PERTEMUAN 5
FUNGSI

A. TUJUAN PEMBELAJARAN

Setelah menyelesaikan materi pada pertemuan ini, mahasiswa mampu mempraktekkan:

1. Definisi fungsi
2. Parameter fungsi array,struktur,dan pointer

B. URAIAN MATERI

1. Definisi Fungsi

Dalam C ++, suatu fungsi adalah sekelompok Pernyataan / intruksi yang bisa dipanggil dari satu titik manasajadalam program,Sintaks paling umum untuk mendefinisikan suatu fungsi adalah:

```
Type nama (parameter1, parameter2, ...)
{
Pernyataan
}
```

Keterangan :

a. type: adalah tipe nilai yang dikembalikan oleh fungsi.

b. nama : adalah pengenal yang dapat digunakan untuk memanggil fungsi tersebut.

c. parameter (sebanyak yang diperlukan): setiap parameter terdiri dari tipe data yang diikuti oleh pengenal, setiap parameter dipisahkan dengan koma(,). Setiap parameter terlihat sangat mirip dengan deklarasi variabel biasanya (misalkan .: int x), dan sebenarnya bertindak dalam fungsi sebagai variabel reguler yang bersifat lokal. Tujuan dari parameter adalah

untuk memungkinkan argumen diteruskan ke fungsi dari mana ia dipanggil.

d. Pernyataan : adalah tubuh dari fungsi tersebut. Ini adalah blok pernyataan yang dikelilingi oleh tanda kurung kurawal ({}) yang menetapkan apa yang sebenarnya dilakukan fungsi. Mari kita lihat contohnya:

```
// contoh fungsi
#include <iostream>
usingnamespacestd;

intPenjumlahan (intx, inty)
{
intz;
z=x+y;
returnz;
}

int main ()
{
intr;
r = Penjumlahan (5,3);
cout<<"The resultis "<< z;
}
```

Program ini dibagi menjadi dua fungsi: Penjumlahan dan Main. Ingatlah bahwa tidak peduli urutan mereka didefinisikan, program C ++ selalu dimulai dengan memanggil main. Faktanya, main adalah satu-satunya fungsi yang dipanggil secara otomatis, dan kode dalam fungsi lain hanya dijalankan jika fungsinya dipanggil dari main (langsung atau tidak langsung).

Dalam contoh di atas, main dimulai dengan mendeklarasikan variabel r bertipe int, dan setelah itu, main melakukan pemanggilan fungsi pertama: memanggil penjumlahan. Panggilan ke suatu fungsi mengikuti struktur yang sangat mirip dengan deklarasinya. Dalam contoh di atas, panggilan ke penambahan dapat dibandingkan dengan definisinya hanya beberapa baris sebelumnya:

```
Int penjumlahan ( int x, int y)


r = Penjumlahan (5,3);
```

Panggilan melewati dua nilai, 5 dan 3, ke fungsi; ini sesuai dengan parameter x dan y, yang dideklarasikan untuk penambahan fungsi.Pada titik di mana fungsi dipanggil dari dalam int main, kontrol diteruskan ke int penjumlahan fungsi: di sini, eksekusi main dihentikan, dan hanya akan dilanjutkan setelah fungsi penambahan berakhir. Pada saat pemanggilan fungsi, nilai dari kedua argumen (5 dan 3) disalin ke variabel lokal int x dan int y di dalam fungsi.

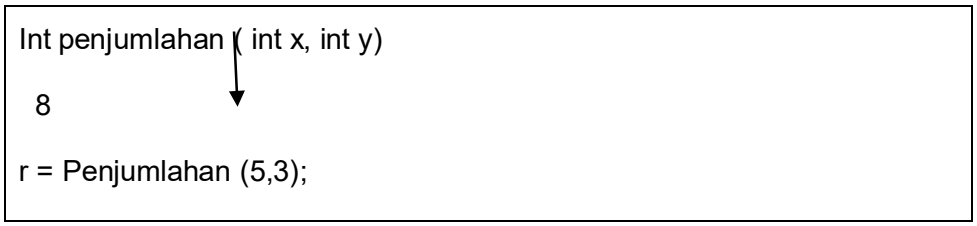
Kemudian, di dalam penjumlahan, variabel lokal lain dideklarasikan (int z), dan dengan menggunakan ekspresi x = x + y, hasil dari x plus y diberikan ke r; yang, untuk kasus ini, di mana x adalah 5 dan y adalah 3, berarti 8 ditetapkan ke z.

Pernyataan terakhir dalam fungsi:

```
Return z;
```

Mengakhiri fungsi penjumlahan, dan mengembalikan kontrol ke titik di mana fungsi itu dipanggil, kemudian, program melanjutkan perjalanannya pada pengembalian pada fungsi main tepat pada titik yang sama di mana ia terputus oleh panggilan untuk penjumlahan. karena penjumlahan memiliki tipe kembalian, panggilan dievaluasi sebagai memiliki nilai, dan nilai ini adalah nilai yang ditentukan dalam pernyataan pengembalian yang mengakhiri penjumlahan: dalam kasus khusus ini, nilai variabel lokal z, yang pada momen

pernyataan pengembalian memiliki nilai 8.



a. FUNGSI VOID

Sintaks yang ditunjukkan di atas untuk fungsi:

```
Type nama ( Argumen1, Argumen2, ...)
{
Pernyataan
}
```

Memerlukan deklarasi dimulai dengan sebuah tipe. Ini adalah tipe nilai yang dikembalikan oleh fungsi. Tetapi bagaimana jika fungsi tersebut tidak perlu mengembalikan nilai? Dalam hal ini, tipe yang akan digunakan adalah void, yaitu tipe khusus untuk merepresentasikan ketiadaan nilai. Misalnya, fungsi yang hanya mencetak pesan mungkin tidak perlu mengembalikan nilai apa pun, contoh nya seperti berikut :

```
// voidfunctionexample
#include <iostream>
usingnamespacestd;
void pesan ()
{
cout<<"saya fungsi void!";
}
int main ()
```

```
{
  pesan ();
}
```

void juga dapat digunakan dalam daftar parameter fungsi untuk secara eksplisit menentukan bahwa fungsi tersebut tidak menggunakan parameter aktual saat dipanggil. Misalnya, void Pesan dapat dideklarasikan sebagai berikut:

```
void pesan (void)
{
cout<<"saya fungsi void!";
}
```

Di C ++, daftar parameter kosong dapat digunakan sebagai pengganti void dengan arti yang sama, tetapi penggunaan void dalam daftar argumen dipopulerkan oleh bahasa C, di mana ini adalah persyaratan.

Sesuatu yang dalam hal apapun tidak bersifat opsional adalah tanda kurung yang mengikuti nama fungsi, baik dalam deklarasi maupun saat memanggilnya. Dan bahkan jika fungsi tidak mengambil parameter, setidaknya sepasang tanda kurung kosong harus selalu ditambahkan ke nama fungsi. Lihat bagaimana printmessage dipanggil dalam contoh sebelumnya:

```
Pesan ();
```

Tanda kurung adalah yang membedakan fungsi dari jenis deklarasi atau pernyataan.. Berikut ini tidak akan memanggil fungsi:

```
Pesan;
```

b. NILAI KEMBALI FUNGSI MAIN

Anda mungkin telah memperhatikan bahwa tipe kembalian dari main adalah int, tetapi sebagian besar contoh sebelumnya tidak benar-benar mengembalikan nilai main.

Nah, ada kekurangannya : jika eksekusi main berakhir secara normal tanpa menemui pernyataan return, compiler berasumsi bahwa fungsi diakhiri dengan pernyataan return implisit:

```
return 0;
```

Harap dicatat bahwa ini hanya berlaku untuk fungsi utama karena alasan sejarah. Semua fungsi lain dengan tipe pengembalian harus diakhiri dengan pernyataan pengembalian yang sesuai yang menyertakan nilai pengembalian, meskipun tidak pernah digunakan.

Ketika main mengembalikan nol (baik secara implisit maupun eksplisit), lingkungan menafsirkannya sebagai program yang berhasil dihentikan. Nilai lain dapat dikembalikan oleh main, dan beberapa lingkungan menyediakan akses ke nilai ini ke pemanggil dengan cara tertentu, meskipun perilaku ini belum tentu portabel di seluruh platform. Nilai utama yang dijamin akan ditafsirkan dengan cara yang sama di semua platform adalah:

Nilai	Keterangan
0	Program Berhasil
EXIT_SUCCESS	Program itu berhasil (sama seperti di atas). Nilai ini

	ditentukan diheader<cstdlib>
EXIT_FAILURE	Program gagal. Nilai ini ditentukan diheader<cstdlib>

Karena implisit return 0; Pernyataan untuk main adalah pengecualian yang rumit, beberapa penulis menganggapnya sebagai praktik yang baik untuk menulis pernyataan secara eksplisit.

c. Fungsi Rekrusi

Rekursi adalah properti yang fungsi harus dipanggil sendiri. Ini berguna untuk beberapa tugas, seperti mengklasifikasikan item atau menghitung faktorial angka. Misalnya, untuk mendapatkan faktorial sebuah bilangan (n!), Rumus matematikanya adalah:

n! = n * (n-1) * (n-2) * (n-3) ... * 1

Lebih khusus lagi, 5! (faktorial 5) akan menjadi:

5! = 5 * 4 * 3 * 2 * 1 = 120

Dan fungsi rekursif untuk menghitung ini di C ++ bisa jadi:

```
// aplikasi faktorial kalkulator
#include <iostream>
usingnamespacestd;

longfactorial (long a)
{
if (a > 1)
return (a * factorial (a-1));
```

```
else
return 1;
}

int main ()
{
longnumber = 9;
cout<<number<<"! = "<<factorial (number);
return 0;
}
```

Perhatikan bagaimana dalam fungsi faktorial kita menyertakan panggilan ke dirinya sendiri, tetapi hanya jika argumen yang diberikan lebih besar dari 1, karena jika tidak, fungsi akan melakukan looprekursif tak terbatas, di mana setelah mencapai 0 maka akan terus dikalikan dengan semua angka negatif (mungkin menyepertemuankan tumpukan melimpah di beberapa titik selama eksekusi).

2. Parameter fungsi pada array,struktur,dan pointer

a. Parameter Fungsi Pada Array

sedikit perbedaan ketika array dijadikan parameter yaitu subscriptnya tak terisi([])agar bisa diubah melalui fungsi yang lain. lebih jelasnya lihatlah program sebagai berikut :

```
// nama : parameter_fungsi_array
#include<iostream>
#include<cstdlib>
```

```
usingnamespacestd;

voidinput_array(int bil[],int data)
{
    int i;
    for(i=0;i<data;i++)
    {
        cout<<"Nilai bil["; cout<<i+1<<"] : ";
        cin>>bil[i];
    }
}
int jumlah_elemen(int bil[],int data)
{
    int i,jumlah=0;
    for(i=0;i<data;i++)
    {
        jumlah=jumlah+bil[i];
    }
    return jumlah;
}
int main(int argc,char*argv[])
{
    int bil[100];
    int data;
    int jumlah;
```

```
        cout<<"masukan berapa elemen data = ";
        cin>>data;
        cout<<endl;

input_array(bil,data);
        jumlah=jumlah_elemen(bil,data);
        cout<<endl;
        cout<<"Hasil penjumlahan elemen = "
        <<jumlah<<endl<<endl;
        return EXIT_SUCCESS;
}
```

Ketika memanggil fungsi(void) yang mempunyai parameterarray maka pengisian cukup berupa nama array-nya saja.

b. Parameter Fungsi Pada Struktur

Supaya pendeklarasian variabel Struct tidak berulang dan salah, disarankan untuk mendeklarasikan variabelnya secara lokal.

Untuk lebih jelasnya perhatikan program seperti berikut :

```
// nama : parameter fungsi pada array
#include<iostream>
#include<cstdlib>

usingnamespacestd;
typedefstruct
```

```
        {
        int bil_a;
        int bil_b;

        } data_bilangan;

        voidtampil_bilangan(data_bilangan bilangan)
        {
        cout<<"Bilangan satu = ";
        cout<<bilangan.bil_a<<endl;
        cout<<"Bilangan dua    = ";
        cout<<bilangan.bil_b<<endl;

        }
        int main(int argc,char*argv[])
        {
        data_bilangan bilangan;
        cout<<"Masukan bilangan kesatu : ";
        cin>>bilangan.bil_a;
        cout<<"Masukan bilangan dua   : ";
        cin>>bilangan.bil_b;
        cout<<endl;

        tampil_bilangan(bilangan);
        return EXIT_SUCCESS;
}
```

c. Parameter Fungsi Pada Pointer

Sama seperti pada array, parameter fungsi juga dapat digunakan pada pointer, karena sifatnya yang hanya sebagai penujuk, maka setiap perubahan yang terjadi pada parameter,sebenarnya terjadi pada variabel yang ditunjuk bukan pada variabel pointer.

dengan cara menambahkan operator (&) di depan argumen pada parameter aktual dan operator, (*) di depan argumen pada parameter resmi. Untuk lebih jelasnya lihat program seperti berikut :

```
#include<iostream>
#include<cstdlib>

using namespace std;

void penambahan ( int *angka)
{
*angka +=20;
}
int main(int argc,char*argv[])
{
int nilai=10;
cout<<"Nilai nya nilai adalah = "<<nilai<<endl<<endl;
penambahan(&nilai);
cout<<"Nilai nya nilai adalah = "<<nilai<<endl;
return EXIT_SUCCESS;
}
```

C. SOAL LATIHAN/TUGAS

Latihan	Petunjuk Pengerjaan Tugas
Latihan 5	1. Jelas menurut andan apa itu fungsi pada c++ 2. Sebutkan dan jelaskan manfaat menggunakan fungsi 3. Jelaskan pemanggilan fungsi dari callbyvalue dan callbyreference 4. Buat program menggunakan overloadimg fungsi 5. Buat program fungsi menggunakan parameter array,struktur dan pointer

D. REFERENSI

Adam Mukharil Bachtiar,(2018) *Pemrograman C dan C++.* Bandung

Abdul Kadir,(2013) *From zero to a pro pemrograman c++.Yogyakarta*

Moh.Sjukani . 2014. ALGORITMA (Algoritma & Struktur Data 1) dengan *C, C++ dan Java, Edisi 9, Mitra Wacana Media.*