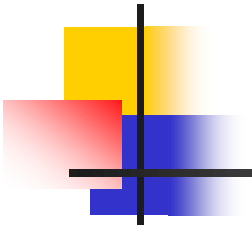


POINTER II

Oleh :
Mike Yuliana
PENS

SASARAN

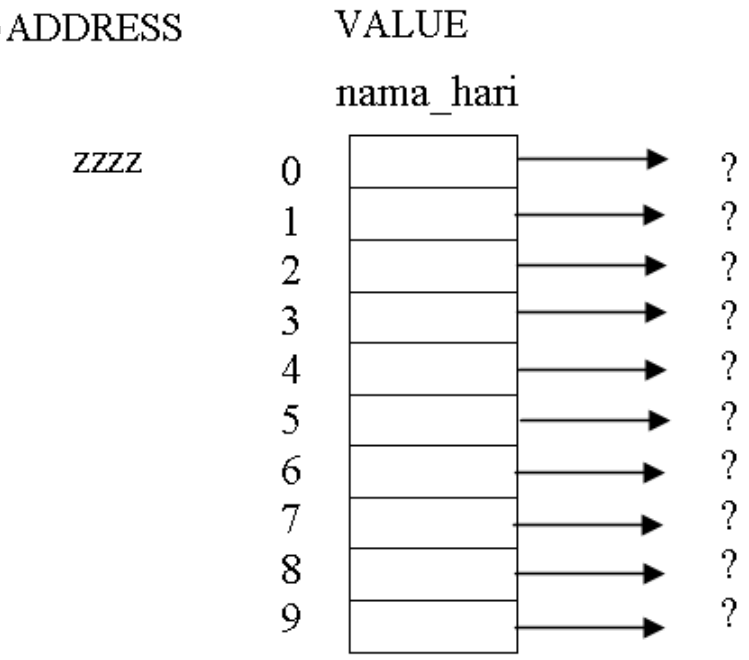
- Menjelaskan tentang array dari pointer
- Menjelaskan tentang pointer menunjuk pointer
- Menjelaskan tentang pointer dalam fungsi

Array of Pointer

- Suatu array bisa digunakan untuk menyimpan sejumlah pointer.
- Jika dideklarasikan :

```
char *nama_hari[10];
```

merupakan pernyataan untuk mendeklarasikan *array of pointer to char*.



- Array `nama_hari` terdiri dari 10 elemen berupa pointer yang menunjuk ke data bertipe *char*.

Array of Pointer

- Array pointer bisa diinisialisasi sewaktu pendeklarasian.
- Jika dideklarasikan:

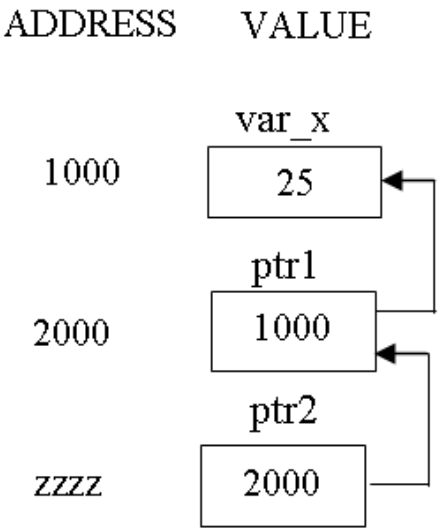
```
char *namahari[] =
      {“Senin”,
      “Selasa”,
      “Rabu”,
      “Kamis”,
      “Jumat”,
      “Sabtu”,
      “Minggu”};
```

- Pada contoh ini :
  - namahari[0] berisi alamat/pointer yang menunjuk ke string “Senin”.
  - namahari[1] berisi alamat/pointer yang menunjuk ke string “Selasa”.
  - namahari[2] berisi alamat/pointer yang menunjuk ke string “Rabu”.
  - dan seterusnya

Contoh #1

```
#include <stdio.h>
main()
{
   int i;
   char hari[7][7]={"senin","selasa","rabu","kamis", "jumat",
   "sabtu", "minggu"};
   char *phari[7];
   for(i=0; i<7;i++)
   phari[i]= hari[i];
   for(i=0; i<7;i++)
   printf("%s", phari[i]);
   }
```

Pointer to Pointer



- Suatu pointer bisa menunjuk ke pointer yang lain
- Jika dideklarasikan :

```
int var_x = 25, *ptr1, **ptr2;
```

  - `var_x` adalah variabel bertipe *int*.
  - `ptr1` adalah variabel bertipe *pointer to int* → pointer yang menunjuk ke sebuah data bertipe *int*
  - `ptr2` adalah variabel bertipe *pointer to pointer to int* → pointer yang menunjuk ke *pointer to int* (itulah sebabnya deklarasinya berupa `int **ptr2;`)
- Agar `ptr1` menunjuk ke variabel `var_x` dan `ptr2` menunjuk ke `ptr1`, instruksinya sbb :

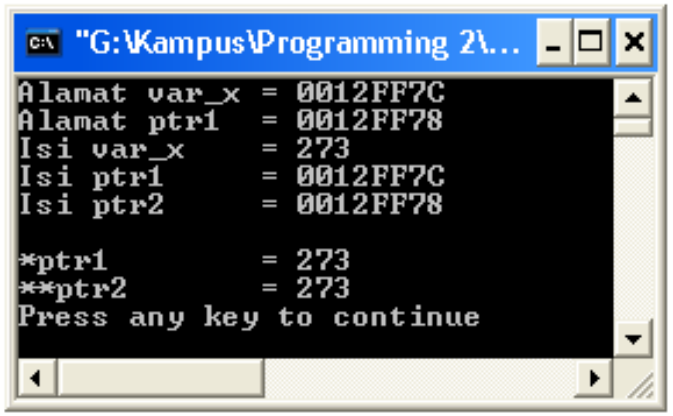
```
ptr1 = &var_x;
ptr2 = &ptr1;
```

Contoh #2

```
#include <stdio.h>
main(){
   int var_x = 273, *ptr1, **ptr2;

   ptr1 = &var_x;
   ptr2 = &ptr1;

   printf("Alamat var_x = %p\n", &var_x);
   printf("Alamat ptr1  = %p\n", &ptr1);
   printf("Isi var_x    = %d\n", var_x);
   printf("Isi ptr1     = %p\n", ptr1);
   printf("Isi ptr2     = %p\n", ptr2);
   printf("\n*ptr1       = %d\n", *ptr1);
   printf("**ptr2       = %d\n", **ptr2);
}
```



Pointer dan Fungsi

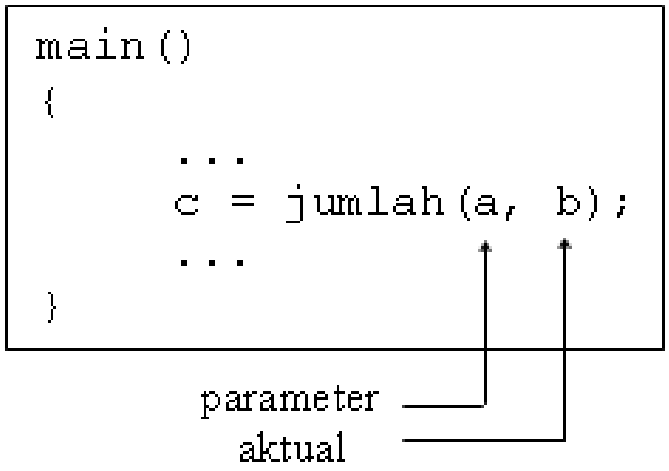
Pointer dalam Fungsi

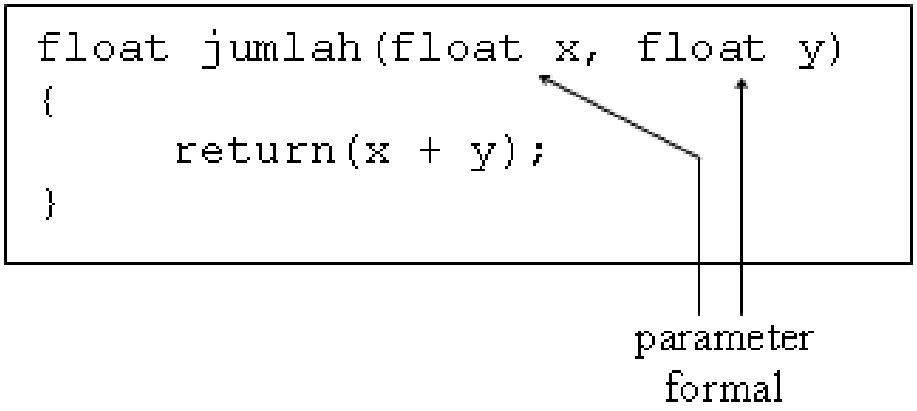
1. Pointer Sebagai Parameter Fungsi
   - Pointer sebagai parameter dalam Fungsi
   - Parameter Formal dan Parameter Aktual
   - Cara Melewatkan Parameter dalam Fungsi
     - *Pass by Value*
     - *Pass by pointer* (Pointer Sebagai Parameter Fungsi)
2. Pointer Sebagai Keluaran Fungsi *(return value)*

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)*

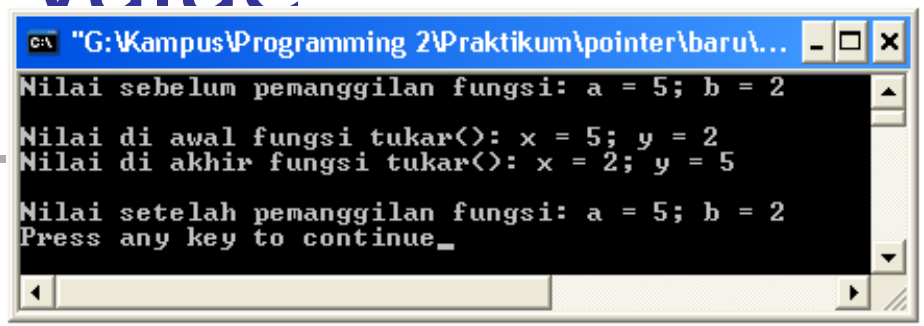
    → semua contoh fungsi yang telah dibahas sebelumnya (pada bab fungsi)
  - Pengiriman berupa address/alamat *(pass by pointer)*

Pengiriman Parameter dalam Fungsi

- PASS BY VALUE
  - yang dikirim sebagai paramenter 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 POINTER
  - 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



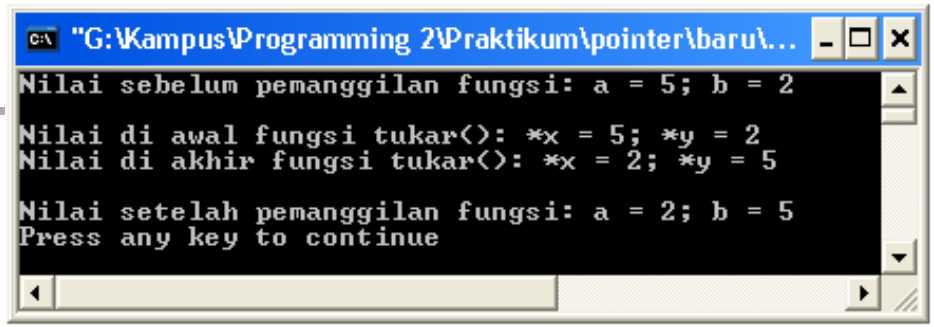

```
#include <stdio.h>
void tukar (int, int);
main(){
   int a = 5, b = 2;

   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);
}
```

Pass by Pointer

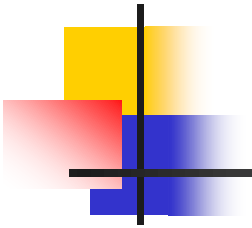



```
#include <stdio.h>
void tukar (int *, int *);
main(){
   int a = 5, b = 2;

   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);
}
```



LATIHAN SOAL

1. Buat program untuk menampilkan `int nomor[3]={10,20,30}` dengan menggunakan array pointer!
2. Buat program dengan menggunakan fungsi! Serta gambarkan ilustrasi alokasi memori dari setiap baris pernyataan!

```
#include <stdio.h>
main()
{
int a = 3;
int b = 7;
int *x, *y;
x=&a;
y=&b;
printf("SEMULA : a = %d  b = %d\n", a, b);
*x = *x + 2;
*y = *y + 2;
printf("KINI   : a = %d  b = %d\n", a, b);
}
```

3. Gambarkan ilustrasi alokasi memori dari setiap baris pernyataan serta perkirakan hasil eksekusinya!

```
int r, q = 7;
int go_crazy(int *, int *);
main()
{
int *ptr1 = &q;
int *ptr2 = &q;
r = go_crazy(ptr1, ptr2);
printf("q=%d, r=%d, *ptr1=%d, *ptr2=%d\n",q,r,*ptr1,*ptr2);

ptr2 = &r;

go_crazy(ptr2, ptr1);
printf("q=%d, r=%d, *ptr1=%d, *ptr2=%d\n",q,r,*ptr1,*ptr2);
}

int go_crazy(int *p1, int *p2)
{
int x = 5;
r = 12;
*p2 = *p1 * 2;
p1 = &x;
return *p1 * 3;
}
```