Pengenalan Struktur Data

Farah Zakiyah Rahmanti

2014

Definisi

- Skema organisasi, seperti struktur dan array, yang diterapkan pada data sehingga data dapat diinterprestasikan dan sehingga operas-operasi spesifik dapat dilaksanakan pada data tersebut.

Latar Belakang

- Tipe data cenderung menggunakan ukuran yang tetap.
- Pada beberapa kasus, ukuran dari sebuah obyek tidak bisa dipastikan sampai dengan waktu dieksekusi (run time).
- Alokasi memori (memory allocation) menyediakan fasilitas untuk membuat ukuran buffer dan array secara dinamik.
- Dinamik artinya bahwa ruang dalam memori akan dialokasikan ketika program dieksekusi.

Pointer

- Suatu variable yang dipakai untuk menyimpan alamat dari suatu data
- Pointer tidak berisi nilai dari data, akan tetapi berisi alamat dari data
- Berguna untuk pengalokasian memori secara dinamis

Pointer

- Bentuk umum :

```
int *data_address;
```

- Untuk mengetahui alamat dari data, dapat dipakai tanda “&” :

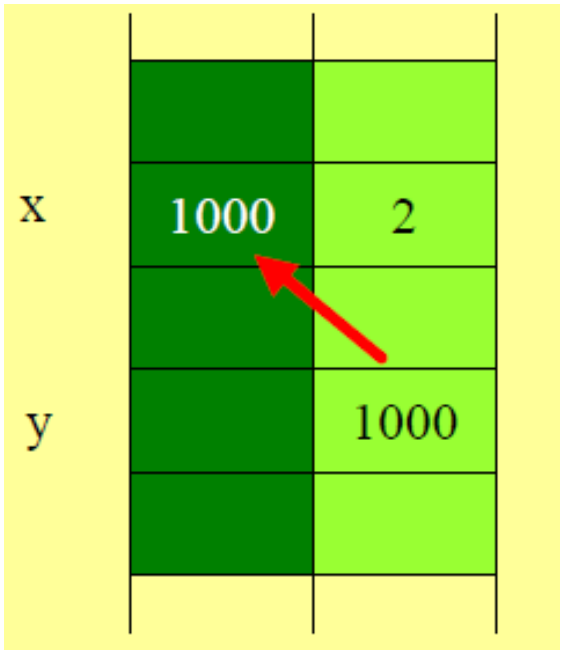
```
data_address = &data;
```

Pointer

```
int x;
int *y;


...


x = 2;
y = &x;
```



Sizeof ()

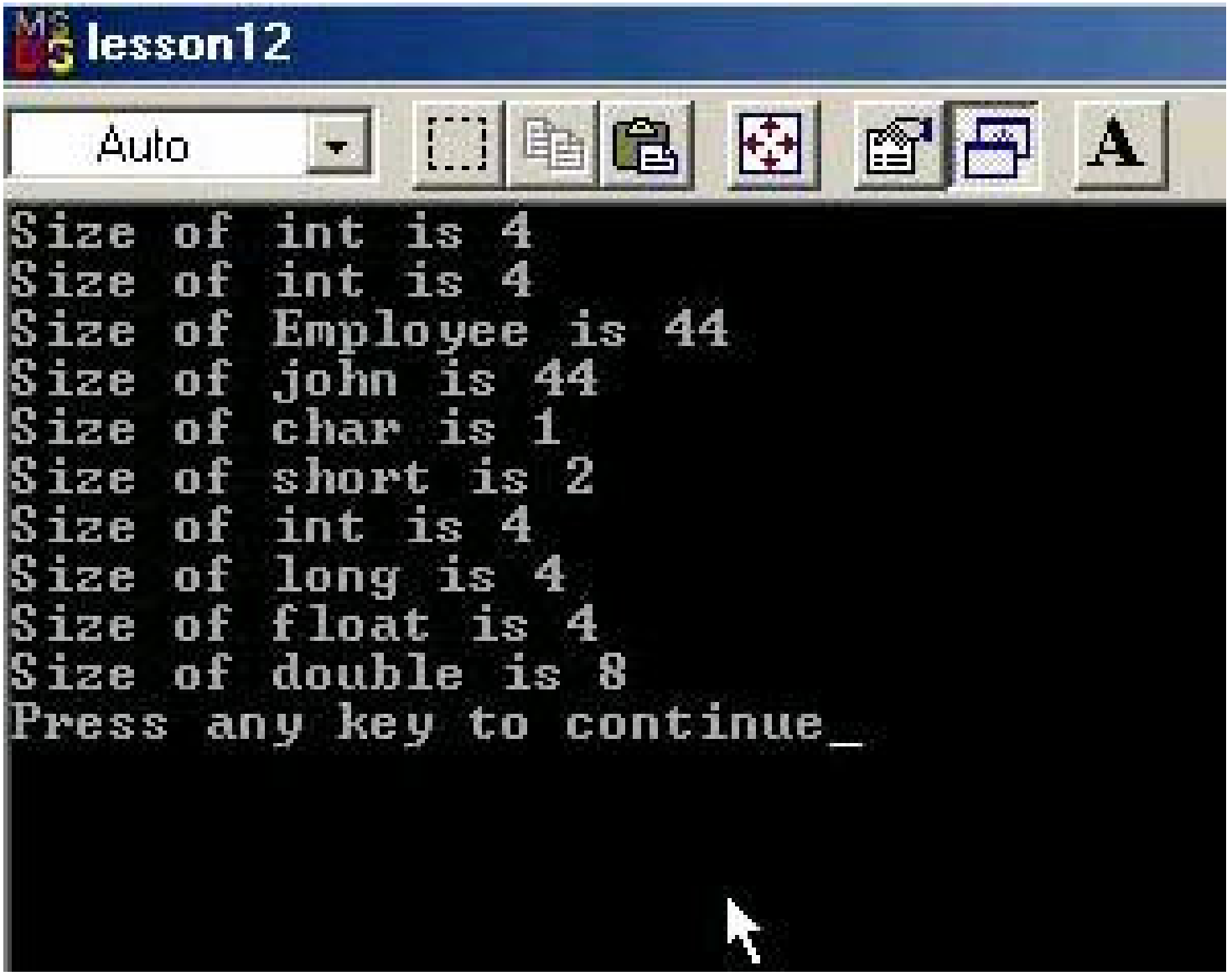
- Untuk mendapatkan ukuran dari berbagai tipe data, variabel ataupun struktur
- Return value : ukuran dari obyek yang bersangkutan dalam byte
- Parameter dari sizeof() : sebuah obyek atau sebuah tipe data

```
typedef struct employee_st {
   char name[40];
   int id;
} employee;

main () {
   int myInt;
   employee john;

   printf("size of int is %d\n", sizeof(myInt));
   printf("size of int is %d\n", sizeof(int));
   printf("size of Employee is %d\n", sizeof(Employee));
   printf("size of john is %d\n", sizeof(john));
   printf("size of char is %d\n", sizeof(char));
   printf("size of short is %d\n", sizeof(short));
   printf("size of long is %d\n", sizeof(long));
   printf("size of float is %d\n", sizeof(float));
   printf("size of double is %d\n", sizeof(double));

   return 0;
}
```


lesson12
Auto
Size of int is 4
Size of int is 4
Size of Employee is 44
Size of john is 44
Size of char is 1
Size of short is 2
Size of int is 4
Size of long is 4
Size of float is 4
Size of double is 8
Press any key to continue_

Malloc ()

- Fungsi standar yang digunakan untuk mengalokasikan memori
- Bentuk prototypenya adalah

  void *malloc (int jml_byte)
- Banyaknya byte yang akan dipesan dinyatakan sebagai parameter fungsi
- Return value dari fungsi ini adalah sebuah pointer yang tak bertipe (*pointer to void*) yang menunjuk ke buffer yang dialokasikan
- Pointer tersebut haruslah dikonversi kepada tipe yang sesuai (dengan menggunakan *type cast*) agar bisa mengakses data yang disimpan dalam buffer

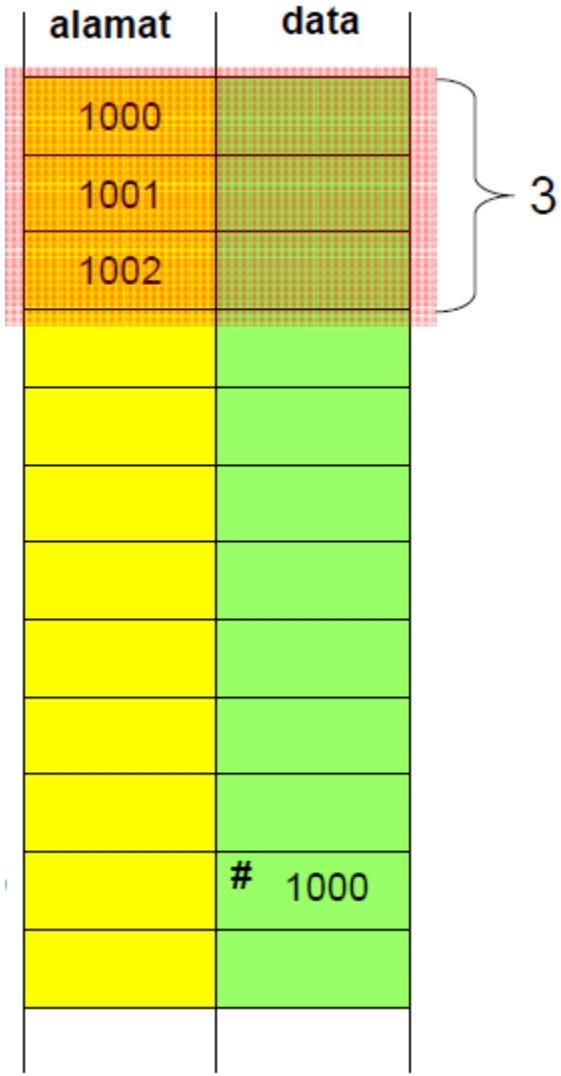
Malloc ()

```
int *x;
X = (int *)malloc(3*sizeof(int));
if (x==NULL) {
  printf("error on malloc\n");
  exit(0);
} else {
  // ...
}
```

Jika proses alokasi gagal dilakukan, fungsi ini akan memberikan return value berupa sebuah pointer NULL

Sebelum dilakukan proses lebih lanjut, perlu terlebih dahulu dipastikan keberhasilan proses pemesanan memori

```
int *p;
P = (int *)malloc(3*sizeof(int));
```



Free ()

- Membebaskan kembali memori
- Jika bekerja dengan menggunakan memori yang dialokasikan secara dinamis, maka seorang programmer haruslah membebaskan kembali memori yang telah selesai digunakan untuk dikembalikan kepada sistem.
- Setelah suatu ruang memori dibebaskan, ruang tersebut bisa dipakai lagi untuk alokasi variable dinamis lainnya.

Free()

```
#include <stdio.h>
#include <stdlib.h>

Main()
{
  char *pblok;
  pblok = (char *)malloc(500*sizeof(char));

  if (pblok == NULL)
     puts("error on malloc");
  else {
  puts("OK, memory allocation has been done");
  puts("--------");
  free(pblok);
  puts("free memory well done");
  }
}
```

Realloc ()

- Mengalokasikan ulang memori
- Ketika hendak mengalokasikan memori, user tidak pasti berapa besar lokasi yang dibutuhkannya.
- Misalnya user tersebut memesan 500 lokasi, ternyata setelah proses pemasukan data membutuhkan lebih dari 500 lokasi dan dialokasikan kembali menjadi 600.
- Maka user dapat mengalokasikan ulang memori yang dipesannya dengan menggunakan fungsi **realloc();**
- Fungsi ini akan mengalokasikan kembali pointer yang sebelumnya telah diatur untuk menunjuk sejumlah lokasi, memberinya ukuran yang baru (bisa jadi lebih besar atau lebih kecil)

Realloc ()

```
...
  pblok = (char *)malloc(500*sizeof(char));
...
...
  pblok = realloc(pblok, 600*sizeof(char));
```

Struct

- Cara pengelompokan data dengan nama yang sama, akan tetapi dapat mempunyai tipe-tipe yang berbeda
- Struktur biasa dipakai untuk pengelompokan beberapa informasi yang berkaitan menjadi sebuah satu kesatuan

- Deklarasi struct :

```
struct nameofstruct {
    type data1;
    ...
    type datan;
} var1, ..., varn;
```

Deklarasi Struct

```
struct data_siswa{
  int nrp;
  char nama[20];
};
struct data_siswa siswa;

struct data_tanggal{
  int tanggal;
  int bulan;
  int tahun;
} tanggal_lahir;

Struct data_tanggal tanggal lahir;
```

Inisialisasi Struct

```
struct data_siswa{
  int nrp;
  char nama[20];
} siswa = {7, “agus”};
```

Mengakses Struct

```
struct data_tanggal{
   int tanggal;
   int bulan;
   int tahun;
} tanggal_lahir;

tanggal_lahir.date  = 17;
tanggal_lahir.month = 8;
tanggal_lahir.year  = 1945;
```

Mengakses struct dapat dilakukan dengan menyebutkan nama variabel struct dan diiringi dengan data pada struct yang ingin diakses (dipisahkan dengan tanda .)

Struct dalam struct

```
struct data_tanggal{
   int tanggal;
   int bulan;
   int tahun;
};

struct data_siswa {
   int nrp;
   char nama[20];
   struct data_tanggal tanggal_lahir;
} siswa;

Student.tanggal_lahir.tanggal = 17;
```