua arah, yaitu dari fungsi pemanggil ke fungsi yang dipanggil dan juga sebaliknya.
   - Pengiriman secara acuan tidak dapat bdilakukan untuk suatu ungkapan.

Contoh 11

```
#include <stdio.h>
void fungsi_nilai (int *b );
main()
{
int a;
a = 10;
printf("nilai a sebelum fungsi = %d\n", a);
fungsi_nilai (&a);
printf("nilai a setelah fungsi = %d\n", a);
}
void fungsi_nilai (int *b)
{
```

```
*b = *b + 5;
printf ("nilai a di fungsi = %d\n",*b);
}
```

Jika program contoh 11 dijalankan, hasil tampilannya di layar adalah sebagai berikut:

```
nilai a sebelum fungsi = 10
nilai a di fungsi = 15
nilai a setelah fungsi = 15
```

Contoh 12

```
#include "stdio.h"
#include "conio.h"
void tukar(int *px, int *py);
void main()
{
int a,b;
clrscr();
a = 15;
b = 10;
printf("Nilai sebelum pemanggilan fungsi\n");
printf("a = %i b = %i\n\n", a, b);
tukar(&a,&b); /* parameter alamat a dan alamat b */
printf("Nilai setelah pemanggilan fungsi\n");
printf("a = %i b = %i\n\n", a, b);
getch();
}
void tukar(int *px, int *py)
{ int z; /* variabel sementara */
z = *px;
*px = *py;
*py = z;
printf("Nilai di akhir fungsi tukar()\n");
printf("*px = %i *py = %i\n\n", *px, *py);
}
```

Jika program contoh 12 dijalankan, tampilannya adalah sebagai berikut:

```
Nilai sebelum pemanggilan fungsi
a = 15 b = 10

Nilai di akhir fungsi tukar()
*px = 10 *py = 15

Nilai setelah pemanggilan fungsi
a = 10 b = 15
```

6. Penggolongan Variabel berdasarkan Kelas Penyimpanan (Storage Class)

1. Variabel lokal

   Variabel lokal adalah variabel yang dideklarasikan di dalam fungsi.

   Sifat-sifat variabel lokal :

   - Secara otomatis akan diciptakan ketika fungsi dipanggil dan akan lenyap ketika proses eksekusi terhadap fungsi berakhir.
   - Hanya dikenal oleh fungsi tempat variabel dideklarasikan.
   - Tidak ada inisialisasi secara otomatis (saat variabel diciptakan nilainya random).
   - Dideklarasikan dengan menambahkan kata "**auto**" (opsional).

Contoh 13

```
/*Contoh Program Penggunaan Variabel Lokal*/
#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; /*variabel lokal*/
      printf("nilai i di dalam fung_1() = %d\n", i);
}
```

Jika program contoh 13 dijalankan, hasil tampilannya adalah sebagai berikut:

```
nilai i di dalam fung_1() = 11
nilai i di dalam main() = 20
```

Contoh 14

```
/*Contoh Program Mencetak Nilai 2 Pangkat 1 sampai 15*/
#include <stdio.h>
#include <math.h>
main()
{
float m;
double pangkat(int a);
int i,k;
clrscr();
for(i=1;i<=15;i++)
{
  printf("2 pangkat %d = %.2f\n",i,pangkat(i));
  }
 return 0;
}
double pangkat(int a){
double  c; int j;
c=1;
for(j=1;j<=a;j++){
```

```
      c=c*2;
}
return c;
}
```

Jika program contoh 14 dijalankan, hasil tampilannya adalah sebagai berikut:

```
2 pangkat 1 = 2.00
2 pangkat 2 = 4.00
2 pangkat 3 = 8.00
2 pangkat 4 = 16.00
2 pangkat 5 = 32.00
2 pangkat 6 = 64.00
2 pangkat 7 = 128.00
2 pangkat 8 = 256.00
2 pangkat 9 = 512.00
2 pangkat 10 = 1024.00
2 pangkat 11 = 2048.00
2 pangkat 12 = 4096.00
2 pangkat 13 = 8192.00
2 pangkat 14 = 16384.00
2 pangkat 15 = 32768.00
```

Contoh 15

```
/*Contoh Program Mencari Nilai x Pangkat y*/
#include<stdio.h>
#include<conio.h>
long int pangkat(int a,int b);
main(){
int i;
int j,b1,b2;
clrscr();
printf("Input nilai x dan y = "); scanf("%d%d",&b1,&b2);
printf("Nilai %d pangkat %d = %ld" ,b1,b2,pangkat(b1,b2));
getch();
return 0;
}

long int pangkat(int a,int b){
long int c; int j;
c=1;
for(j=1;j<=b;j++){
   c=c*a;
}
return c;
}
```

Jika program contoh 15 dijalankan, contoh tampilannya adalah sebagai berikut:

```
Input nilai x dan y = 3 4
Nilai 3 pangkat 4 = 81
```

2. Variabel global (eksternal)
   Variabel global (eksternal) adalah variabel yang dideklarasikan di luar fungsi.
   Sifat-sifat variabel global :
   - Dikenal (dapat diakses) oleh semua fungsi.
   - Jika tidak diberi nilai awal secara otomatis berisi nilai nol.
   - Dideklarasikan dengan menambahkan kata "**extern**" (opsional).

Contoh 16

```
/*Contoh Program Penggunaan Variabel Global*/
#include <stdio.h>
int i = 273; /* variabel global */
void tambah(void);
main()
{
      printf("\nNilai awal i = %d\n", i);
      i += 7;
      printf("Nilai i kini = %d\n", i);
      tambah();
      printf("Nilai i kini = %d\n", i);
      tambah();
      printf("Nilai i kini = %d\n", i);
}
void tambah(void)
{
      i++;
}
```

Jika program contoh 16 dijalankan, hasil tampilannya adalah sebagai berikut:

```
Nilai awal i = 273
Nilai i kini = 280
Nilai i kini = 281
Nilai i kini = 282
```

Contoh 17

```
#include <stdio.h>
int i = 273; /* variabel global */
void tambah(void);
main()
{
      extern int i; /* variabel eksternal */
      printf("\nNilai awal i = %d\n", i);
      i += 7;
      printf("Nilai i kini = %d\n", i);
      tambah();
      printf("Nilai i kini = %d\n", i);
      tambah();
      printf("Nilai i kini = %d\n", i);
}
```

```
void tambah(void)
{
      extern int i; /* variabel eksternal */
      i++;
}
```

Pada contoh 17 ada penambahan kata "extern" pada deklarasi variabel "i", yang menunjukkan bahwa "i" adalah variabel global. Jika program contoh 17 dijalankan, hasil tampilannya akan sama dengan tampilan program contoh 16.

Contoh 18

```
#include "stdio.h"
#include "conio.h"
void tampil(void);
int i = 25; /* variabel global */
void main()
{
clrscr();
printf("Nilai variabel i dalam fungsi main() adalah %i\n", i);
tampil();
i = i * 4; /* nilai i yang dikali 4 adalah 25 (global) bukan 10 */
printf("Nilai variabel i dalam fungsi main() sekarang adalah %i\n",i);
getch();
}
void tampil(void){
int i = 10; /* variabel lokal */
printf("Nilai variabel i dalam fungsi tampil() adalah %i\n", i);
}
```

Jika program contoh 18 dijalankan, hasil tampilannya adalah sebagai berikut:

```
Nilai variabel i dalam fungsi main() adalah 25
Nilai variabel i dalam fungsi tampil() adalah 10
Nilai variabel i dalam fungsi main() sekarang adalah 100
```

3. Variabel Statis

Variabel statis adalah variabel yang nilainya tetap dan bisa berupa variabel lokal (internal) ataupun variabel global (eksternal). Sifat-sifat variabel statis :

- Jika bersifat internal (lokal), maka variabel hanya dikenal oleh fungsi tempat variabel dideklarasikan.
- Jika bersifat eksternal (global), maka variabel dapat dipergunakan oleh semua fungsi yang terletak pada program yang sama.
- Nilai variabel statis tidak akan hilang walau eksekusi terhadap fungsi telah berakhir.
- Inisialisasi hanya perlu dilakukan sekali saja, yaitu pada saat fungsi dipanggil pertama kali.
- Jika tidak diberi nilai awal secara otomatis berisi nilai nol.
- Dideklarasikan dengan menambahkan kata "**static**".

Contoh 19

```
/*Contoh Program Penggunaan Variabel Statis*/
#include <stdio.h>
```

```
void fung_y(void);
main()
{
      int y = 20;
      fung_y();
      fung_y();
      printf("Nilai y dalam main() = %d\n", y);
}
void fung_y(void)
{
      static int y; /*Variabel statis*/
      y++;
      printf("Nilai y dalam fung_y() = %d\n", y);
}
```

Jika program contoh 19 dijalankan, hasil tampilannya adalah sebagai berikut:

```
Nilai y dalam fung_y() = 1
Nilai y dalam fung_y() = 2
Nilai y dalam main() = 20
```

4. Variabel Register

Variabel Register adalah variabel yang nilainya disimpan dalam resister dan bukan dalam memori RAM. Sifat-sifat variabel register :

- Hanya dapat diterapkan pada variabel lokal yang bertipe int dan char.
- Digunakan untuk mengendalikan proses perulangan (looping).
- Proses perulangan akan lebih cepat karena variabel register memiliki kecepatan yang lebih tinggi dibandingkan variabel biasa.
- Dideklarasikan dengan menambahkan kata "**register**".

Contoh 20

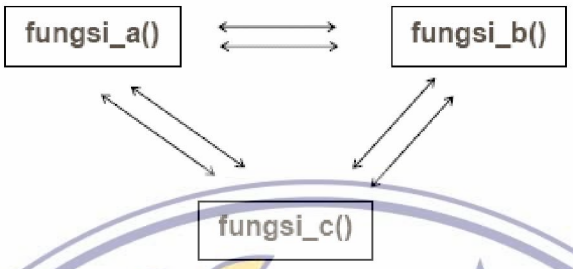
```
#include "stdio.h"
#include "conio.h"
void main(){
register int i; /* variabel register */
int jumlah;
clrscr();
for(i=1; i<=100; i++)
jumlah = jumlah + i;
printf("1+2+3+...+100 = %i\n", jumlah);
getch();
}
```

Jika program contoh 20 dijalankan, hasil tampilannya adalah sebagai berikut:

```
1+2+3+...+100 = 5918
```

7. Menciptakan Sejumlah Fungsi

Pada Bahasa 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.



Contoh 21

```
/*Contoh fungsi dalam fungsi*/
#include <stdio.h>
void fungsi_1(void);
void fungsi_2(void);
main()
{
      fungsi_1();
}
void fungsi_1()
{
      puts("fungsi 1 dijalankan");
      fungsi_2();
}
void fungsi_2()
{
      puts("fungsi 2 dijalankan");
}
```

Jika program contoh 21 dijalankan, hasil tampilannya adalah sebagai berikut:

```
fungsi 1 dijalankan
fungsi 2 dijalankan
```

8. Fungsi Rekursif

Fungsi rekursif adalah fungsi yang memanggil dirinya sendiri.

Contoh 22

```
/*Contoh fungsi rekursif untuk menyelesaikan faktorial sebuah bilangan*/
#include "stdio.h"
#include "conio.h"
long int faktorial(int N); /* prototype fungsi faktorial */
void main(){
int N;
clrscr();
printf("Berapa factorial ? "); scanf("%i", &N);
```

```
printf("Faktorial dari %i = %ld\n", N, faktorial(N));
getch();
}
long int faktorial(int N) /* definisi fungsi faktorial */
{
if(N==0)
return(1);
else
return(N * faktorial(N - 1));
/* fungsi faktorial() memanggil fungsi faktorial()*/
}
```

Jika program contoh 22 dijalankan, contoh hasil tampilannya adalah sebagai berikut:

```
Berapa factorial ? 7
Faktorial dari 7 = 5040
```

D. DAFTAR PUSTAKA

Budi Rahardjo, 2006, **Pemrograman C++: Mudah dan Cepat Menjadi Master C++ dengan Mengungkap Rahasia-rahasia Pemrograman dalam C++**, Bandung: Informatika.

Budi Sutedjo, S.Kom., MM dan Michael AN, S.Kom., 2004, **Algoritma dan Teknik Pemrograman Konsep, Implementasi dan Aplikasi**, Yogyakarta: ANDI.

Fathul Wahid, 2004, **Dasar-Dasar Algoritma dan Pemrograman**, Yogyakarta: ANDI.

Heri Sismoro, 2005, **Pengantar Logika Informatika, Algoritma dan Pemrograman Komputer**, Yogyakarta: ANDI.

Jogiyanto Hartono, 2000, **Konsep Dasar Pemrograman Bahasa C**, Yogyakarta: ANDI.

..... , 2004, **Pengenalan Komputer: Dasar Ilmu Komputer, Pemrograman, Sistem Informasi dan Inteligensi Buatan**, Yogyakarta: ANDI.

Yulikuspartono, S.Kom., 2004, **Pengantar Logika dan Algoritma**, Yogyakarta: ANDI.