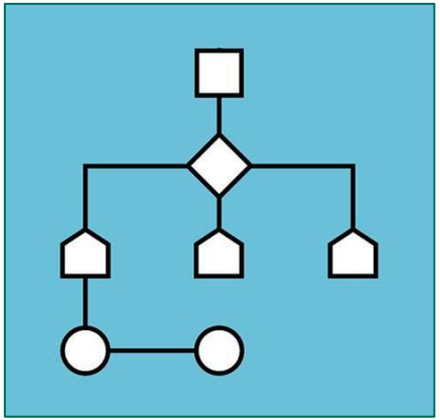
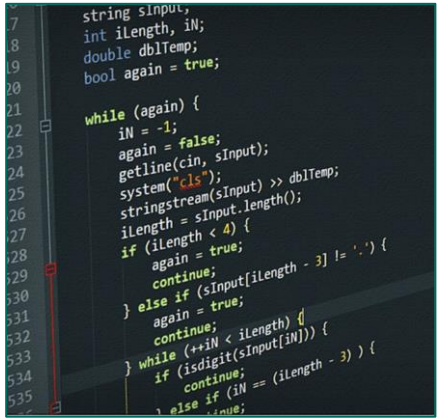
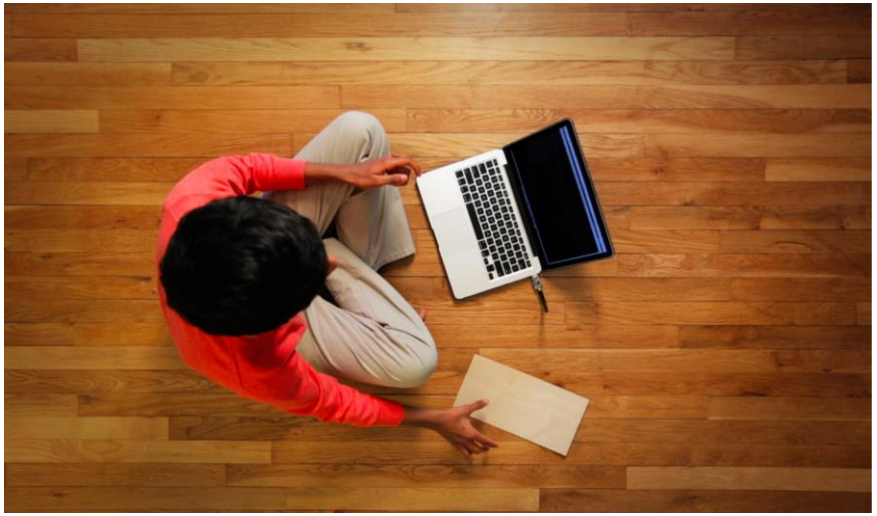
IPB University
Bogor Indonesia

KOM120B #10

Pointer dan Fungsi







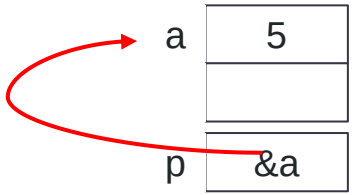
Tim Pengajar

KOM120B – Algoritme dan Dasar Pemrograman

Departemen Ilmu Komputer - FMIPA

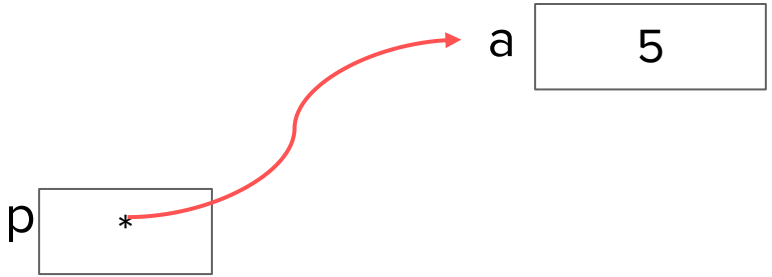
Pointer

- Pointer adalah variabel yang berisi alamat memori sebagai nilainya, berbeda dengan variabel biasa yang berisi nilai tertentu.
- Berisi alamat dari variabel yang mempunyai nilai tertentu.
- Dua jenis variabel:
  - secara langsung menunjuk ke suatu nilai tertentu, misalnya: `int a=5;`
  - secara tidak langsung (merupakan variabel pointer), menunjuk ke alamat dari suatu nilai, mis: `int *p=&a;`
- Format penulisan alamat : "%p"



Variabel Pointer

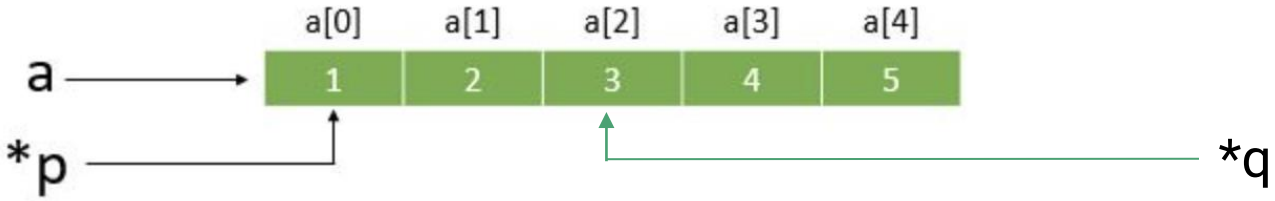
```
int a=5;
int *p;
p=&a;       // p berisi alamat dari a
            // & adalah operator referensi

printf("%d %d %p %p\n", a, *p, p, &a);
```



Array dan Pointer

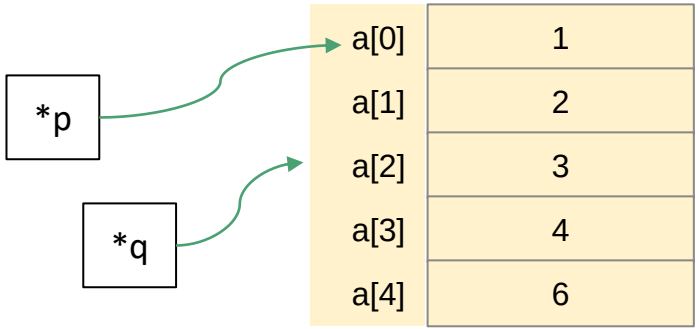
- Dalam C, array dapat dinyatakan sebagai pointer ke elemen pertama.
- Jadi alamat dari `a[i]` dapat ditunjuk oleh suatu variabel pointer.
- Contoh:

```
int *p, *q;
int a[5]={1,2,3,4,5};
p = a;      // atau p = &a[0] → p menunjuk ke elemen pertama dari array a
q = &a[2]; // q menunjuk ke elemen ketiga dari array a
```



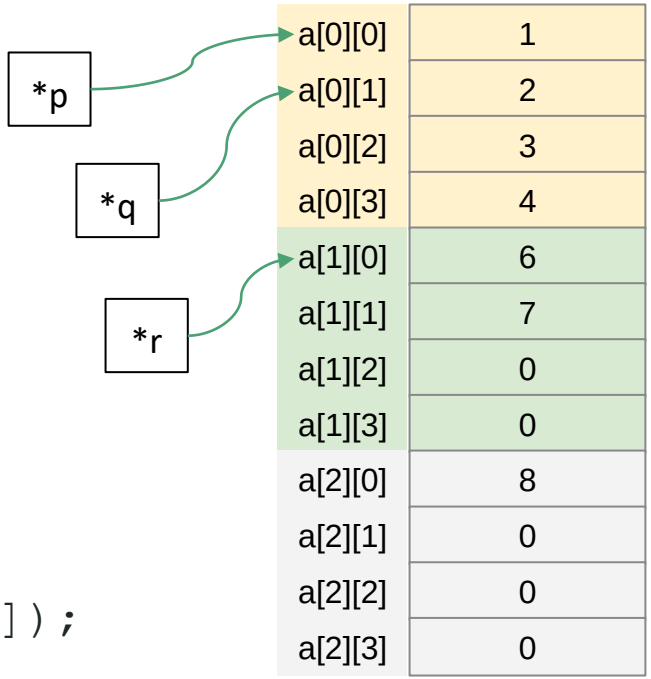
Apa output program berikut?

```
#include <stdio.h>
int main()
{
    int *p, *q;
    int a[5]={1,2,3,4,5};
    p=&a[0];  q=&a[2];
    printf("%d %d %d\n", *p, p[0], a[0]);
    printf("%d %d %d\n", *(p+1), p[1], a[1]);
    printf("%d %d\n", *q, q[1]);
    return 0;
}
```



Apa output program berikut?

```
#include <stdio.h>
int main()
{
    int *p, *q, *r;
    int a[3][4]={{1,2,3,4},{6,7},{8}};
    p=a[0];  q=&a[0][1];  r=a[1];
    printf("%d %d %d\n", *p, *q, *r);
    printf("%d %d %d\n", *(p+1), q[2], r[1]);
    printf("%d %d\n", *(q+2), *(r+1));
    return 0;
}
```



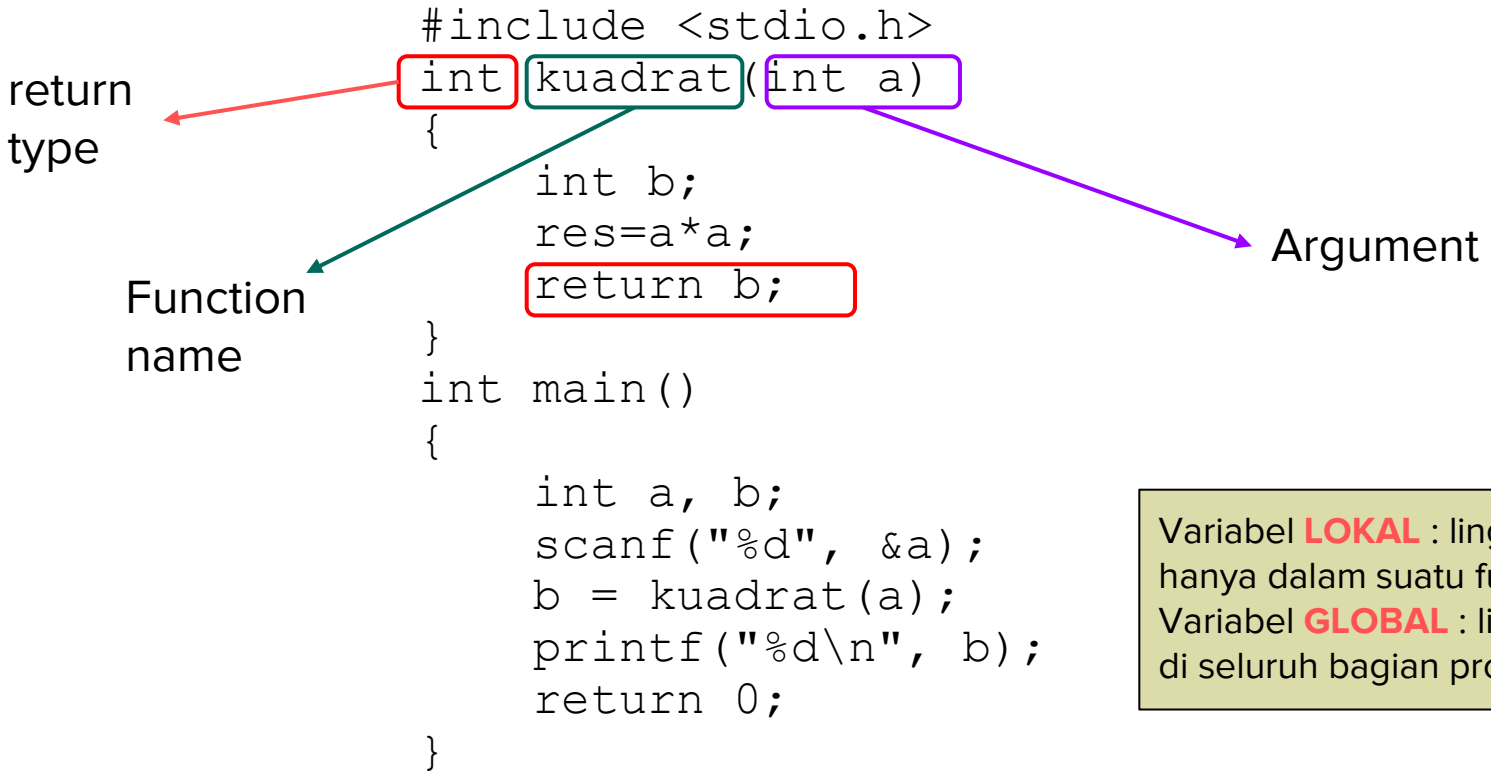
Fungsi

- Fungsi dalam pemrograman adalah blok instruksi yang dapat dieksekusi berulang-ulang sesuai kebutuhan.
- Implementasi dari konsep modularitas → dekomposisi
- 2 tipe fungsi:
  - Built-in (Library) Functions : disediakan oleh sistem dan tersimpan dalam pustaka (library) sehingga siap digunakan. Contoh: printf(), strcpy(), dsb
  - User Defined Functions : dibuat oleh user pada saat menulis program

Struktur Fungsi

- Sebuah fungsi terdiri dari empat bagian, yaitu:
  - tipe kembali (return type),
  - nama fungsi (function name),
  - daftar argumen (argument list), dan
  - tubuh fungsi (function body).
- Tiga bagian pertama membentuk apa yang disebut dengan fungsi prototipe (prototype function).
- Daftar argumen dibatasi oleh sepasang kurung biasa (...), yang dapat terdiri dari nol atau lebih argumen yang masing-masing dipisahkan oleh tanda koma (,).
- Tubuh fungsi dibatasi oleh sepasang kurung kurawal { ... } dan terdiri dari baris-baris pernyataan.

Contoh : Fungsi Kuadrat Bilangan Bulat



Variabel **LOKAL** : lingkupnya hanya dalam suatu fungsi.
Variabel **GLOBAL** : lingkupnya di seluruh bagian program

Apa output program berikut?

```
#include <stdio.h>
void swap(int a, int b)
{
    int t=a;
    a=b;
    b=t;
}
int main()
{
    int a=5, b=10;
    swap(a,b);
    printf("a=%d, b=%d\n", a, b);
    return 0;
}
```

Call by Value
versus
Call by Reference

Call by Reference

```
#include <stdio.h>
void swap(int *a, int *b)
{
    int t=*a;
    *a=*b;
    *b=t;
}
int main()
{
    int a=5, b=10;
    swap(&a,&b);
    printf("a=%d, b=%d\n", a, b);
    return 0;
}
```

Call by Value
versus
Call by Reference

Yang dikirim adalah alamat memori (referensi), dan diterima oleh variabel pointer

Array sebagai Argumen Fungsi

- Seperti variabel lainnya, array juga dapat digunakan sebagai argumen dari fungsi.
- Nilai variabel array yang dideklarasikan sebagai argumen fungsi akan mempengaruhi nilai variabel array pada fungsi pemanggilnya (*call by reference*).
- Contoh deklarasi array dalam fungsi:

```
int printarray(int n, int a[n]) {
    ...
}
```

- Perhatikan bahwa argumen `n` harus disebut sebelum `a[n]`

Contoh Argumen Fungsi Array

```
#include <stdio.h>

// Fungsi menuliskan elemen array berukuran n
void printarray(int n, int a[n]) {
    int i;
    for (i=0; i<n; i++)
        printf("%d\n", a[i]);
}

int main() {
    int x[5] = {1,4,2,-3,9};
    printarray(5,x);
    return 0;
}
```

Latihan 1 : Fungsi Membalik Array 1D

Buat fungsi untuk membalik array 1D, dan implementasikan dalam driver untuk problem membalik data seperti pada contoh. Ukuran maksimum data: 1000.

Contoh Input:

```
5
1 0 8 6 1
```

Contoh Output:

```
1 6 8 0 1
```

Latihan 2 : Flip Horisontal

Dengan menggunakan fungsi membalik array 1D, buat program untuk menampilkan matrik setelah dilakukan flip secara horisontal. Ukuran maksimum: 100x100.

Contoh Input:

```
4 5
1 0 8 6 1
0 0 0 0 2
0 0 0 3 3
4 5 1 2 4
```

Contoh Output:

```
1 6 8 0 1
2 0 0 0 0
3 3 0 0 0
4 2 1 5 4
```

Latihan 3 : Prefix Sum Array 1D

Diberikan sebuah array arr berukuran n. Prefix sum dari array arr adalah array lain (misalkan prefixSum) yang berukuran sama, dimana prefixSum[i] = arr[0] + arr[1] + arr[2] ... arr[i]. Buatlah program untuk membuat prefix sum tersebut.

Contoh Input:

```
5
10 20 10 5 15
```

Contoh Output:

```
10 30 40 45 60
```

Latihan 4 : Prefix Sum Array 2D

Buatlah program untuk membuat prefix sum dari suatu array 2D yang telah diketahui.

Contoh Input:

```
3 3
10 20 30
 5 10 20
 2  4  6
```

Contoh Output:

```
10 30 60
15 45 95
17 51 107
```