Keg. Pembelajaran 7 : Fungsi dalam C++

1. Tujuan Kegiatan Pembelajaran

Setelah mempelajari materi kegiatan pembelajaran ini mahasiswa akan dapat :

1) Memahami konsep fungsi dalam pemrogram C++ secara benar.
2) Mengenal bentuk fungsi dalam pemrograman v secara benar
3) Dapat membuat fungsisendiri dalam aplikasinya pada pembuatan program secara tepat
4) Dapat mengembangkan bentuk-bentuk fungsi dalam pemrograman secara benar

2. Uraian Materi

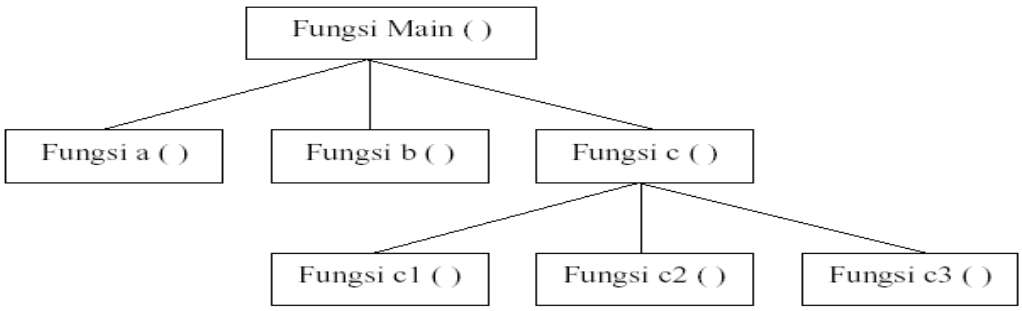
a. Pengertian Fungsi

Fungsi adalah sebuah fungsi berisi sejumlah pernyataan yang dikemas dalam sebuah nama. Selanjutnya nama ini dapat dipanggil di beberapa tempat dalam program. 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. Tujuan pembuatan fungsi adalah memudahkan dalam pengembangan program. Ini merupakan kunci dalam pembuatan program yang terstruktur. Menghemat ukuran program. 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 (1) fungsi pustaka

atau fungsi yang telah tersedia dalam Turbo C++ ; dan (2) fungsi yang didefinisikan atau dibuat oleh programmer.

Fungsi digunakan agar pemrogram dapat menghindari penulisan bagian program (kode) berulang-ulang, dapat menyusun kode program agar terlihat lebih rapi dan kemudahan dalam *debugging* program. Parameter adalah nama-nama peubah yang dideklarsikan pada bagian header fungsi. Pemrogram dapat membuat fungsi yang didefinisikan sendiri olehnya.



Gambar 32. Diagram definisi fungsi dalam fungsi

Contoh Sebuah fungsi yang namanya void garis

```
// contoh pembuatan dan pemanggilan fungsi garis
#include <iostream.h>
#include <conio.h>
#include <iomanip.h>
void garis();          //prototipe fungsi garis ();
void main()
{
   clrscr ();          //hapus layar
   garis();            //panggil fungsi garis ();

 cout <<setiosflags(ios::left);              //atur rata kiri
 cout <<setw(26) << "J u d u l  B u k u "<< "PENGARANG " << endl;
      garis ();
 cout << setw(26) <<"Mastering Borland C++"<<"Tom Swan" << endl;
 cout << setw(26) <<"Turbo C++ By Example"<<"M Johnson" << endl;
 cout << setw(26) <<"Converting C to  C++"<<"Lan Dorfman" << endl;
      garis ();
}
```

Keterangan :

```
void garis()          //definisi fungsi garis (tidak memakai titik koma)
{
  int i;
    for (i=0; i< 49; i++)
    cout << '-';
    cout << endl;
}
```

Catatan :

Di dalam fungsi, pemrogram dapat mendefinisikan variabel fungsi (seperti pada fungsi garis() ataupun membuat konstanta, akan tetapi di dalam fungsi tidak dianjurkan untuk membuat fungsi yang lain.

- Umumnya fungsi menerima masukkan (disebut **Argumen** atau **parameter**)
- Masukkan ini selanjutnya diproses oleh fungsi.
- Hasil akhir fungsi berupa sebuah nilai yang disebut **Nilai Balik** (*return value*)
- Sebagai contoh, kalau terdapat pernyataan :

  Kap = toupper (huruf);

  Maka :Huruf adalah argumen bagi fungsi toupper()

  Toupper() merupakan nilai balik (berupa huruf kapital dari huruf) ke variabel kap.

Gambar 33. Bentuk blackbook fungsi

b. Proptotipe Fungsi

Sebuah fungsi tidak dapat dipanggil kecuali sudah dideklarasikan.

Perhatikan contoh berikut ini :

```
void main ()
{
   printf ("hai\n");
}
```

Jika program ini dijalankan , C++ akan memberikan kesalahan, yang menyatakan bahwa **printf()** tidak memiliki prototipe.

Hal seperti ini masih bisa lolos pada bahasa C, akan tetapi tidak demikian pada C++.

Kesalahan tersebut tidak akan terjadi jika ditulis seperti berikut :

```
#include <stdio.h>
Void main ()
{
        printf ("Hai\n");
}
```

Mengapa ? Tidak lain karena prototipe **fungsi printf()** ada pada file **stdio.h.**

Oleh karena itu diperlukan untuk menyertakan baris berbentuk :

#include <nama_file_header>

Sekiranya program melibatkan fungsi-fungsi yang disediakan sistem. Sebuah fungsi tidak dapat dipanggil kecuali sudah dideklaraikan, deklarasi fungsi dikenal dengan sebutan **prototipe fungsi.**

Prototipe fungsi berupa :1. Nama Fungsi; 2. Tipe nilai fungsi; 3. Jumlah dan tipe argument, dan diakhiri dengan **titik koma,** sebagaimana pada pendeklarasian variabel. Sebagai Contoh perhatikan berikut ini:

1. **Long** kuadrat (**long** l) ;

   Pada Contoh pertama, fungsi kuadrat ( ) mempunyai argumen bertipe long dan nilai balik bertipe **long**.

2. **Double** maksimal (**double** x, **double** y)

   Pada Contoh kedua, fungsi maks( ) mempunyai dua buah argumen, dengan masing-masing argumen bertipe double.

3. **Void** garis ( );

   Pada Contoh ketiga, fungsi garis ( ) tidak memiliki argumen dan nilai baliknya tidak ada (**void**).

Manfaat dari prototipe fungsi adalah untuk menjamin tipe argumen yang dilewatkan pada pemanggilan fungsi benar-benar sesuai. Tanpa adanya prototipe fungsi, amatlah mudah propgramer melakukan kesalahan tanpa sengaja dalam melewatkan argumen.

Pada prototipe fungsi, nama argumrn boleh ditiadakan. Sebagai contoh :

long kuadrat (long) ;

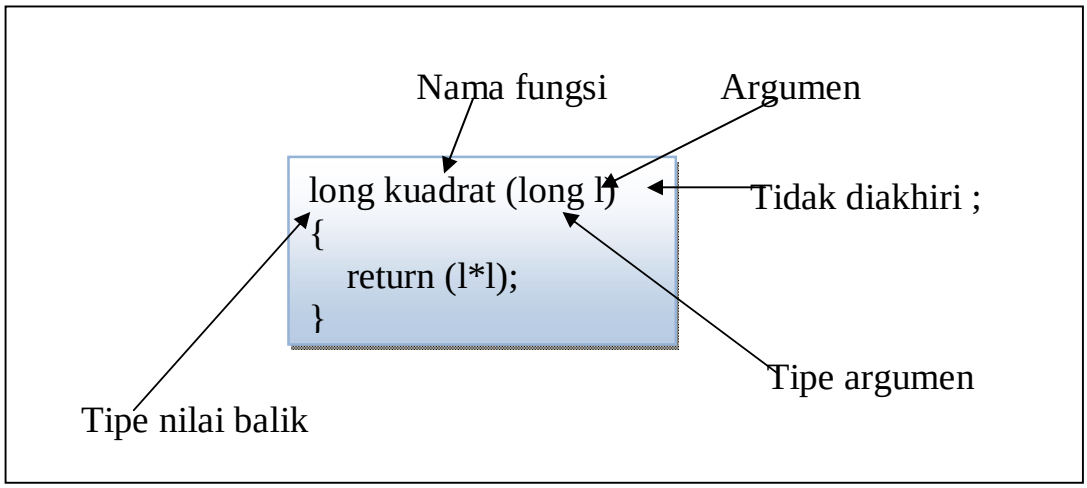
(Sebaiknya dihindari agar tidak membingungkan)

Berikut merupakan alternatif lain dari :

long kuadrat (long l) ;

c. Definisi Fungsi

Setiap fungsi yang dipanggil di dalam program harus didefinisikan. Letaknya boleh dimana saja. Khusus fungsi yang disediakan sistem, definisinya sudah ada di dalam pustaka.



Gambar 35. Bentuk definisi fungsi

Berikut contohnya :

```
/ / prototipe fungsi
long kuadrat (long l);
--------------------------------
/ / definisi fungsi
long kuadrat (long l)
{
  return ( l * l );
 }
```

Penjelasan :

- ✓ Pada pendefinisian fungsi (perhatikan) tidak ada titik koma. Pada prototipe harus pakai titik koma.
- ✓ Pernyataan ***return*** di dalam fungsi digunakan untuk memberikan nilai balik fungsi. Dalam hal ini fungsi kuadrat () memberikan nilai balik berupa nilai kuadrat dari argurmen.
- ✓ Perhatikan contoh program berikut ;

```
// contoh pembuatan fungsi dengan argumen bertipe long
// dan nilai balik juga berupa long
#include <iostream.h>
#include <conio.h>
#include <iomanip.h>
long kuadrat (long l);          //prototipe fungsi
void main()
{
  clrscr();
  for (long bil = 200; bil < 2000; bil +=200)
    cout <<setw(8) << bil
  << setw(8) << kuadrat (bil) << endl; / / pemanggilan fungsi kuadrat
}

/ / definisi fungsi
long kuadrat (long l)
{
  return (l * l);
}
```

Contoh Program pendefinisian fungsi maks (). Program ini berguna untuk mendapatkan nilai terbesar dari dua buah argumen

```
double maksimal (double x; double y)
{
   if (x>y)
         return(x);
   else
         return (y);
}
```

Catatan : penempatan return boleh dimana saja. Begitu perintah ini dijalankan, eksekusi terhadap fungsi berakhir.

Perhatikan contoh berikut ini:

// cintoh fungsi untuk memperoleh nilai terbesar di antara keduanya

```
#include <iostream.h>
#include <conio.h>
double maksimal (double x, double y);                    //prototipe fungsi
void main()
{
  clrscr();
   cout << maksimal (6578, 123) << endl;
   cout << maksimal (123, 6578) << endl;
}
// definisi fungsi
double maksimal (double x; double y)
{
  if (x>y)
       return(x);
  else
      return (y);
}
```

d. Fungsi tanpa Nilai Balik

Adakalanya suatu fungsi tidak perlu memiliki nilai balik. Misalnya fungsi hanya untuk menampilkan suatu keterangan saja. Pada fungsi ini tipe nilai balik fungsi yang diperlukan adalah **void.**

Perhatikan contoh berikut ini :

```
void tampilkan_judul ()
{
    cout << " PT Aria Angkasa "<<endl;
    cout << " JL Kadipura 38  "<<endl;
    cout << " Kabupaten KUDUS"<<endl;
    return ;        / / ada return di sini.
  }
```

e. Lingkup Variabel

Tujuannya mempelajari ini adalah : agar tidak salah dalam menggunakan suatu variable, lingkup variabel menentukan keberadaan suatu variabel ttt di dalam fungsi, ada variabel yang hanya dikenal di suatu fungsi dan tidak dikenal di fungsi lain, namun ada variabel yang dapat diakses oleh semua fungsi. Jenis variabel berdasarkan kelas penyimpanannya dibedakan menjadi :

1) Variabel lokal / Internal (otomatis)
2) Variabel eksternal
3) Variabel statis

Keterangan :

(1) Variabel Lokal

Variabel lokal adalah variabel yang dideklarasikan di dalam fungsi, dan hanya dikenal oleh di dalam fungsi tempat variabel didefinisikan.

Contoh:

```
void alpha ()
{
        int x = 20;
        double y = 3.14;
        cout <<"Pada alpha()  : x = "<<x
                <<" y =  " << y<< endl;
}
```

Dalam hal ini, x dan y berlaku sebagai variabel lokal di dalam fungsi alpha().

Perhatikan contoh programnya:

```
// Contoh Program untuk memperlihatkan efek variabel lokal

                #include <iostream.h>
#include <conio.h>
#include <iostream.h>
void alpha();                    // prototipe fungsi
void main()
{
   int x = 22;              // variabel lokal pada fungsi main()
   double y= 2.22;

   clrscr();
   cout << "Pada main () : x "<<x
        << " y = " << y <<endl<<endl;
   alpha();                 // panggil fungsi alpha
   cout << "Pada main() : x = " <<x
         << " y = " << y <<endl;
}
// definisi fungsi alpha()
void alpha ()
{
    int x = 20;
    double y = 3.14;
    cout <<"Pada alpha()  : x = "<<x
    <<" y =  " << y<< endl;
}
```

Dari program ini, perubahan x dan y pada alpha() tidak mempengaruhi variabel bernama sama fungsi main(). Variabel seperti ini pada fungsi alpha()maupun fungsi main() sering disebut variabel otomatis. Maka hal ini sering ditulis dengan kata kunci **auto.**

Contoh :

```
void alpha ()
{
        auto int x = 20;                // identik dengan int x;
        auto double y = 3.14;

        cout <<"Pada alpha()  : x = "<<x
                <<" y =  " << y<< endl;
}
```

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 .

(2) Variabel eksternal;

- Variabel eksternal merupakan kebalikan dari variabel otomatis.
- variabel yang didefinisikan di luar fungsi manapun
- sering disebut sebagai variabel global
- variabel dikenal di semua fungsi

Perhatikan contoh berikut :

```
// Contoh Program untuk memperlihatkan efek variabel global
#include <iostream.h>
#include <conio.h>

int skorku = 550;                   // variabel eksternal
void tambah();                      // prototipe fungsi
void main()
{
   clrscr();
   cout << "Pada awalnya Skorku adalah : "
         <<skorku<<    endl<<endl;
       tambah ();
```

```
        cout << "Setelah ada  fungsi tambah, Sekarang skorku: "
        <<skorku<< endl<<endl;
        tambah ();
        cout << "Setelah ada tambahan lagi, Sekarang skorku : "
          <<skorku<< endl;
}
//definisi fungsi tambah()
void tambah()
{
        skorku++;              //varibel eksternal dinaikkan.
}
```

Tampak bahwa sekalipun di dalam fungsi main() dan tambah() tidak ada pendefinisian variabel skorku, ternyata variabel ini dikenal (diadop) di kedua fungsi tersebut. Nilai dari variabel skorku() dapat diubah dari dalam fungsi tambah.

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).
- Dalam program yg besar dianjurkan sedikit mungkin menggunakan variabel eksternal (sering membingungkan)

(3) Variabel Statis

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

a) Jika variabel lokal berdiri sebagai variabel satatis, maka:
   - variabel tetap hanya dapat diakses pada fungsi yang mendefinisikannya
   - variabel tidak hilang saat eksekusi fungsi berakhir. Nilainya akan tetap dipertahankan , sehingga akan dikenali pada pemanggilan fungsi berikutnya.
   - Inisialisasi oleh pemrogram akan dilakukan sekali saja selama program dijalankan.
   - Jika tidak diberi nilai awal secara otomatis berisi nilai nol.

b) Jika variabel eksternal dijadikan sebagai variabel statis, maka:

- Variabel ini dapat diakses oleh semua file yang didefinisiakn pada file yang sama dengan variabel eksternal tsb.
- Variabel satais diperoleh dengan menambahkan kata kunci "**static**" di depan tipe variabel pada pernyataan pendefinisian.

Contoh :

```
static int waterloo;
Void abba()
{
        Static mamamia;
        ....
}
```

Perhatikan contoh program berikut :

```
// Contoh Program untuk memperlihatkan efek variabel lokal
#include <iostream.h>
#include <conio.h>
void saya_ingat();                    // prototipe fungsi
void main()
{
        int mamamia = 50;
        clrscr();
        saya_ingat ();
        saya_ingat ();
        saya_ingat ();
        cout << "Pragram utama main(): mamamia="
                <<mamamia<<endl;
}
    // pada fungsi berikut, mamamia didefinisikan sbg Var. Statistis
void saya_ingat()
{
        static int mamamia = 77;          // variabel statis
        mamamia++;
        cout<< "Saya_ingat () : mamamia = " <<mamamia<<endl;
}
```

Berdasarkan hasil ini, tampak bahwa :

- vaiabel statis mamamia pada fungsi saya_ingat() hanya diinisialisasi (bernilai 77) sekali saja.
- Kemudian setiap pemanggilan thd saya_ingat (), nilai variabel tsb dinaikkan sebesar 1 (mamamia++)
- Variabel bernama sama yang didefinisikan pada fungsi main() tidak ada kaitannya dengan variabel yang ada pada fungsi saya_ingat().
- Jika kata kunci **static** dihilangkan, maka var tsb selalu diinisialisasi = 77. Akibatnya hasil yang ditampilkan selalu 78 (mamamia++).

f. Operator Resolusi Lingkup (::).

Operator dengan menggunakan titik dua (::) disebut *scope resolution operator).*

Gunanya untuk mengakses variabel yang didefinisikan di luar suatu fungsi.

Manfaatanya adalah kalau di dalam suatu fungsi yang hendak mengakses variabel tsb terdapat variabel lokal (variabel yang didefinisikan di dalam fungsi dengan nama yang sama dg variabel di luar fungsi ybs).

Perhatikan contoh program berikut :

```
// Contoh Program pemakaian operator resolusi lingkup (:: )
#include <iostream.h>
#include <conio.h>

int x = 50;
void main()
{
        double x;               // devinisi variabel lokal
        clrscr();

        x = 5.678901;  // var lokal yg diberi nilai
```

```
        cout << x << " "<< ::x << endl;

        ::x = 77;                   / / variabel eksternal yg diberi nilai
        cout << x << " "<< ::x << endl;
}
```

g. Referensi

Fungsinya untuk memberikan nama alias dari variabel. Bentuk pendeklarasiannya :

Int &ref = nama_variabel

Tanda **&** mengawali nama referensi

Pengubahan nilai terhadap nama_variabel dapat dilakukan melalui nama_variabel itu sendiri melalui referensi ref. Perhatikan contoh berikut ini:

```
/ / Contoh penggunaan referensi
#include <iostream.h>
#include <conio.h>
    void main()
    {
        int i;
        int &r = i;      //deklarasi referensi
        clrscr ();
        i = 10;
        cout << "i = " << i << endl;
        cout << "r = " << r << endl;
                    r = 55;
        cout << "i = " << i << endl;
        cout << "r = " << r << endl;
    }
```

Dalam hal ini tampak bahwa pengubahan nilai terhadap i maupun r memberikan efek yang sama. Operator juga bekerja pada suatu variabel maupun referensinya dengan efek yang sama. *Perhatikan contoh berikut :*

```
// Contoh penggunaan operasi penaikan isi variabel
// melalui referensi
#include <iostream.h>
#include <conio.h>
void main()
    {
        int i = 55;
        int &r = i;      //deklarasi referensi
        clrscr ();

        cout << "i = " << i << " r  = "<< r << endl;
        i++;
        cout << "i = " << i << " r  = "<< r << endl;
        r++;
        cout << "i = " << i << " r  = "<< r << endl;
    }
```

Dalam hal ini ternyata karena **i** dan **r** menyiratkan **memori** yang sama. Perhatikan contoh berikut :

```
// Contoh untuk melihat alamat variabel alamat referensi
#include <iostream.h>
#include <conio.h>
void main()
    {
        int i = 55;
        int &r = i;      //deklarasi referensi
        clrscr ();

        cout << "alamat  : i  = " << &i
                << alamat : r = " << &r << endl;
    }
```

h. Rekursi

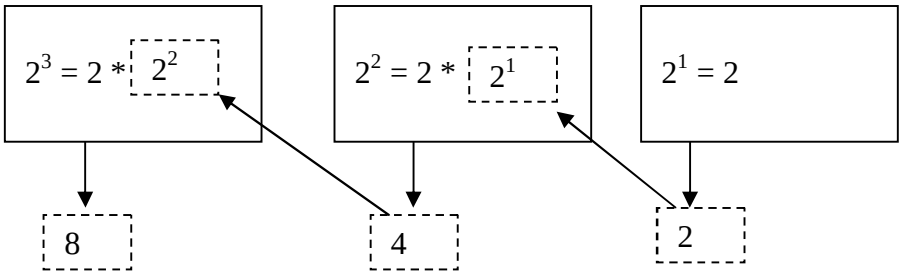
Dalam C++ dikenal fungsi rekursi, yaitu suatu fungsi dapat memanggil dirinya sendiri. Penggunaan fungsi rekursi ini bisa dilakukan pada saat menghitung nilai X^n , dengan n berupa bilangan positif.

Solusi persoalan ini adalah :

JIKA n = 1 MAKA $X^n = X$

SEALIN itu : $X^n = X * X^{n-1}$

Misalnya X = 2 dan n = 3, maka pemecahannya dapat dilihat pada gambar berikut ini:



Gambar 36. Bentuk Fungsi Rekursi

Implementasi programnya:

```
//Contoh program dengan fungsi Rekursi untuk bil pangkat n
#include <iostream.h>
#include <conio.h>
long int pangkat (int m, int n);   //prototype fungsi

void main()
{
        int x, y;

        clrscr ();
        cout << "MENGHITUNG x ^ y " << endl;
        cout <<" x = " ;
        cin >> x;
        cout <<" y = " ;
        cin >> y;

        cout << x << " ^ " << y << " = " << pangkat(x, y ) << endl;
}
long pangkat ( int m, int n)    //definisi fungsi
{
        if (n == 1)
                return (m);
        else
                return (m * pangkat(m, n-1));
}
```

i. Beberapa Fungsi Pustaka Dalam Bahasa C++

1. Fungsi Operasi String (tersimpan dalam header file “string.h”)

♣ strcpy()

a. Berfungsi untuk menyalin suatu string asal ke variable string tujuan.
b. Bentuk umum : strCpy(var_tujuan, string_asal);

♣ strlen()

a. Berfungsi untuk memperoleh jumlah karakter dari suatu string.
b. Bentuk umum : strlen(string);

♣ strcat()

a. Digunakan untuk menambahkan string sumber ke bagian akhir dari string tujuan.
b. Bentuk umum : strCat(tujuan, sumber);

♣ strupr()

a. Digunakan untuk mengubah setiap huruf dari suatu string menjadi huruf Capital.
b. Bentuk umum : strupr(string);.

♣ strlwr()

a. Digunakan untuk mengubah setiap huruf dari suatu string menjadi huruf kecil semua.
b. Bentuk umum : strlwr(string);

♣ strcmp()

a. Digunakan untuk membandingkan dua buah string.
b. Hasil dari fungsi ini bertipe integer dengan nilai :
   (a) Negative, jika string pertama kurang dari string kedua.
   (b) Nol, jika string pertama sama dengan string kedua
   (c) Positif, jika string pertama lebih besar dari string kedua.
c. Bentuk umum : **strcmp(string1, string2);**

2. **Fungsi Operasi Karakter (tersimpan dalam header “ctype.h”)**

♣ islower()

a. Fungsi akan menghasilkan nilai benar (bukan nol) jika karakter merupakan huruf kecil.
b. Bentuk umum : islower(char);

♣ isupper()

a. Fungsi akan menghasilkan nilai benar (bukan nol) jika karakter merupakan huruf kapital.
b. Bentuk umum : isupper(char);

♣ isdigit()

a. Fungsi akan menghasilkan nilai benar (bukan nol) jika karakter merupakan sebuah digit.
b. Bentuk umum : isdigit(char);

♣ tolower()

a. Fungsi akan mengubah huruf Capital menjadi huruf kecil.
b. Bentuk umum : tolower(char);.

♣ toupper()

a. Fungsi akan mengubah huruf kecil menjadi huruf kapital.
b. Bentuk umum : toupper(char);

3. **Fungsi Operasi Matematik (tersimpan dalam header “math.h”)**

♣ sqrt()

a. Digunakan untuk menghitung akar dari sebuah bilangan.
b. Bentuk umum : sqrt(bilangan);

♣ pow()

a. Digunakan untuk menghitung pemangkatan suatu bilangan.
b. Bentuk umum : pow(bilangan, pangkat);

♣ sin(), Cos(), tan()

a. Masing-masing digunakan untuk menghitung nilai sinus, Cosinus dan tangens dari suatu sudut.

b. Bentuk umum :

```
sin(sudut);
Cos(sudut);
tan(sudut);
```

♣ atof()

a. Digunakan untuk mengkonversi nilai string menjadi bilangan bertipe double.

b. Bentuk umum : atof(Char x);

♣ atoi()

a. Digunakan untuk mengkonversi nilai string menjadi bilangan bertipe integer.

b. Bentuk umum : atoi(Char x);

♣ div()

a. Digunakan untuk menghitung hasil pembagian dan sisa pembagian.

b. Bentuk umum : **div_t div(int x, int y)**

c. Strukturnya :

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

♣ max()

a. Digunakan untuk menentukan nilai maksimal dari dua buah bilangan.

b. Bentuk umum : max(bilangan1, bilangan2);

♣ min()

a. Digunakan untuk menentukan bilangan terkecil dari dua buah bilangan.

b. Bentuk umum : min(bilangan1, bilangan2);

2). Hal-hal yang perlu diperhatikan dalam penggunaan fungsi :

a. Kalau tipe fungsi tidak disebutkan, maka akan dianggap sebagai fungis dengan nilai keluaran bertipe integer.
b. Untuk fungsi yang memiliki keluaran bertipe bukan integer, maka diperlukan pendefinisian penentu tipe fungsi.
C. Untuk fungsi yang tidak mempunyai nilai keluaran maka dimasukkan ke dalam tipe void
d. Pernyataan yang diberikan untuk memberikan nilai akhir fungsi berupa pernyataan return.
e. Suatu fungsi dapat menghasilkan nilai balik bagi fungsi pemanggilnya.

3). Parameter Formal dan Parameter Aktual

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

Dalam Contoh program pertambahan 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

……..

}

4). Cara Melewatkan Parameter

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

1. **Pemanggilan Secara Nilai (*Call by Value*)**
   a. Call by value akan menyalin nilai dari parameter aktual ke parameter formal.
   b. Yang dikirimkan ke fungsi adalah nilai dari datanya, bukan alamat memori letak dari datanya.
   c. Fungsi yang menerima kiriman nilai akan menyimpannya di alamat terpisah dari nilai aslinya yang digunakan oleh bagian program yang memanggil fungsi.
   d. Perubahan nilai di fungsi (parameter formal) tidak akan merubah nilai asli di bagian program yang memanggilnya.
   e. Pengiriman parameter secara nilai adalah pengiriman searah, yaitu dari bagian program yang memanggil fungsi ke fungsi yang dipanggil.
   f. Pengiriman suatu nilai dapat dilakukan untuk suatu ungkapan, tidak hanya untuk sebuah variabel, elemen array atau konstanta saja.

2. Pemanggilan SeCara Referensi (*Call by Referene*)

c) Pemanggilan secara Referensi merupakan upaya untuk melewatkan alamat dari suatu variabel ke dalam fungsi.

d) Yang dikirimkan ke fungsi adalah alamat letak dari nilai datanya, bukan nilai datanya.

e) Fungsi yang menerima kiriman alamat ini akan menggunakan alamat yang sama untuk mendapatkan nilai datanya.

f) Perubahan nilai di fungsi akan merubah nilai asli di bagian program yang memanggil fungsi.

g) Pengiriman parameter secara referensi adalah pengiriman dua arah, yaitu dari fungsi pemanggil ke fungsi yang dipanggil dan juga sebaliknya.

h) Pengiriman secara acuan tidak dapat dilakukan untuk suatu ungkapan.

3. Rangkuman

Fungsi merupakan elemen utama dalam bahasa C++ karena bahasa C++ sendiri terbentuk dari kumpulan fungsi-fungsi. Tujuan pembuatan fungsi adalah memudahkan dalam pengembangan program. Ini merupakan kunci dalam pembuatan program yang terstruktur. Menghemat ukuran program. Dalam setiap program bahasa C++, minimal terdapat satu fungsi yaitu fungsi main(). Fungsi banyak diterapkan dalam program-program C++ yang terstruktur.

Fungsi digunakan agar pemrogram dapat menghindari penulisan bagian program (kode) berulang-ulang, dapat menyusun kode program agar terlihat lebih rapi dan kemudahan dalam debugging program. Kajian fungsi setidaknya meliputi defenisi fungsi, prototype fungsi , fungsi tanpa nilai balik, variable local dan ekternal. Jenis-jenis fungsi bawaan dari C++, antara lain fungsi : strcpy(); strlen();strcat();strupr();strlwr(); dan ♣ strcmp().

4. Latihan :

Buatkan dua buah program untuk menyelesaikan masalah sederhana, dengan ketentuan:

1) Program 1: Dengan melibatkan pemanfatan array (Dimensi 1 atau 2) di luar apa yang telah diberikan (analogi boleh)
2) Program 2: Dengan melibatkan fungsi untuk menyelesaikan tugas tertentu, yang dipasang pada program utama.