



# **Dasar Pemrograman**

Program Studi Informatika

## **Sesi 14 – Fungsi (Part 2)**

Syahid Abdullah, S.Si, M.Kom



# Outline Perkuliahan

- Jenis Variabel
- Fungsi Inline
- Function Overloading



# Manfaat Jenis Variabel

- Pada pemrograman C++, jenis variabel sangat berguna dalam penulisan suatu fungsi agar penggunaan suatu variabel tidak salah



# Jenis Variabel

## Variabel Lokal/Otomatis

Variabel yang dideklarasikan di dalam fungsi dan hanya dikenal oleh fungsi yang bersangkutan

## Variabel Eksternal/Global

variabel yang dideklarasikan diluar fungsi yang bersifat global yang artinya dapat digunakan bersama-sama tanpa harus dideklarasikan berulang-ulang.

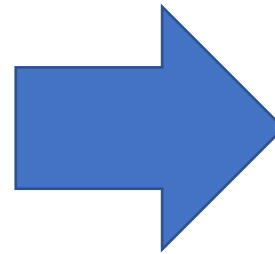
## Variabel Statis

- Variabel statis bersifat lokal
- Variabel statis bersifat eksternal
- Jika variabel statis tidak diinisialisasi maka otomatis diberikan nilai awal = 0



# Variabel Lokal

```
#include<conio.h>
#include<stdio.h>
#include<iostream.h>
lokal();
main()
{
    int a = 15;
    clrscr();
    cout<<"Pemanggilan Variabel Lokal"<<endl;
    cout<<"\nNilai didalam fungsi main() = : "<<a;
    lokal();
    cout<<"\nNilai didalam fungsi main() = : "<<a;
    getch();
}
lokal()
{
    int a = 10;
    cout<<"\nNilai a didalam fungsi lokal() = "<<a;
```



Hal ini terlihat bahwa variabel a yang berada diluar fungsi local tidak dikenal oleh fungsi local.

Output yang dihasilkan:

```
C:\BC5\LATIHAN CPP\Lat87.exe
Pemanggilan Variabel Lokal
Nilai didalam fungsi main() = : 15
Nilai a didalam fungsi lokal() = 10
Nilai didalam fungsi main() = : 15_
```

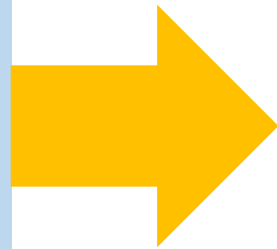


# Variabel Eksternal

```
#include<conio.h>
#include<stdio.h>
#include<iostream.h>
int a = 6; //--> deklarasi variabel eksternal
void lokal();
void main()
{
    clrscr();
    cout<<"Penggunaan Variabel Eksternal"<<endl;
    cout<<"\nNilai didalam fungsi main() = : "<<a;
    lokal(); //--> pemanggilan fungsi lokal
    cout<<"\nNilai Setelah panggilan fungsi lokal() = ";
    cout<<a;
    getch();
}

void lokal()
{
    a+=10;
}
```

Untuk pendeklarasian variabel eksternal ini, **diluar dari fungsi main()**



Hal ini terlihat bahwa variabel a yang dideklarasikan diluar fungsi main(), dapat digunakan didalam fungsi main() dan fungsi lokal().

Output yang dihasilkan:

```
Penggunaan Variabel Eksternal
Nilai didalam fungsi main() = : 6
Nilai Setelah panggilan fungsi lokal() = 16
```



# Variabel Statis

- Jika variabel statis bersifat local, maka variabel hanya dikenal oleh fungsi tempat variabel dideklarasikan.
- Jika variabel statis bersifat eksternal, maka variabel dapat dipergunakan oleh semua fungsi yang terletak pada file yang sama ditempat variabel statis dideklarasikan.
- Jika tidak ada inisialisasi oleh pemrograman secara otomatis akan diberikan nilai awal nol.



# Variabel Statis

Suatu variabel statis diperoleh dengan **menambahkan kata-kunci static didepan** penentu tipe data variabel.

Hal ini terlihat bahwa :

- Pada pada prototipe fungsi walah() tidak tedapat nilai awal, maka secara otomatis variabel k = 0.
- Pada pemanggilan fungsi walah() pertama, tercetak nilai variabel k = 4, didapat dari k=0+4.
- Pada pemanggilan fungsi walah() kedua, tercetak nilai variabel k = 8, didapat dari k=4+4, karena nilai k yang terbaru adalah 4.
- Pada pencetakan k didalam fungsi main(), adalah 5, karena variabel k, didalam fungsi main() bersifat lokal.

```
#include<conio.h>
#include<stdio.h>
#include<iostream.h>
walah(); //--> prototipe fungsi walah
main()
{
    int k = 5;
    clrscr();
    walah();
    walah();
    cout<<"\nNilai K didalam fungsi main() = "<<k;
    getch();
}
walah()
{
    static int k; //--> deklarasi variabel statis
    k += 4;
    cout<<"\nNilai K didalam fungsi() = "<<k;
}
```

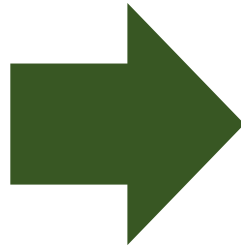




# Variabel Statis

Suatu variabel statis diperoleh dengan **menambahkan kata-kunci static didepan** penentu tipe data variabel.

```
#include<conio.h>
#include<stdio.h>
#include<iostream.h>
walah(); //--> prototipe fungsi walah
main()
{
    int k = 5;
    clrscr();
    walah();
    walah();
    cout<<"\nNilai K didalam fungsi main() = "<<k;
    getch();
}
walah()
{
    static int k; // --> deklarasi variabel statis
    k += 4;
    cout<<"\nNilai K didalam fungsi() = "<<k;
}
```



Output yang dihasilkan:

```
C:\BC5\LATIHAN CPP\lat89.exe
Nilai K didalam fungsi walah() = 4
Nilai K didalam fungsi walah() = 8
Nilai K didalam fungsi main() = 5_
```



# Fungsi Inline

- Fungsi inline (inline function) digunakan untuk mengurangi lambatnya eksekusi program dan mempercepat eksekusi program terutama pada program yang sering menggunakan atau memanggil fungsi yang berlebih. terutama program-program yang menggunakan pernyataan perulangan proses seperti for, while dan do – while.
- Inline function dideklarasikan dengan menambahkan kata kunci inline didepan tipe data.

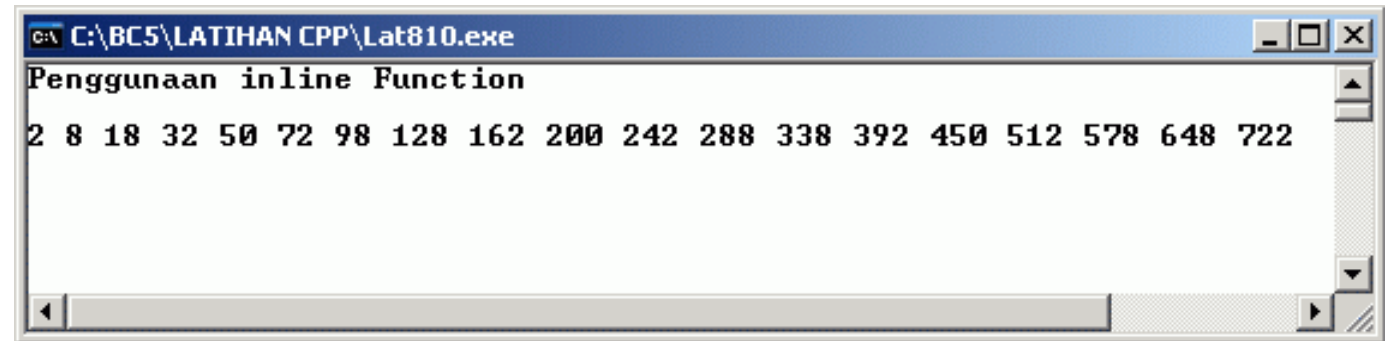


# Fungsi Inline

```
#include<conio.h>
#include<stdio.h>
#include<iostream.h>
inline int kali(int i, int j)
{
    return(i * j);
}

main()
{
    int k;
    clrscr();
    for(k = 1; k < 20; k++)
        cout<<kali(k, k*2)<<" ";
    getch();
}
```

Output yang dihasilkan:



The screenshot shows a Windows command prompt window titled "C:\BC5\LATIHAN CPP\Lat810.exe". The output of the program is displayed as a single line of numbers: "Penggunaan inline Function" followed by "2 8 18 32 50 72 98 128 162 200 242 288 338 392 450 512 578 648 722". The numbers are separated by spaces and represent the result of the inline function for k from 1 to 19.



# Fungsi Inline

```
#include<conio.h>
#include<stdio.h>
#include<iostream.h>
inline static void cplusplus()
{
    cout << "Pemrogramman C++\n";
    cout << "C++ Programming, ";
}

int main()
{
    cout << "Kami Sedang Belajar, ";
    cplusplus();
    cout << "Sedang Kami Pelajari.\n\n";
}
```

```
{
    cout << "Mari Belajar, ";
    cplusplus();
    cout << "Mudah Untuk Dipelajari.\n\n";
}
{
    cout << "Jika Sudah Mengerti, ";
    cplusplus();
    cout << "Berarti Sudah Anda Kuasai";
}

getche();
}
```

Output yang dihasilkan:



# Fungsi Overloading

- Mendefinisikan beberapa fungsi, sehingga memiliki nama yang sama tetapi dengan parameter yang berbeda
- Dapat diartikan bahwa fungsi yang overload berarti menyediakan versi lain dari fungsi tersebut
- Overloading merupakan salah satu kelebihan dari C++



# Fungsi Overloading

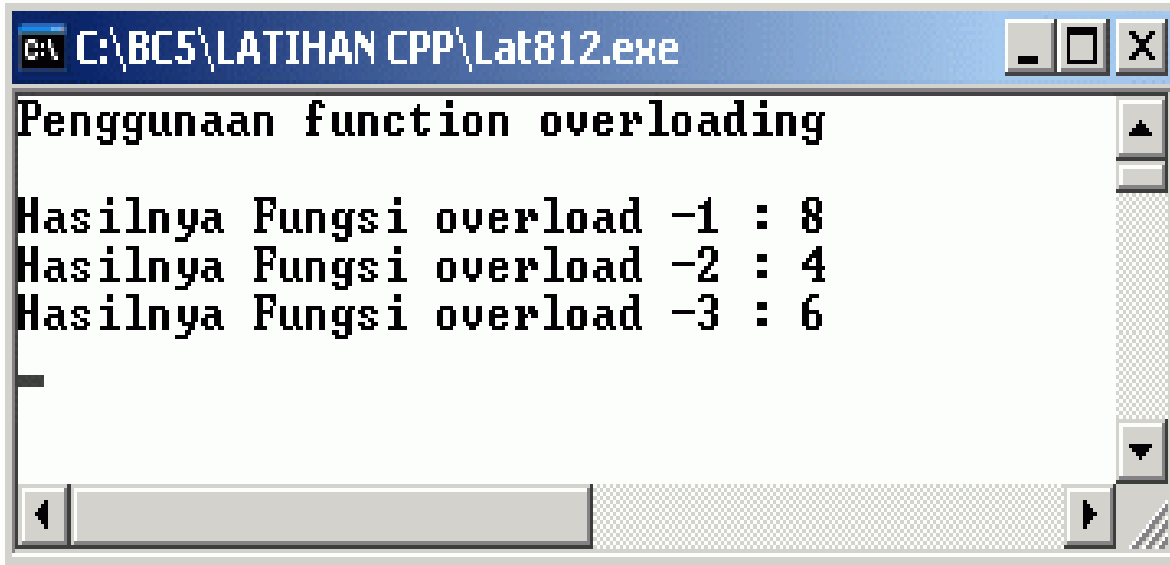
```
#include<conio.h>
#include<stdio.h>
#include<iostream.h>
int hitung(int b);
long hitung(long c);
float hitung(float d);
void main()
{
    clrscr();
    cout<< "Hasilnya Fungsi overload -1 : ";
    cout<< hitung(4)<< endl;
    cout<< "Hasilnya Fungsi overload -2 : ";
    cout<< hitung(2)<< endl;
    cout<< "Hasilnya Fungsi overload -3 : ";
    cout<< hitung(3)<< endl;
    getch();
}
```

```
int hitung(int b)
{
    return(b*b);
}

long hitung(long c)
{
    return(c*c);
}

double hitung(double d)
{
    return(d*d);
}
```

Output yang dihasilkan:



```
C:\BC5\LATIHAN CPP\Lat812.exe
Penggunaan function overloading
Hasilnya Fungsi overload -1 : 8
Hasilnya Fungsi overload -2 : 4
Hasilnya Fungsi overload -3 : 6
```



# Daftar Pustaka

- Frieyadie, *Pemrograman C++ dengan Borland C++ 5.02 (Edisi Revisi)*. DIKTAT KULIAH PEMROGRAMAN KOMPUTER BINA SARANA INFORMATIKA, 2007.
- Goodrich, Michael, Roberto Tamassia, and David Mount. *Data structures and algorithms in C++*. John Wiley & Sons, 2011.
- Mehlhorn, Kurt, and Peter Sanders. *Algorithms and data structures: The basic toolbox*. Springer, 2010.



Terima Kasih