

JOB SHEET 08

Fungsi & Prosedur

*"Everybody should learn to program a computer,
because it teaches you how to think."*

TUJUAN PEMBELAJARAN

1. Mampu menjelaskan dan mengimplementasikan *Function & Procedure* dalam pemrograman menggunakan IDE.

POKOK MATERI

1. Pengertian Fungsi
2. Penggunaan Fungsi
3. Fungsi Rekursif
4. Variabel Lokal & Global
5. Pass by Value
6. Pass by Reference

URAIAN MATERI

A. Pengertian Fungsi

Fungsi adalah sub-program yang bisa digunakan kembali baik di dalam program itu sendiri, maupun di program yang lain.

Contoh fungsi yang sering kita buat adalah fungsi `main()`.

Fungsi ini memang wajib ada di setiap program C karena akan dieksekusi pertama kali.

Fungsi pada bahasa pemrograman C dapat kita buat dengan cara seperti ini:

```
int nama_fungsi(int parameter) {  
    // tubuh fungsi berisi  
    // kode program dari fungsi  
}
```

Fungsi biasanya akan mengembalikan sebuah nilai dari hasil prosesnya. Karena itu, kita harus menentukan tipe data untuk nilai yang akan dikembalikan. Apabila fungsi tersebut tidak memiliki nilai kembalian, maka kita harus menggunakan tipe void untuk menyatakan kalau fungsi tersebut tidak akan mengembalikan nilai apa-apa.

Contoh:

```
void nama_fungsi(){  
    printf("Ini adalah sebuah fungsi\n");  
}
```

Sekarang mari kita coba membuat program C dengan fungsi.

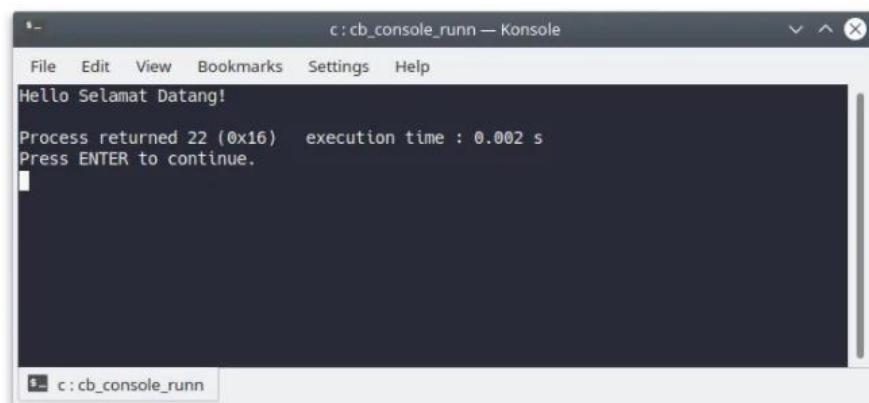
Silahkan buat file baru bernama contoh_fungsi.c kemudian isi dengan kode berikut:

```
#include <stdio.h>

// membuat fungsi say_hello()
void say_hello(){
    printf("Hello Selamat Datang!\n");
}

void main(){
    // memanggil fungsi say_hello()
    say_hello();
}
```

Hasilnya:



Fungsi say_hello() dapat kita panggil berulang kali pada fungsi main().

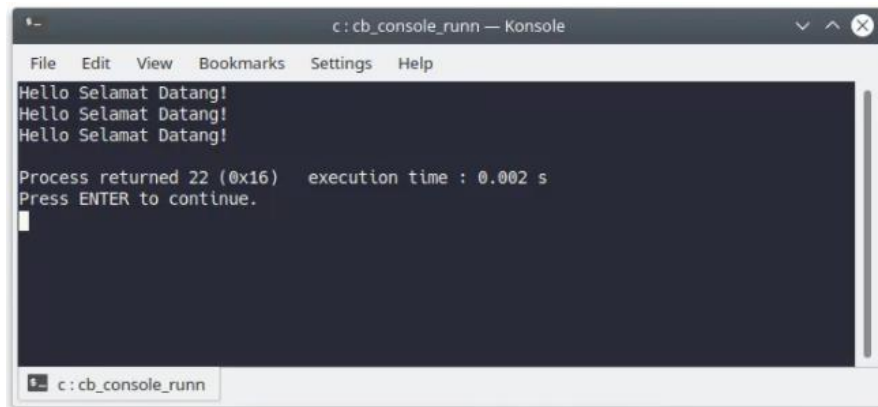
Contoh:

```
#include <stdio.h>

// membuat fungsi say_hello()
void say_hello(){
    printf("Hello Selamat Datang!\n");
}

void main(){
    // memanggil fungsi say_hello()
    say_hello();
    say_hello();
    say_hello();
}
```

Maka hasilnya:



Jadi, cukup buat fungsi satu kali. Kita bisa panggil berkali-kali.

Fungsi dengan Parameter

Parameter adalah variabel yang menyimpan nilai untuk diproses di dalam fungsi.

Parameter akan menyimpan nilai yang akan diinputkan ke dalam fungsi.

Contoh:

```
void say_hello(char name[]){  
    printf("Hello %s!\n", name);  
}
```

Pada contoh di atas, name adalah sebuah parameter berupa array dengan tipe char. Parameter ini hanya akan dikenali di dalam fungsi.

Lalu, bagaimana cara memanggil fungsi yang memiliki parameter?

Berikut caranya:

```
say_hello("Petani Kode");
```

Perhatikan! "Petani Kode" adalah nilai yang akan kita berikan ke parameter.

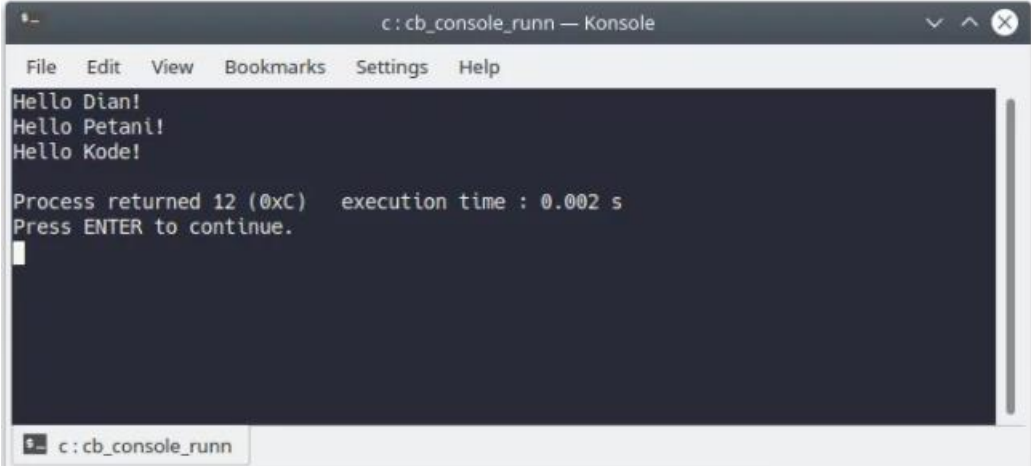
Silahkan buat program baru dengan nama parameter.c, kemudian isi dengan kode berikut:

```
#include <stdio.h>

void say_hello(char name[]){
    printf("Hello %s!\n", name);
}

void main(){
    say_hello("Dian");
    say_hello("Petani");
    say_hello("Kode");
}
```

Hasilnya:



Hasil outputnya akan menyesuaikan dengan nilai parameter yang kita berikan ke dalam fungsi. Lalu bagaimana kalau ada lebih dari satu parameter? Tinggal ditambahkan dan dipisah dengan tanda koma seperti ini:

```
void add(int a, int b){
    printf("%d + %d = %d\n", a, b, a+b);
}
```

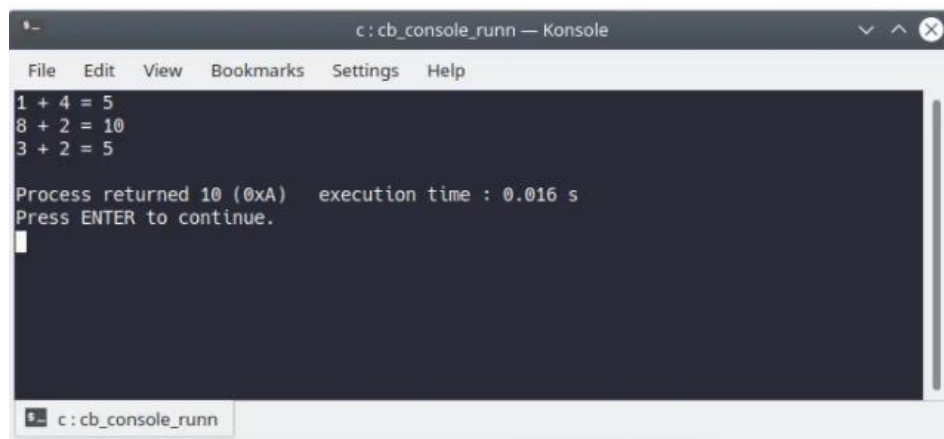
Buatlah program baru bernama dua_parameter.c, kemudian isi dengan kode berikut:

```
#include <stdio.h>

void add(int a, int b){
    printf("%d + %d = %d\n", a, b, a+b);
}

void main(){
    add(1, 4);
    add(8, 2);
    add(3, 2);
}
```

Hasilnya:



```
c: cb_console_runn — Konsole
File Edit View Bookmarks Settings Help
1 + 4 = 5
8 + 2 = 10
3 + 2 = 5
Process returned 10 (0xA) execution time : 0.016 s
Press ENTER to continue.
c: cb_console_runn
```

Fungsi yang mengembalikan nilai

Contoh:

```
void add(int a, int b){
    printf("%d + %d = %d\n", a, b, a+b);
}
```

Fungsi ini tidak akan mengembalikan apa-apa, karena tipe data yang diberikan pada nilai kembalian adalah void. Fungsi kadang harus mengembalikan sebuah nilai agar dapat diproses di tahap berikutnya. Kita bisa menggunakan kata kunci return untuk mengembalikan nilai dari fungsi.

Contoh:

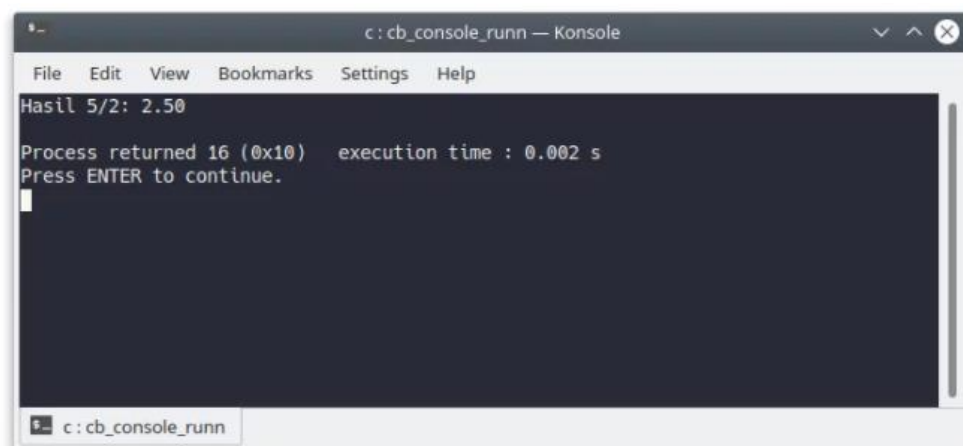
```
int add(int a, int b){  
    return a+b;  
}
```

Maka fungsi add() akan mengembalikan nilai berupa integer dari hasil penjumlahan nilai a dan b.

Silahkan buat program baru bernama fungsi_bagi.c, kemudian isi dengan kode berikut:

```
#include <stdio.h>  
  
float bagi(int a, int b){  
    float hasil = (float)a / (float)b;  
    return hasil;  
}  
  
void main(){  
    printf("Hasil 5/2: %.2f\n", bagi(5, 2));  
}
```

Hasilnya:



The screenshot shows a console window titled "c:cb_console_runn — Konsole". The window has a menu bar with "File", "Edit", "View", "Bookmarks", "Settings", and "Help". The output text is as follows:

```
Hasil 5/2: 2.50  
Process returned 16 (0x10)   execution time : 0.002 s  
Press ENTER to continue.
```

The console window has a dark background and a light-colored border. The output text is white. The window title bar is dark gray with standard Windows window controls (minimize, maximize, close).

Fungsi Rekursif pada C

Fungsi rekursif adalah fungsi yang memanggil dirinya sendiri.

Biasanya kita memanggil fungsi pada fungsi main atau fungsi yang lainnya.

Namun, pada fungsi rekursif, fungsi itu akan memanggil dirinya sendiri di dalam tubuh fungsi.

Coba perhatikan contoh berikut:

```
#include <stdio.h>
int sum(int n);

void main(){
    int number, result;

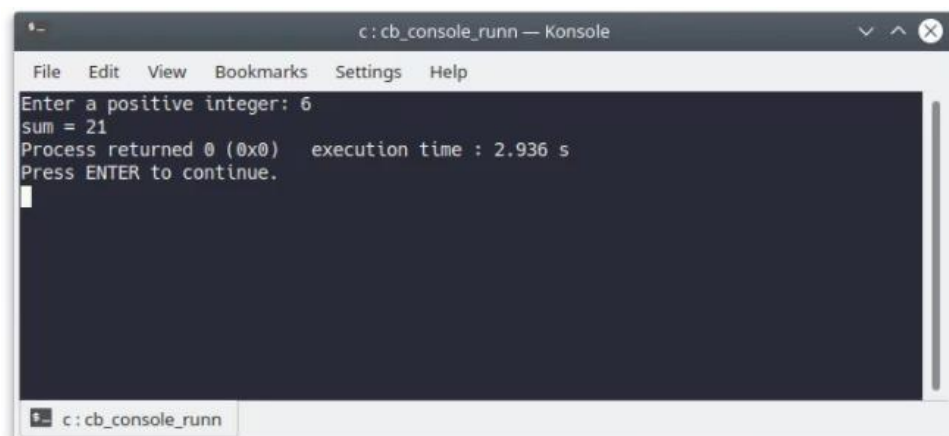
    printf("Enter a positive integer: ");
    scanf("%d", &number);

    result = sum(number);

    printf("sum = %d", result);
}

int sum(int num){
    if (num!=0)
        return num + sum(num-1); // fungsi sum() memanggil dirinya sen
    else
        return num;
}
```

Hasilnya:

A screenshot of a Windows console window titled "c : cb_console_runn — Konsole". The window has a menu bar with "File", "Edit", "View", "Bookmarks", "Settings", and "Help". The console output shows the program's execution: it prompts "Enter a positive integer: 6", displays "sum = 21", shows "Process returned 0 (0x0) execution time : 2.936 s", and ends with "Press ENTER to continue." The user has pressed the Enter key, indicated by a cursor on the next line. The taskbar at the bottom shows the active window as "c : cb_console_runn".

```
c : cb_console_runn — Konsole
File Edit View Bookmarks Settings Help
Enter a positive integer: 6
sum = 21
Process returned 0 (0x0) execution time : 2.936 s
Press ENTER to continue.

```


Mengapa hasilnya bisa 21?

Karena kita menginputkan nilai 6, maka akan sama dengan:

```
1 + 2 + 3 + 4 + 5 + 6 = 21
```

Variabel Lokal & Variabel Global

Variabel lokal dan variabel global akan sering kita temukan dalam pembuatan fungsi.

Variabel global adalah variabel yang bisa diakses dari semua fungsi. Sedangkan variabel lokal adalah variabel yang hanya bisa diakses dari dalam fungsi itu sendiri.

Contoh:

```
#include <stdio.h>

// membuat variabel global
int nilai = 9;

void main(){
    // membuat variabel lokal
    int nilai = 7;

    // mencetak variabel
    printf("Nilai: %d\n", nilai);
}
```

Pada contoh di atas, kita membuat variabel global bernama nilai.

Lalu di dalam fungsi main(), kita membuat variabel lagi bernama nilai dengan nilai yang berbeda.

Variabel yang ada di dalam fungsi main() adalah variabel lokal.

Lalu, berapakah hasil outputnya?

Jawabannya: 7

Mengapa bisa 7?

Karena variabel nilai kita buat ulang di dalam fungsi main.

Sekarang coba hapus variabel lokal yang ada di dalam main, sehingga akan menjadi seperti ini:

```
#include <stdio.h>

// membuat variabel global
int nilai = 9;

void main(){
    // mencetak variabel
    printf("Nilai: %d\n", nilai);
}
```

Maka hasil outputnya akan 9. Karena variabel yang dipakai adalah variabel global.

Pass by Value & Pass by Reference

Pass by value dan pass by reference adalah cara untuk memberikan nilai pada parameter. Biasanya kita langsung memberikan nilai kepada parameter dengan cara seperti ini:

```
kali_dua(4);
```

Ini disebut pass by value, karena di sana kita memberikan nilai 4.

kalau seperti ini:

```
kali_dua(&nama_variabel);
```

Ini disebut pass by reference, karena kita memberikan alamat memori.

Contoh:

```
#include <stdio.h>

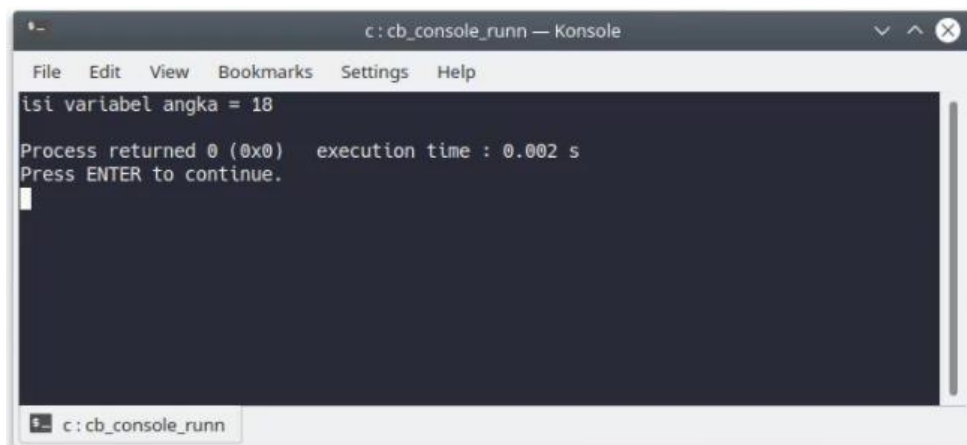
void kali_dua(int *num){
    *num = *num * 2;
}

void main(){
    int angka = 9;

    // memanggil fungsi
    kali_dua(&angka);

    // mencetak isi variabel
    // setelah fungsi dipanggil
    printf("isi variabel angka = %d\n", angka);
}
```

Hasilnya:



```
c:cb_console_runn — Konsole
File Edit View Bookmarks Settings Help
isi variabel angka = 18
Process returned 0 (0x0) execution time : 0.002 s
Press ENTER to continue.
```

Fungsi `kali_dua()` memiliki parameter berupa pointer, artinya kita harus memberikan alamat memori untuk parameter ini. Pada saat pemanggilan, fungsi `kali_dua()` kita isi parameternya dengan alamat memori dari variabel `angka`. Maka hasilnya nilai variabel `angka` akan dikalikan dengan 2 berdasarkan rumus pada fungsi yang kita berikan.

LATIHAN

1. Perkalian dua buah bilangan bulat positif dapat dilakukan dengan metode penjumlahan sebagai berikut: $12 \times 6 = 12 + 12 + 12 + 12 + 12 + 12$
Buatlah sebuah program menggunakan fungsi rekursif untuk menghitung hasil perkalian di atas.
2. Buatlah program yang menggunakan fungsi untuk perhitungan aritmetik yaitu penjumlahan, pengurangan, perkalian dan pembagian.
3. Buatlah program untuk menghitung luas dan keliling lingkaran dengan menggunakan fungsi. Fungsi yang harus dibuat yaitu:
 - a. luas() untuk menghitung luas lingkaran.
 - b. keliling() untuk menghitung keliling lingkaran.
4. Buatlah program untuk menghitung besarnya diskon yang diberikan atas besarnya jumlah pembelian, dengan ketentuan:
 - a. Jika belanja dibawah 1 jt, maka tidak mendapatkan diskon
 - b. Jika belanja diatas 1 jt dan dibawah 3 jt, maka mendapat diskon 20%
 - c. Jika belanja diatas 3 jt, maka mendapat diskon 35%

Fungsi yang harus ada adalah potong() untuk menghitung besar potongan yang akan diberikan. Berikut tampilan yang diinginkan pada output:

Program Hitung Potongan

Total Pembelian (Rp.)	: [diinputkan]
Besar Diskon	: [hasil proses]
Besar Yang Harus Dibayarkan	: [hasil proses]

REFERENCES

1. The C Programming Language. 2nd Edition
2. [https://en.wikipedia.org/wiki/C_\(programming_language\)](https://en.wikipedia.org/wiki/C_(programming_language))
3. https://en.wikipedia.org/wiki/Imperative_programming
4. <https://www.petanikode.com/tutorial/c/>
5. <https://www.learn-c.org/>
6. <https://www.tutorialspoint.com/cprogramming/index.htm>
7. <https://www.programiz.com/>
8. <https://www.dicoding.com/>
9. <https://data-flair.training/blogs/c-tutorials-home/>