

Pertemuan 9



Fungsi

Mata Kuliah : Algoritma & Pemrograman
Dosen : Tessy Badriyah, SKom., MT., PhD.



Tujuan Pembelajaran

- Penggunaan Fungsi dalam program Bahasa C untuk menjalankan bagian program yang mengerjakan suatu tugas tertentu.
- Fungsi yang mengembalikan suatu nilai (return) dan fungsi yang tidak mengembalikan suatu nilai (void)
- Penggunaan fungsi library standart yang disediakan oleh C
- Pembuatan user defined function atau fungsi yang didefinisikan sendiri oleh user.



Tentang Fungsi

- Fungsi adalah blok kode yang mengerjakan satu hal tertentu.
- Misal, program yang berhubungan dengan tampilan grafik perlu membuat lingkaran dan mewarnainya sesuai dengan radius dan warna yang diinputkan oleh user.
- Maka untuk menyelesaikan hal tersebut, perlu dibuat :
 - Fungsi createLingkaran
 - Fungsi warna
- Membagi masalah yang kompleks menjadi bagian2 kecil akan membuat program menjadi lebih mudah dipahami.





Tipe Fungsi

- Fungsi standart library
 - Fungsi yang disediakan oleh C
- User defined functions
 - Fungsi yang dibuat oleh user



Contoh fungsi Standard library

- Contoh:
- Fungsi `printf()` => digunakan untuk menampilkan output akan bisa tampil di layar
- Fungsi ini didefinisikan dalam header file `"stdio.h"`.
- Fungsi standart yang lainnya banyak sekali, termasuk `scanf()`, `fprintf()`, `getchar()`, dll.
- Untuk satu include `"stdio.h"` pada program, semua fungsi standart library yang ada di dalamnya dapat digunakan.



USER DEFINED FUNCTIONS



User-defined Functions

- User defined functions adalah fungsi yang didefinisikan oleh user.
- Keuntungan dari penggunaan **user-defined function**:
 - Program menjadi lebih mudah dipahami, di-maintain dan diperiksa salahnya dimana.
 - Reusable codes artinya kode yang dibuat dapat digunakan kembali.
 - Program yang besar dapat dibagi ke dalam modul-modul yang lebih kecil.



Contoh: User-defined function

```
#include <stdio.h>
int addNumbers(int a, int b);    // function prototype
int main()
{
    int n1,n2,sum;
    printf("Masukkan dua bilangan positif: ");
    scanf("%d %d",&n1,&n2);
    sum = addNumbers(n1, n2);    // function call
    printf("sum = %d",sum);
    return 0;
}
int addNumbers(int a,int b)    // function definition
{
    int result;
    result = a+b;
    return result;              // return statement
}
```




Function prototype

- function prototype adalah deklarasi dari suatu fungsi yang berisi nama fungsi, parameter dan tipe kembaliannya.
- function prototype perlu diberikan jika fungsi letaknya ada di bawah program utama (main function).

```
returnType functionName(type1 argument1, type2 argument2,...);
```



Pemanggilan suatu fungsi

- Kontrol dari program akan dipindah ke user-defined function dengan cara memanggилnya.
- Cara penulisan dari fungsi pemanggilannya:


```
functionName(argument1, argument2, ...);
```



Definisi Fungsi

- Definisi fungsi berisi blok kode yang membentuk tugas tertentu, misal menambahkan dua bilangan dan mengembalikan hasilnya.
- Cara penulisan dari definisi fungsi:

```
returnType functionName(type1 argument1, type2 argument2, ...)  
{  
    //body of the function  
}
```



Melewatkan argumen ke fungsi

- Pada pemrograman, argumen mengacu pada variabel yang dilewatkan ke fungsi.
- Ada dua macam tipe argumen:
 - Argumen formal
 - Argumen pemanggilan

Cara melewati argumen ke fungsi



```
#include <stdio.h>

int addNumbers(int a, int b);

int main()
{
    ... ..

    sum = addNumbers(n1, n2);
    ... ..
}

int addNumbers(int a, int b)
{
    ... ..
    ... ..
}
```

Diagram illustrating argument passing: Arrows point from `n1` and `n2` in the function call `addNumbers(n1, n2);` to the parameters `a` and `b` in the function definition `int addNumbers(int a, int b)`.



Perintah Return

- Perintah return mengembalikan nilai dari fungsi ke program yang memanggil fungsi tersebut.

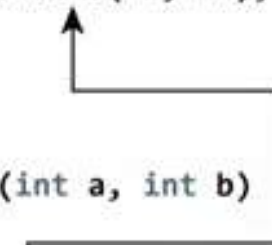
- Cara penulisannya :

```
return (expression);
```

- Contoh :

```
return a;  
return (a+b);
```

```
#include <stdio.h>  
  
int addNumbers(int a, int b);  
  
int main()  
{  
    ... ..  
    sum = addNumbers(n1, n2);  
    ... ..  
}  
  
int addNumbers(int a, int b)  
{  
    ... ..  
    return result;  
}
```



The diagram illustrates the execution flow. An arrow originates from the `return result;` statement within the `addNumbers` function and points to the `sum = addNumbers(n1, n2);` statement in the `main` function. A callout box labeled `sum = result` is positioned near the arrow, indicating that the value returned by the function is assigned to the variable `sum` in the calling function.



Tipe User-defined Functions

User-defined functions dapat dikategorikan sebagai :

- Fungsi tanpa argumen dan tanpa return
- Fungsi tanpa argumen dan dengan return
- Fungsi dengan argumen dan tanpa return
- Fungsi dengan argumen dan dengan return



Contoh : No arguments dan no return

```
#include <stdio.h>
void checkPrimeNumber();
int main()
{
    checkPrimeNumber();
    // no argument dilewatkan
    return 0;
}
// tipe dari return adalah void karena
fungsi tidak me-return suatu nilai
apapun
```

```
void checkPrimeNumber()
{
    int n, i, flag=0;
    printf("Enter a positive integer: ");
    scanf("%d",&n);
    for(i=2; i <= n/2; ++i)
    {
        if(n%i == 0)
        {
            flag = 1;
        }
    }
    if (flag == 1)
        printf("%d is not a prime number.", n);
    else
        printf("%d is a prime number.", n);
}
```




Contoh : No arguments tapi ada return

```
#include <stdio.h>
int getInteger();
int main()
{   int n, i, flag = 0;
    // no argument dilewatkan ke fungsi
    // nilai yang dikembalikan dari fungsi ditampung
    di variabel n
    n = getInteger();
    for(i=2; i<=n/2; ++i)
    {
        if(n%i==0){
            flag = 1;
            break;
        }
    }
}
```

```
if (flag == 1)
    printf("%d adalah bukan prima.", n);
else
    printf("%d adalah bilangan prima.", n);
return 0;
}

// fungsi getInteger() mengembalikan integer yang
dimasukkan oleh user
int getInteger()
{   int n;

    printf("Masukkan integer positif: ");
    scanf("%d",&n);

    return n;
}
```



Contoh : Ada arguments tapi tidak ada yang di-return

```
#include <stdio.h>
void checkPrimeAndDisplay(int n);
int main()
{   int n;
    printf("Masukkan integer positif: ");
    scanf("%d",&n);
    // n dilewatkan ke fungsi
    checkPrimeAndDisplay(n);
    return 0;
}
// void menunjukkan bahwa tidak ada yang di-return
// dari fungsi
void checkPrimeAndDisplay(int n)
{   int i, flag = 0;
```

```
    for(i=2; i <= n/2; ++i)
    {
        if(n%i == 0){
            flag = 1;
            break;
        }
    }
    if(flag == 1)
        printf("%d bukan bilangan prima.",n);
    else
        printf("%d adalah bilangan prime.", n);
}
```



Contoh : Ada arguments dan ada yang di-return

```
#include <stdio.h>
int checkPrimeNumber(int n);
int main()
{
    int n, flag;
    printf("Enter a positive integer: ");
    scanf("%d",&n);

    // n dilewatkan ke fungsi checkPrimeNumber()
    // nilai yang dikembalikan dari fungsi di-assign ke flag
    flag = checkPrimeNumber(n);
    if(flag==1)
        printf("%d bukan bilangan prima",n);
    else
        printf("%d adalah bilangan prima",n);
    return 0;
}
```

```
// integer dikembalikan dari fungsi
int checkPrimeNumber(int n)
{
    /* Nilai Integer dikembalikan dari fungsi
    checkPrimeNumber() */
    int i;
    for(i=2; i <= n/2; ++i)
    {
        if(n%i == 0)
            return 1;
    }
    return 0;
}
```



Yang sudah dipelajari

- Penggunaan Fungsi dalam program Bahasa C untuk menjalankan bagian program yang mengerjakan suatu tugas tertentu.
- Fungsi yang mengembalikan suatu nilai (return) dan fungsi yang tidak mengembalikan suatu nilai (void)
- Penggunaan fungsi library standart yang disediakan oleh C
- Pembuatan user defined function atau fungsi yang didefinisikan sendiri oleh user.



Referensi

- Robertson, Lesley Anne. (1992). *Students' guide to program design*. Oxford : Newnes
- Santner, Williams, and Notz (2003), *Design and Analysis of Computer Experiments*, Springer.
- Deitel & Deitel, *C How to Program*, Prentice Hall 1994 (2nd edition)
- Brookshear, J.G., *Computer Science: An Overview*, Benjamin-Cummings 2000 (6th edition)
- Kernighan & Ritchie, *The C Programming Language*, Prentice Hall