# BAB VIII

# FUNCTION dan VOID

## Non Void (Fungsi dengan Nilai Kembalian)

Fungsi merupakan statemen yang dikelompokkan menjadi satu bagian kode (blok program) untuk menyelesaikan tugas spesifik tertentu. Intinya kita ingin **membagi-bagi program dari program utama (main program)** sehingga program dapat lebih rapih dan juga lebih terstruktur dan dapat dijalankan berulang kali.

Dalam sebuah program C++ minimal kita memiliki 1 method, karena pada dasarnya main() juga merupakan sebuah method. Method main adalah program utama kita.

1. **Jenis-jenis fungsi ada dua yaitu:**
2. Non *void* adalah fungsi dengan nilai balik atau fungsi dengan ***return* *value*** ketika fungsi dipanggil. Fungsi Non *Void*  wajib memiliki ***return* *value*** ketika ingin digunakan.
3. *Void* (Prosedur) adalah Fungsi yang tidak mengembalikan nilai atau fungsi yang tidak menghasilkan (***return value***) ketika fungsi dipanggil, atau biasa disebut dengan sebuah prosedur. Namun jika *Void* tetap diberikan ***return value*,** maka fungsi tersebut tetap bisa dijalankan.
4. **Keuntungan memakai fungsi :**
5. Menguraikan tugas pemrograman rumit menjadi langkah-langkah yang lebih sederhana atau kecil.
6. Mengurangi duplikasi kode (kode yang sama ditulis berulang-ulang) dalam program.

**Fungsi *non-void***  disebut juga *function*. Disebut *non-void* karena mengembalikan nilai kembalian yang berasal dari keluaran hasil proses *function* tersebut .

Ciri-ciri dari jenis fungsi *non-void* adalah sebagai berikut:

1. Ada *keyword* ***return***
2. Ada tipe data yang mengawali fungsi
3. Tidak ada *keyword* ***void***
4. Memiliki nilai kembalian

Bentuk umumnya:

|  |
| --- |
| tipe\_data\_kembalian nama\_fungsi (parameter1, parameter2,…)  {  /\* Badan Fungsi \*/  statemen\_yang\_akan\_dilakukan;  **return** nilai\_yang\_akan\_dikembalikan;  } |

Dari bentuk umum tersebut, yang perlu dijelaskan, yaitu:

1. **tipe\_nilai\_balik**

Tipe nilai balik adalah tipe data dari nilai yang akan dikembalikan oleh fungsi. Khusus untuk Fungsi yang tidak ditulis tipenya maka akan dianggap bertipe int atau char.

1. **nama\_fungsi**

Nama fungsi adalah nama yang akan digunakan dalam pemanggilan. Pemanggilan fungsi harus disesuaikan dengan daftar parameter yang disertakan pada saat pendefinisian fungsi.

1. **Parameter**

Fungsi bisa saja memiliki satu atau lebih parameter. Parameter itu sendiri merupakan variabel berisi nilai yang akan disertakan ke dalam proses yang dilakukan oleh fungsi. Jika tak ada parameter, judul fungsi berupa:

|  |
| --- |
| tipe\_nilai\_balik nama\_fungsi() |

1. **Badan fungsi**

Kode atau statemen-statemen yang akan dilaksanakan di dalam fungsi guna menyelesaikan tugas spesifik tertentu. Untuk bagian ini harus diawali dengan tanda **{** dan diakhiri dengan tanda **}.**

1. **Return Value**

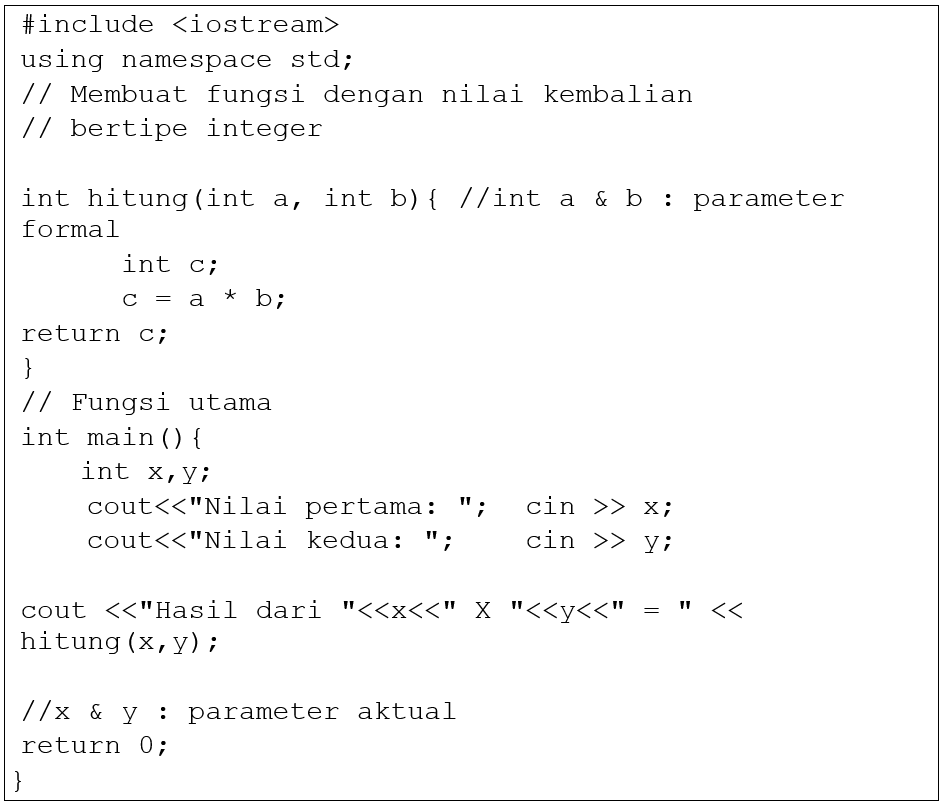
Perintah return pada fungsi artinya mengembalikan nilai yang tersimpan dalam sebuah variabel kepada kode yang akan memanggil function tersebut. Tujuan dari sebuah return adalah untuk mengembalikan sebuah nilai pada function.

### Fungsi dengan paramater

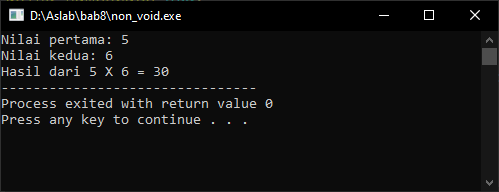
Fungsi dengan parameter yang mana parameter ini akan digunakan sebagai nilai masukkan (input) dalam sebuah fungsi. Nilai tersebut kemudian akan diproses oleh fungsi untuk menghasilkan sebuah nilai kembalian (*return* *value*).

Secara teori, parameter yang terdapat pada pendefinisian sebuah fungsi disebut dengan parameter formal, sedangkan parameter yang terdapat pada saat pemanggilan disebut dengan parameter aktual.

Untuk dapat lebih memahaminya, perhatikan contoh kode berikut :



Tampilan :

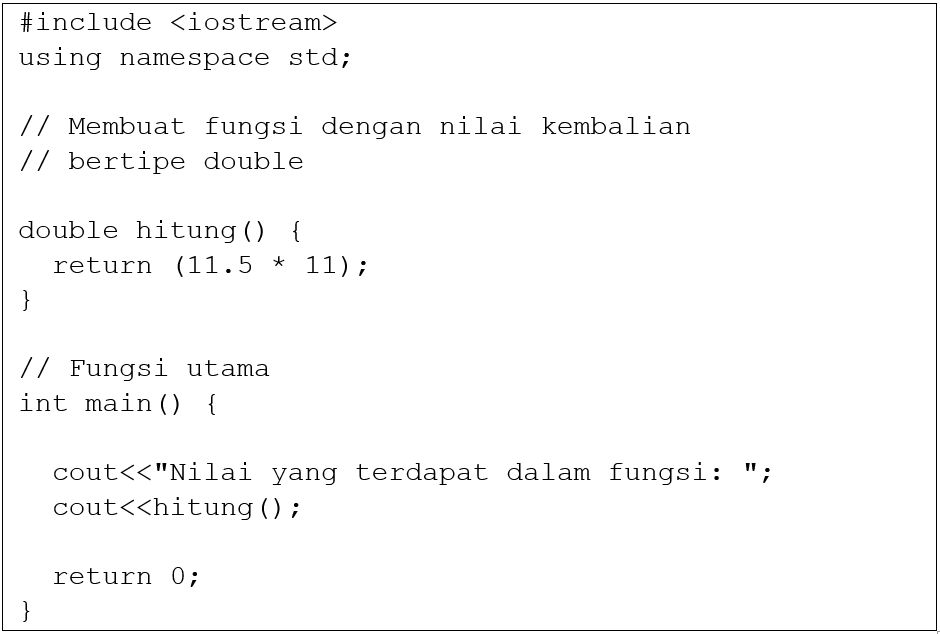


Analisa :

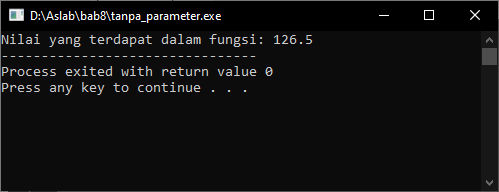
Pada kode diatas, variabel a dan b di atas sebagai parameter masukkan. Selain itu, a dan b disebut dengan parameter formal, sedangkan x dan y disebut dengan parameter aktual. Pada saat pemanggilan fungsi jumlah parameter aktual harus sama dengan jumlah parameter formal.

### Fungsi tanpa parameter

Fungsi tanpa parameter bisa didefinisikan sebagai fungsi yang setiap kali pemanggilan fungsi tersebut, hasil yang didapatkan selalu bernilai tetap. Untuk lebih memahami perbedaannya, pahami contoh berikut:



Tampilan :



Analisa :

Di program ini, tiap rumus memiliki tubuh/bagian tersendiri dan terpisah. Sehingga untuk merubahnya sangat mudah, jika kita memiliki rumus penghitungan yang banyak, pada cout<<hitung() adalah pemanggilan fungsi hitung untuk dieksekusi.

Perlu diingat, bahwa program C++ membaca program selalu dimulai dari main. Inisialisasi fungsi di awal diperlukan, karena sebagai pemberitahuan bahwa ada fungsi diluar dari main. Sehingga, saat eksekusi main, fungsi yang tertera di main akan dieksekusi, walaupun fungsi tersebut diluar dari main.

|  |  |  |
| --- | --- | --- |
| **#taugaksih**  **Default Parameter Value Dan Function Overloading**   1. **Default parameter value**   adalah nilai yang diberikan dalam deklarasi fungsi yang secara otomatis ditetapkan jika pemanggil fungsi tidak memberikan nilai untuk argumen dengan nilai default.  Contoh Program :   |  | | --- | | #include <iostream>  using namespace std;  int sum(int x,int y=0,int z=0){  int hasil;  hasil = x+y+z;  return hasil;  }  int main(){  cout<<sum(5)<<endl;  cout<<sum(5,2)<<endl;  cout<<sum(5,2,3)<<endl;  } |   Tampilan Program :    Analisa :  Pada program di atas, nilai default di paramater y dan z adalah 0, apabila kita memasukan nilai ke default value parameter maka nilai default value akan di gantikan dengan nilai yang kita input, misal seperti contoh di atas, nilai default y = 0, lalu setelah user memanggil fungsi sum dengan parameter y = 2, maka nilai y sekarang adalah 2, sehingga jika user memanggil fungsi sum(5,2) maka hasil nya adalah 7   1. **Function Overloading**   Pada function kita dimungkinkan untuk mendirikan function lebih dari satu dengan identitas yang sama, selama masing-masing function tersebut memiliki perbedaan pada function parameter. Hal tersebut disebut sebagai fitur Overloading. Function yang memiliki fitur overloading, memberikan kemudahan dengan kemungkinkan untuk memberikan argument secara fleksibel sesuai keadaan dan kebutuhkan.  Contoh Program :   |  | | --- | | #include <iostream>  using namespace std;  int hasil(int x){  int hasil;  hasil = x+x;  return hasil;  }  string hasil(string x){  return x;  }  int main(){  cout<<hasil(5)<<endl;  cout<<hasil("Lab. RPL")<<endl;  } |   Tampilan Program :    Analisa :  Pada contoh di atas, kita hanya memanggil satu identitas fungsi yaitu fungsi hitung, dengan otomatis CPU akan mengerti function mana yang cocok untuk mendapatkan argumen tersebut |

## Void (Fungsi tanpa Nilai Balik)

C++ tidak mengenal istilah prosedur (*procedure*) seperti pada saat kita melakukan pemrograman dengan menggunakan bahasa Pascal. Dalam bahasa Pascal, prosedur didefinisikan sebagai suatu proses yang tidak mengembalikan nilai. Untuk melakukan hal ini di dalam C++, maka harus membuat suatu fungsi dengan tipe *void,* yang berarti tidak memiliki nilai balik (*return value*)*.*

Ciri-ciri dari jenis fungsi void adalah sebagai berikut:

1. Secara umum tidak ada keyword return.
2. Tidak adanya tipe data di dalam deklarasi fungsi.
3. Menggunakan keyword void.

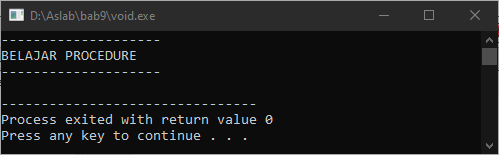
Bentuk umum dari pembuatan fungsi tanpa nilai balik ini adalah seperti yang terlihat di bawah ini.

|  |
| --- |
| void nama\_fungsi (parameter1, parameter2, …)  {  Statemen\_yang\_akan\_dilakukan;  ....  } |

Sebagai contoh penggunaannya perhatikan contoh dibawah ini :

|  |
| --- |
| #include<iostream>  using namespace std;  void garis(){ //tanpa parameter  for(int a=1;a<=20;a++){  cout<<"-";  }  }  int main(){  garis();  cout<<"\nBELAJAR PROCEDURE"<<endl;  garis();  cout<<endl;  return 0;  } |

Tampilan :



Analisa :

Pada contoh diatas, kita memiliki sebuah fungsi void dengan *statement* perulangan*.* Disana, tidak ada statement *return* karenatidak ada nilai yang dikembalikan*.* Untuk pemanggilan fungsi *void* tersebut tinggal menambahkan kata(); didalam fungsi utama.

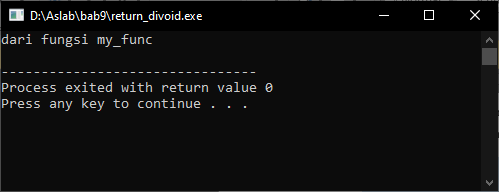
Dari fungsi void, kita tidak bisa mengembalikan nilai apa pun, tapi kita **bisa mengembalikan sesuatu selain nilai.** Beberapa diantaranya seperti di bawah ini.

1. **Return di fungsi void**

Fungsi void tidak dapat mengembalikan nilai apa pun. Tapi kita bisa menggunakan pernyataan return. return digunakan untuk menunjukkan bahwa fungsi tersebut **dihentikan**.

|  |
| --- |
| #include <iostream>  using namespace std;  void my\_func() {  cout << "dari fungsi my\_func" << endl;  return;  cout << "output yang tidak di baca";  }  int main() {  my\_func();  return 0;  } |

Tampilan :



Analisa :

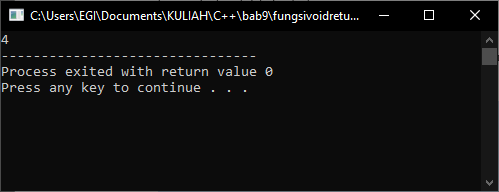
Di fungsi void kita bisa menggunakan pernyataan return, return disini digunakan untuk menghentikan fungsi my\_func, sehingga kode apapun setelah keyword return tidak di jalankan

1. **Fungsi void dapat mengembalikan fungsi void lainnya**

Fungsi void dapat me return kan fungsi lain, dengan syarat fungsi tersebut **harus fungsi void** juga.

|  |
| --- |
| #include <iostream>  using namespace std;  void hitung(int a)  {  int hasil;  hasil = a\*a;  cout<<hasil;  }  void panggil()  {  return hitung(2);  }  int main(){  panggil();  } |

Tampilan :



Analisa :

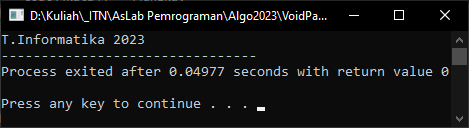
Selama fungsi void me return kan fungsi void juga, maka program tidak akan error. Pada contoh di atas, kita memiliki fungsi panggil() yang me return kan fungsi hitung(2), setelah itu nilai 2 masuk ke parameter void hitung yang selanjutnya melakukan operasi aritmatika.

1. **Fungsi void dengan banyak parameter**

Fungsi void dapat memakai lebih dari satu parameter dengan mendeklarasikan tipe\_data kemudian nama\_parameter.

|  |
| --- |
| #include <iostream>  using namespace std;  void contoh(int angka, string kata){  cout<<kata<<" "<<angka;  }  int main(){  int angka;  string kata;  angka = 2023;  kata = "T.Informatika";  contoh(angka,kata);  } |

Tampilan :



|  |  |
| --- | --- |
| **#taugaksih**  **Function Prototype**  Fungsi prototype atau prototype function sebenarnya merupakan sebuah variasi gaya dalam mendeklarasikan sebuah fungsi. dari yang di pelajari sebelumnya, penulisan fungsi di deklarasikan sebelum fungsi main, dengan tujuan agar fungsi main dapat mengenali fungsi-fungsi yang ada. Lalu bagaimana kalau suatu fungsi berada di bawah fungsi main, jelas terjadi error karena fungsi main tidak mengenali fungsi tersebut  Untuk menanggulangi hal tersebut, dalam pemrograman C++ dikenal istilah prototype function, dimana kamu wajib mendelarasikan nama fungsi, jenis fungsi, dan parameternya saja sebelum fungsi main agar dikenali oleh program utama. Untuk selanjutnya kamu bebas menyusun isi suatu fungsi mengenai apa saja yang hendak dilakukan dengan fungsi tersebut serta meletakanya setelah fungsi main().  Contoh Program :   |  | | --- | | #include <iostream>  using namespace std;  void persegi(int a); //function prototype  int main(){  persegi(5);  }  void persegi(int a){  cout<<"sisi = "<<a<<endl;  cout<<"Luas persegi adalah "<<a\*a;  } |   Tampilan Program : |

**Latihan!**

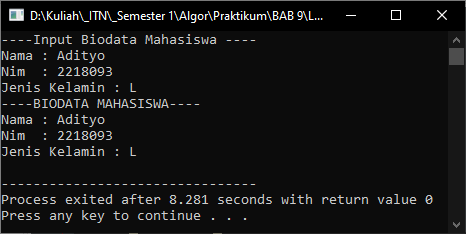
1. Membuat program memilih perhitungan Volume Bola dan Volume ½ Bola dengan menerapkan **fungsi dengan parameter dan fungsi tanpa parameter**. Dengan ketentuan :

a. Nilai jari-jari merupakan 2 digit NIM anda + **10**

contoh : NIM 2318001

Maka nilai jari-jari : 01 + 10 = 11

1. Buatlah program untuk menampilkan biodata mahasiswa, yang mempunyai inputan nama, nim dan jenis kelamin hasil dari inputan user menerapkan **void**.



1. Membuat program sub-rutin menentukan nilai fungsi nilai x adalah NIM anda