



# Bab 13. Pointer 3

Konsep Pemrograman  
Politeknik Elektronika Negeri Surabaya  
2017



# Overview

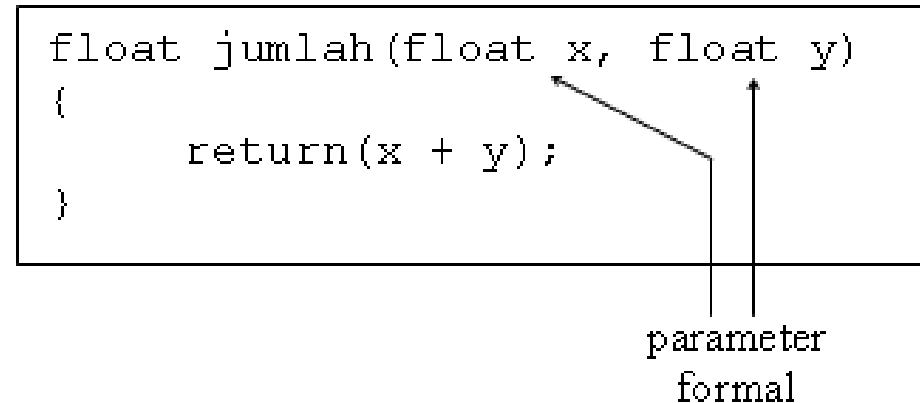
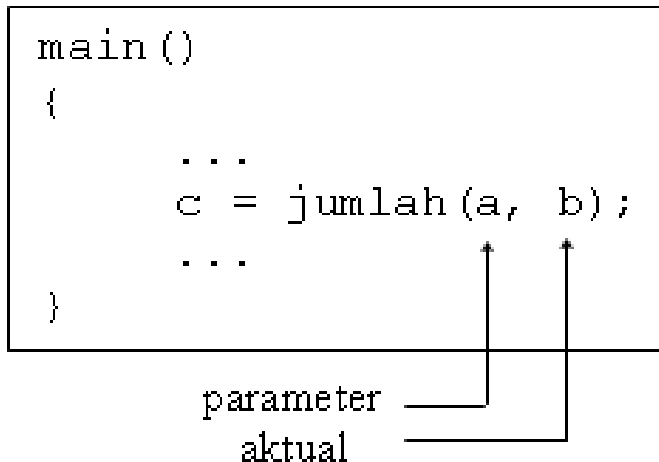
- Pointer dalam Fungsi
  1. Pointer Sebagai Parameter Fungsi
    - Parameter Formal dan Parameter Aktual
    - Cara Melewatkan Parameter dalam Fungsi
      - *Pass by Value*
      - *Pass by Reference* (Pointer Sebagai Parameter Fungsi)
  2. Pointer Sebagai Keluaran Fungsi (*return value*)



# Parameter Formal dan Parameter Aktual

- Parameter formal adalah variabel yang ada pada daftar parameter dalam definisi fungsi.
- Parameter aktual adalah parameter (tidak selalu berupa variabel) yang dipakai dalam pemanggilan fungsi.

# Parameter Formal dan Parameter Aktual



- Pada contoh program di atas misalnya, maka dalam fungsi `jumlah()` variabel `x` dan `y` dinamakan sebagai parameter formal, sedangkan variabel `a` dan `b` adalah parameter aktual



# Pengiriman Parameter dalam Fungsi

- Ada dua cara untuk melewatkan parameter ke dalam fungsi, yaitu berupa :
  - Pengiriman berupa nilai/value (*pass by value*)
    - semua contoh fungsi yang telah dibahas sebelumnya (pada bab fungsi)
  - Pengiriman berupa address/referensi (*pass by reference*) → **WHEN?**



# Pengiriman Parameter dalam Fungsi

- PASS BY VALUE
  - yang dikirim sebagai parameter aktual adalah value/nilainya
  - parameter aktual akan dicopy oleh parameter formal
  - perubahan apapun pada parameter formal tidak akan berpengaruh kepada parameter aktual
    - perubahan di dalam fungsi tidak bisa terbaca di tempat fungsi tsb dipanggil
- PASS BY REFERENCE
  - yang dikirim sebagai parameter aktual adalah address/alamat dari sebuah value
  - yang menerima kiriman tsb atau yang menjadi parameter formal adalah variabel POINTER (→ variabel yang khusus untuk menampung address)
  - perubahan di dalam fungsi bisa terbaca kembali di tempat fungsi tsb dipanggil



# Pass by Value

```
#include <stdio.h>
void tukar (int, int);
main(){
    int a = 5, b = 2;

    printf("Nilai sebelum pemanggilan fungsi: a = %d; b = %d\n",a,b);
    tukar(a,b);
    printf("\nNilai sesudah pemanggilan fungsi: a = %d; b = %d\n",a,b);
}

void tukar(int x, int y){
    int z;

    printf("\nNilai di awal fungsi tukar(): x = %d; y = %d\n", x, y);
    z = x;
    x = y;
    y = z;
    printf("\nNilai di akhir fungsi tukar(): x = %d; y = %d\n", x, y);
}
```

```
C:\ "G:\Kampus\Programming 2\Praktikum\pointer\baru\... - □ ×
Nilai sebelum pemanggilan fungsi: a = 5; b = 2
Nilai di awal fungsi tukar(): x = 5; y = 2
Nilai di akhir fungsi tukar(): x = 2; y = 5
Nilai setelah pemanggilan fungsi: a = 5; b = 2
Press any key to continue_
```



# Pass by Reference

```
#include <stdio.h>
void tukar (int *, int *);
main() {
    int a = 5, b = 2;
```

```
    printf("Nilai sebelum pemanggilan fungsi: a = %d; b = %d\n", a, b);
    tukar(&a,&b);
    printf("\nNilai setelah pemanggilan fungsi: a = %d; b = %d\n",a,b);
}
```

```
void tukar(int *x, int *y){
    int z;

    printf("\nNilai di awal fungsi tukar(): *x = %d; *y = %d\n",*x,*y);
    z = *x;
    *x = *y;
    *y = z;
    printf("Nilai di akhir fungsi tukar(): *x = %d; *y = %d\n",*x,*y);
}
```

```
C:\ "G:\Kampus\Programming 2\Praktikum\pointer\baru\... - □ ×
Nilai sebelum pemanggilan fungsi: a = 5; b = 2
Nilai di awal fungsi tukar(): *x = 5; *y = 2
Nilai di akhir fungsi tukar(): *x = 2; *y = 5
Nilai setelah pemanggilan fungsi: a = 2; b = 5
Press any key to continue
```





# Pointer Sebagai *return value* Fungsi

- Suatu fungsi dapat dibuat agar *return value*-nya berupa pointer.
- Misalnya, suatu fungsi menghasilkan *return value* berupa pointer yang menunjuk ke string nama-nama bulan
  - Di `main()` user diminta memasukkan bulan ke berapa → variabel `bln` bertipe *int*
  - Dalam fungsi `nama_bulan()` kode `int tsb` akan diubah ke dalam nama stringnya



# Pointer Sebagai *return value* Fungsi

```
#include <stdio.h>

char *nama_bulan(int n);

main()
{
    int bln;
    char *pkar;

    printf("Masukkan Bulan 1..12 : ");
    scanf("%d", &bln);
    pkar = nama_bulan(bln);
    printf("Bulan ke-%d adalah %s\n", bln, pkar);
}
```



# Pointer Sebagai *return value* Fungsi

```

char *nama_bulan(int n){
    char *month[] = {
        "NGAWUR",
        "Januari",
        "Februari",
        "Maret",
        "April",
        "Mei",
        "Juni",
        "Juli",
        "Agustus",
        "September",
        "Oktober",
        "November",
        "Desember"
    };

    return ((n<1 || n>12) ? month[0] : month[n]);
}

```

Three screenshots of a Windows command prompt window showing the execution of a C program. The first screenshot shows the user entering '4' and the program outputting 'Bulan ke-4 adalah April'. The second screenshot shows the user entering '13' and the program outputting 'Bulan ke-13 adalah Kode bulan salah'. The third screenshot shows the user entering '0' and the program outputting 'Bulan ke-0 adalah Kode bulan salah'.



# Pointer Sebagai *return value* Fungsi

- Pada definisi fungsi di atas,

```
char *nama_bulan()
```

menyatakan bahwa *return value* dari fungsi **nama\_bulan()** berupa *pointer to char* (pointer yang menunjuk ke obyek `char` atau string).

- Dalam fungsi `nama_bulan()`, mula-mula array bernama `month[]` dideklarasikan dan sekaligus diinisialisasi agar elemen-elemennya yang berupa pointer menunjuk ke sejumlah string yang menyatakan nama bulan.
- Di bagian akhir fungsi, pernyataan

```
return ( (n<1 || n>12) ? month[0] : month[n] );
```

menyatakan bahwa hasil fungsi (*return value* = *RV*) berupa pointer sbb:

- jika masukan fungsi `n<1` atau `n>12` →  $RV = month[0]$ , yaitu array yang berisi address/pointer yang menunjuk ke string “NGAWUR”
- jika masukan fungsi berupa `n` yang terletak antara 1 sampai dengan 12 →  $RV = bulan[n]$ , yaitu array yang berisi address/pointer yang menunjuk ke salah satu string “Januari” s/d “Desember”



# Latihan

Untuk semua contoh program yang ada pada teori Pointer 3 :

1. Gambarlah ilustrasi alokasi memori dari setiap baris pernyataan yang diproses
2. Perkirakan hasil eksekusinya



# Latihan

## 1. Untuk program di bawah ini,

- gambarkan ilustrasi alokasi memori dari setiap baris pernyataan yang diproses
- perkirakan hasil eksekusinya

```
#include <stdio.h>
void naikkan_nilai(int *x, int *y);
main() {
    int a = 3, b = 7;

    printf("SEMULA : a = %d b = %d\n", a, b);
    naikkan_nilai(&a, &b);
    printf("KINI    : a = %d b = %d\n", a, b);
}

void naikkan_nilai(int *x, int *y){
    *x = *x + 2;
    *y = *y + 2;
}
```



# Latihan

2. Buatlah sebuah program dengan mendefinisikan sebuah fungsi `rotasi()` yang menerima tiga parameter berupa variabel `a`, `b`, dan `c`. Fungsi ini melakukan rotasi sehingga nilai `a` berpindah ke `b`, `b` ke `c` dan nilai `c` ke `a` sekembalinya ke fungsi `main()`.