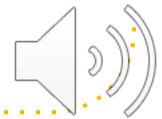
Sub Program: Function

ISA-105 - Algorithm & Pemrograman

Sofia Umaroh, S.Pd. M.T.
Sistem Informasi ITENAS

Jenis Parameter

- Penggunaan Parameter terbagi menjadi 2:
  - Parameter *pass by value*
  - Parameter *pass by reference*

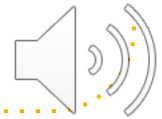


Parameter Pass by value

- Pada program di atas, ketika pemanggilan fungsi **exchange**, **nilai** variabel **x** dimasukkan ke parameter **a**. Sedangkan **nilai** variabel **y** dimasukkan ke parameter **b**.
    - void exchange (int a, int b)
    - exchange (x,y);
- Di dalam fungsi **exchange**, nilai variabel **a** dan **b ditukar**.
- Tapi penukaran tersebut **tidak berpengaruh** pada nilai **x** dan **y**. Yang notabene, yang dimasukkan ke parameter **a** dan **b** adalah variabel **x** dan **y**
- Ini disebabkan karena yang dimasukkan ke parameter **a** dan **b** Rdak lain hanyalah **NILAI** dari variabel **x** dan **y**, bukan **alamat variabel x dan y** itu sendiri

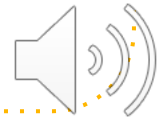
Parameter Pass by reference

- Perbedaan dengan program sebelumnya terletak pada tanda **ampersand** (&) yang terletak di depan parameter **a** dan **b**
- Pada program di atas, ketika pemanggilan fungsi **exchange**, **alamat** variabel **x** dimasukkan ke parameter **a**. Sedangkan **alamat** variabel **y** dimasukkan ke parameter **b**.
  - void exchange (int *a, int *b)
  -
  - exchange (x,y);
- Di dalam fungsi **exchange**, nilai variabel **a** dan **b** **ditukar**.



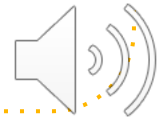
Parameter Pass by reference

- Ternyata penukaran nilai parameter **a** dan **b** yang terjadi di fungsi **exchange** tersebut berpengaruh terhadap nilai variabel **x** dan **y**.
- Ketika ditampilkan nilai **x** dan **y ikut tertukar** seperR halnya parameter **a** dan **b**
- Ini menunjukkan bahwa yang dimasukkan ke dalam parameter **a** dan **b** keRka pemanggilan fungsi **exchange** adalah **alamat x** dan **y**.
- Jadi ketika terjadi **perubahan** di argumen **a** dan **b**, maka isi variabel **x** dan **y juga berubah**.

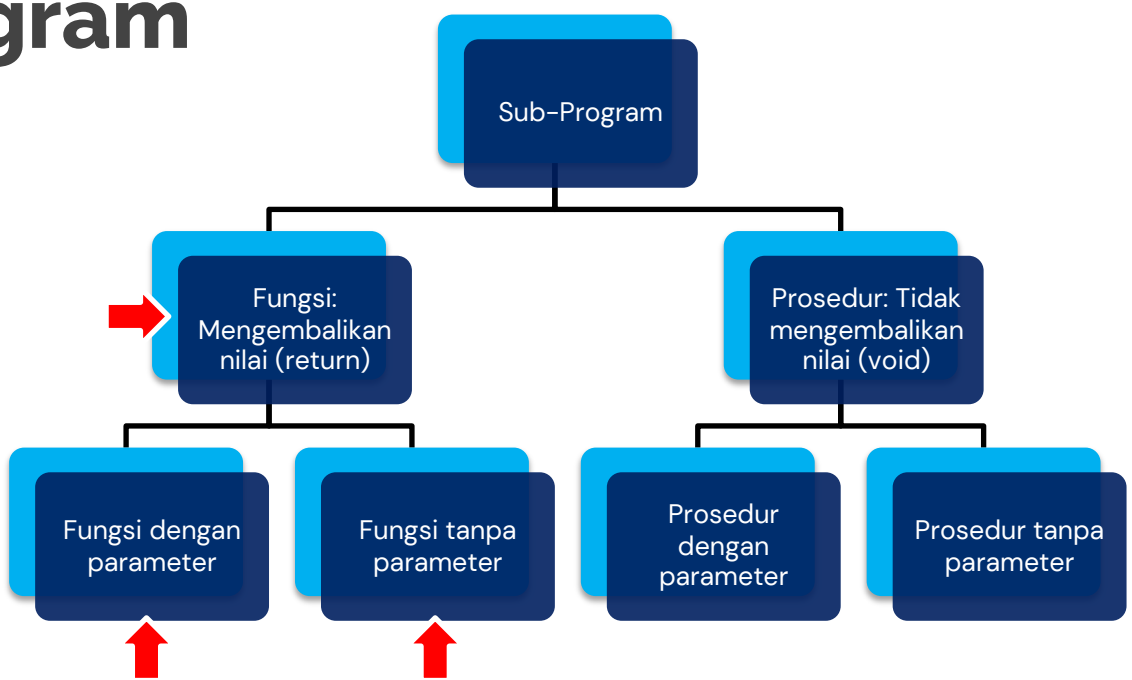


Parameter Pass by reference

- Pada konsep “Arguments Passed by Reference”, yang dilewatkan atau dimasukkan ke parameter fungsi adalah berupa **alamat dari variabel**
- Ketika terjadi perubahan terhadap parameter fungsi yang bersangkutan, **akan berpengaruh** terhadap nilai variabel yang dimasukkan ke dalam parameter

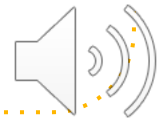


Peta Sub-Program



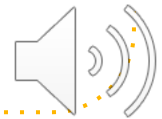
Jenis Sub-Program

- Sub-Program terbagi menjadi 2:
  - Prosedur, sub-program yang tidak mengembalikan nilai *(void)*
  - Fungsi, sub-program yang mengembalikan nilai *(return)*



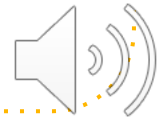
Definisi Fungsi

- Fungsi merupakan sekelompok statement (group of statement) dengan penamaan tertentu, yang akan dijalankan ketika ia dipanggil dari suatu bagian dalam program.
- Setiap kali fungsi dipanggil, maka terdapat nilai yang dihasilkan/dikemblaikan (return) ke program di mana fungsi dipanggil
- Dengan menggunakan fungsi, program bisa disusun secara lebih terstruktur (lebih modular) dan lebih efektif



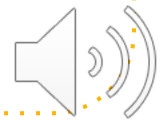
Variabel Global dan Lokal

- Variabel Lokal: variabel yang dideklarasikan dalam suatu fungsi, dan hanya bisa diakses atau dikenali dari dalam fungsi itu sendiri
- Variabel Global: variabel yang dideklarasikan di luar blok fungsi, dan bisa diakses atau dikenali dari fungsi manapun



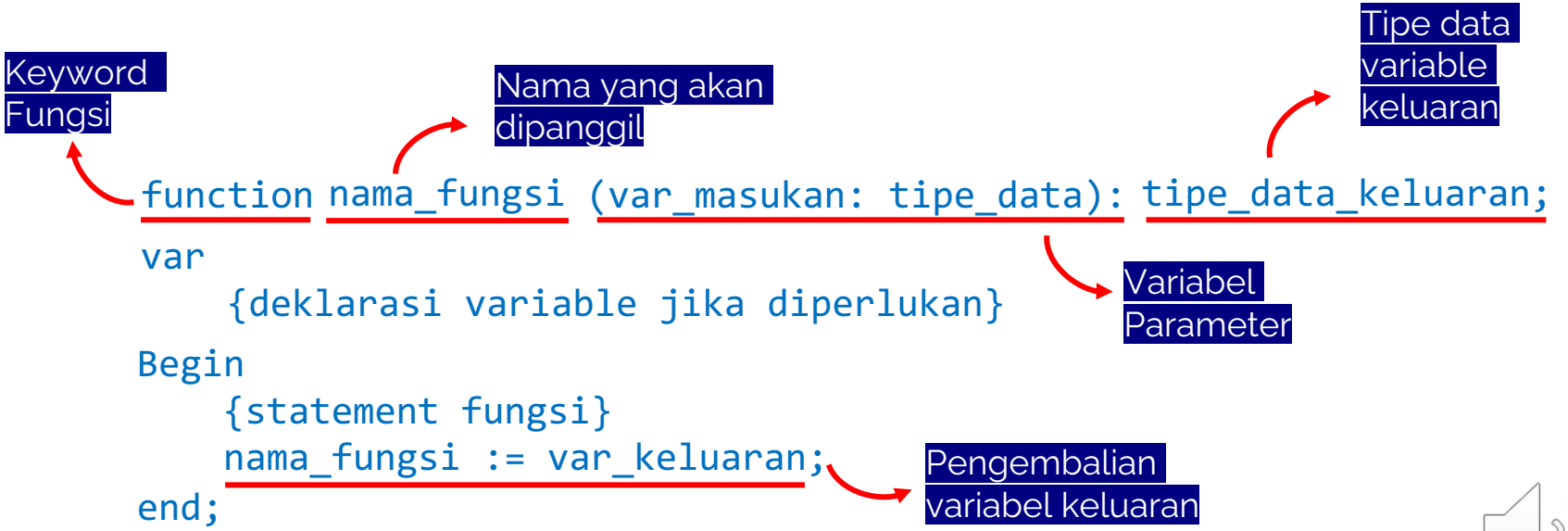
Deklarasi Fungsi

- Sebagaimana sebuah program, fungsi juga memiliki header dan block.
- Header fungsi terdiri atas:
  1. Kata kunci: `function` (Pascal), dan `tipe_data_keluaran` (bahasa C)
  2. `nama_fungsi`, nama yang akan dipanggil setiap kali fungsi akan digunakan.
  3. `tipe_data_keluaran`, INGAT! Fungsi adalah sub-program yang mengembalikan nilai, maka tentukan tipe data dari nilai yang akan dikembalikan
  4. `var_masukan` sebagai parameter fungsi ,jika diperlukan
- Jika `hasil` adalah variable keluaran, maka *hasil* dikembalikan dengan cara:
  - Pascal: `nama_fungsi := hasil;`
  - Bahasa C: `return hasil;`



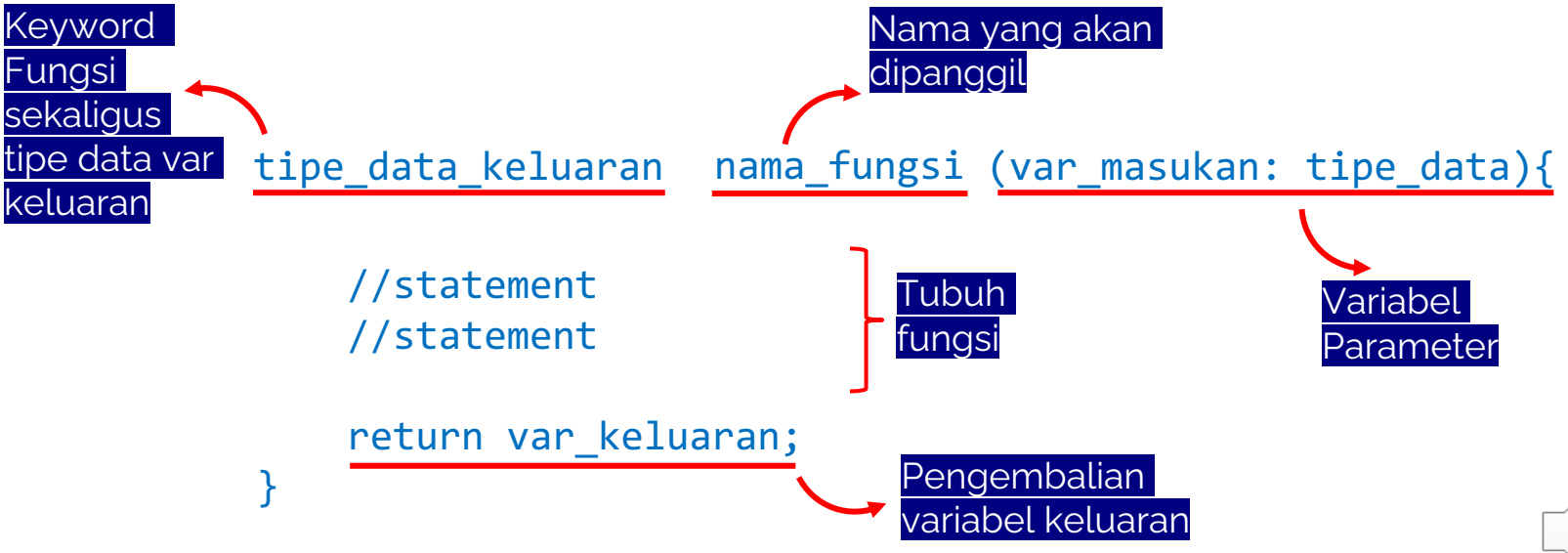
Deklarasi Fungsi

- Bentuk umum suatu fungsi dalam bahasa PASCAL:



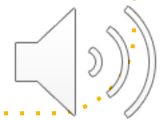
Deklarasi Fungsi

- Bentuk umum suatu fungsi dalam bahasa C:



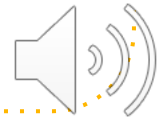
Parameter Fungsi

- Parameter fungsi digunakan untuk menerima masukan nilai dari luar fungsi, yang akan diolah dalam fungsi
- Kapan fungsi memerlukan parameter?
  - Ketika fungsi tsb membutuhkan data yang asalnya dari luar fungsi untuk diolah dalam fungsi
- Fungsi boleh tidak memiliki sama sekali parameter fungsi
- Jumlah parameter fungsi yang bisa dimiliki fungsi menyesuaikan kebutuhan, dan tidak ada batasan maksimalnya
- Pada saat deklarasi fungsi, penulisan parameter adalah dengan cara:
  - Bahasa C: `tipe_data nama_parameter`
  - Bahasa Pascal: `nama_parameter: tipe_data;`

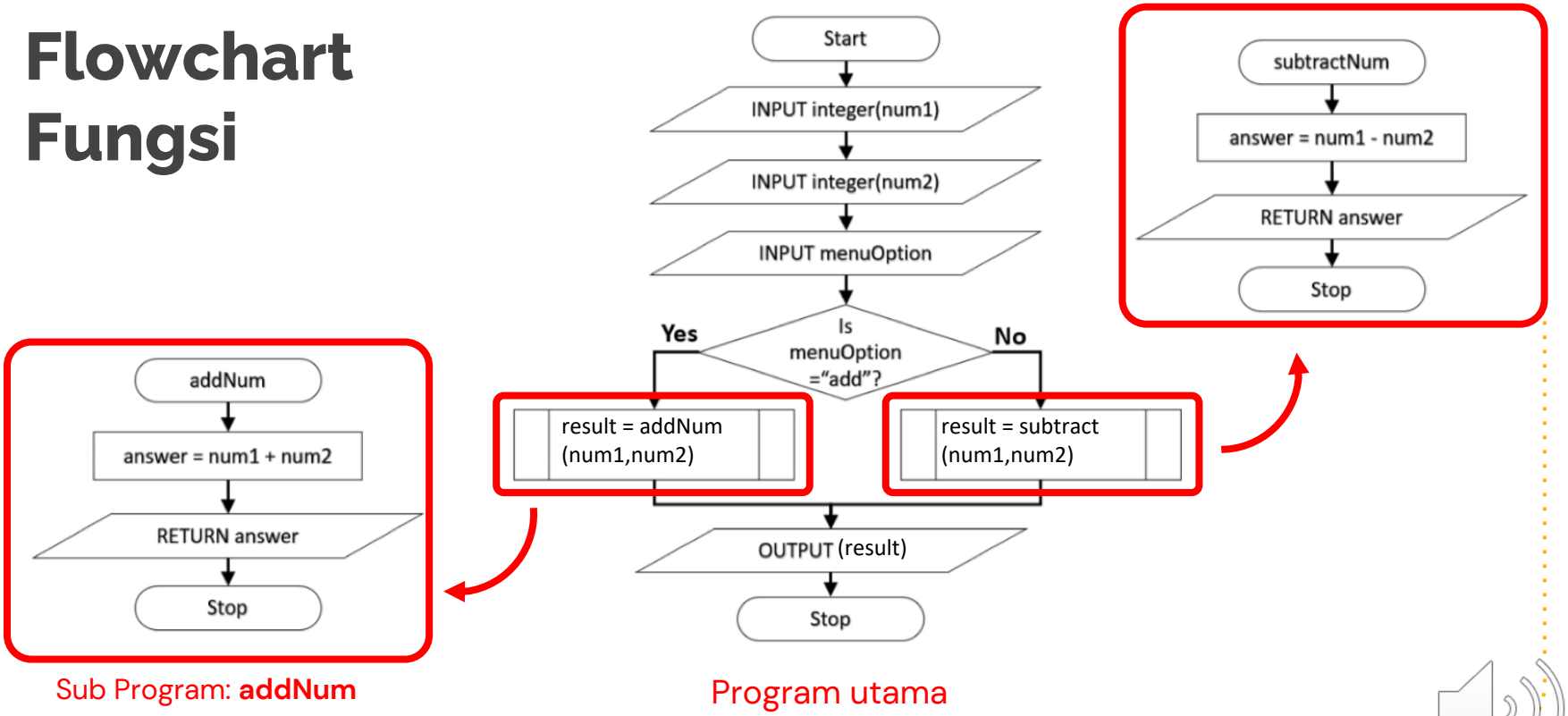


Memanggil Fungsi

- INGAT! Fungsi mengembalikan nilai, maka fungsi harus di-assign ke dalam variable atau di cetak menggunakan placeholder.
- Contoh memanggil fungsi:
  - **Fungsi dipanggil untuk disimpan nilainya ke dalam variabel hasil:**
    - **`hasil = addNum(num1, num2);`**
  - **Fungsi dipanggil untuk dicetak nilainya ke layar**
    - **`printf("Hasil: %d",addNum(num1, num2));`**



Flowchart Fungsi



Latihan: Ubah menjadi program

- Langkah 1: Implementasi program utama

```
int main(){
    int num1, num2, menuOption, answer;

    printf("Num1: "); scanf("%d",&num1);
    printf("Num2: "); scanf("%d",&num2);
    printf("Opsi(1/2): ");
    scanf("%d",&menuOption);
    if (menuOption == 1){
        answer = addNum(num1,num2);
    }else{
        answer = subtractNum(num1,num2);
    }
    printf("Hasil: %d",answer);
}
```

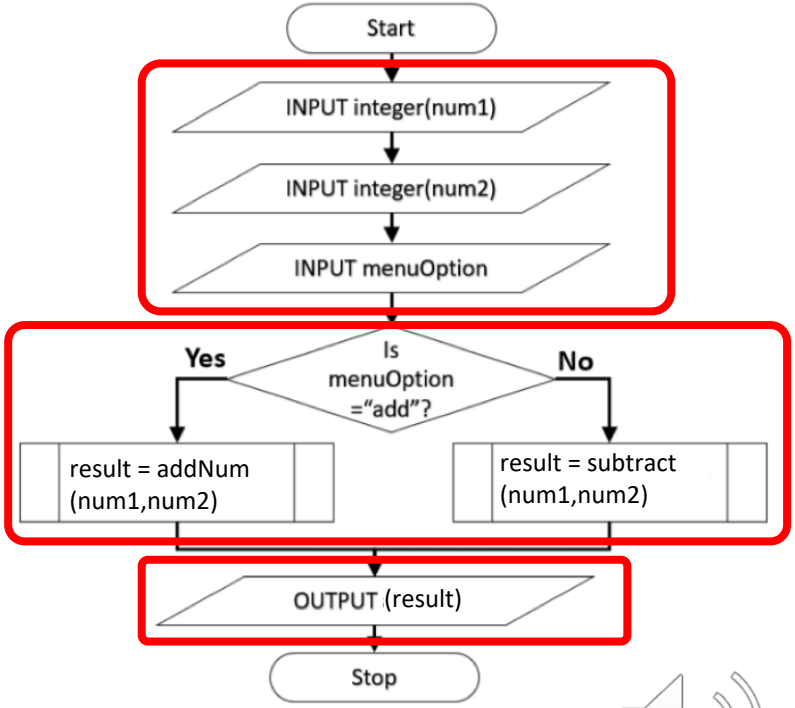
Deklarasi var

Input **num1, num2, menuOption**

Fungsi dipanggil untuk di-assign ke var **answer**

Kondisi jika **menuOption** = 1 atau tidak

Cetak hasil yang dikembalikan dari salah satu fungsi



Contoh Fungsi

- Langkah 2: Implementasi fungsi `addNum(num1, num2)`

```
int addNum(int num1, int num2);

int main(){
   …
}
int addNum(int num1, int num2){
    int answer;
    answer = num1 + num2;
    return answer;
}
```

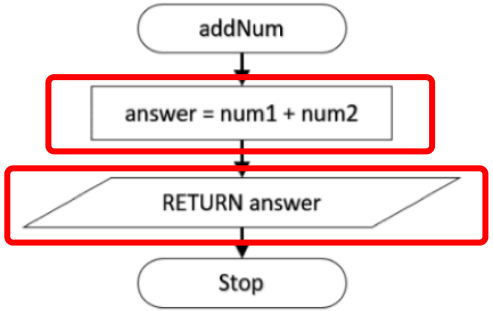
Deklarasi header fungsi dengan 2 parameter **num1** dan **num2**

Implementasi fungsi **addNum()**

Deklarasi var keluaran **answer**

Proses penjumlahan var parameter masukan

Mengembalikan nilai **answer** ke program pemanggil untuk di-assign ke variable **result**



Contoh Fungsi

- Langkah 3: Implementasi fungsi `subtractNum(num1, num2)`

Deklarasi header fungsi dengan 2 parameter **num1** dan **num2**

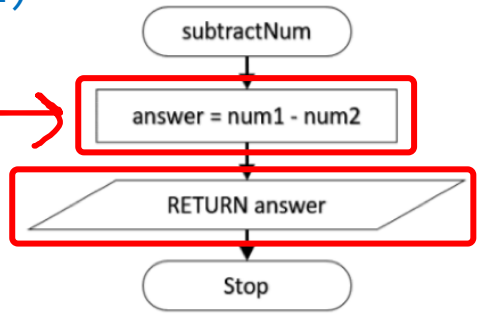
```
int subtractNum(int num1, int num2);

int main(){
   …
}
int subtractNum(int num1, int num2){
    int answer;
    answer = num1 - num2;
    return answer;
}
```

Implementasi fungsi **addNum()**

Deklarasi var keluaran **answer**

Proses penjumlahan var parameter masukan

Mengembalikan nilai **answer** ke program pemanggil untuk di-assign ke variable **result**



Contoh Fungsi

- Diperoleh kode program lengkapnya:

```
1. #include <stdio.h>
2.  // header fungsi
3. int addNum(int num1, int num2);
4. int subtractNum(int num1, int num2);
5.
6. // program utama
7. int main(){
8.    int num1, num2, menuOption, answer;
9.    printf("Num1: "); scanf("%d",&num1);
10.   printf("Num12 "); scanf("%d",&num2);
11.   printf("Opsi(1/2: ");
12.   scanf("%d",&menuOption);
13.   if (menuOption == 1){
14.      answer = addNum(num1,num2);
15.   }else{
16.      answer = subtractNum(num1,num2);
17.   }
18.   printf("Hasil: %d",answer);
19.   return 0;
20.}
```

```
21.// implementasi fungsi
22.int addNum(int num1, int num2){
23.   int answer;
24.   answer = num1 + num2;
25.   return answer;
26.}
27.
28.int subtractNum(int num1, int num2){
29.   int answer;
30.   answer = num1 - num2;
31.   return answer;
32.}
33.
```

```
Num1: 3
Num2: 4
Opsi(1/2): 1
Hasil: 7

...Program finished with exit code 0
Press ENTER to exit console.
```