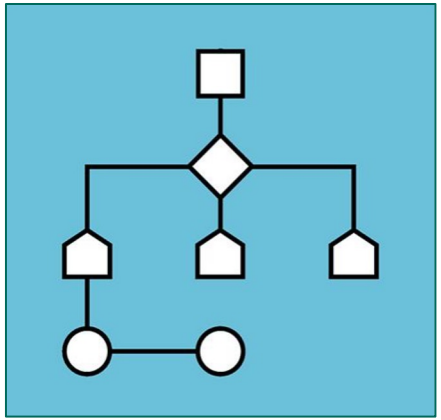
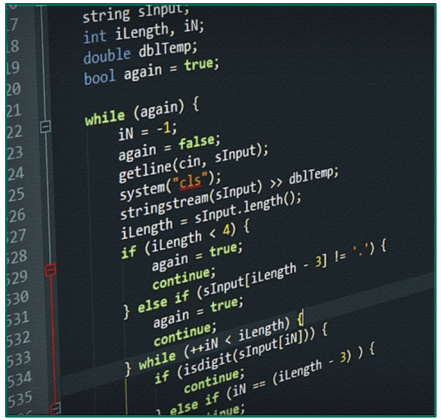
IPB University
Bogor Indonesia

KOM120B #11

Dynamic Memory Allocation





Tim Pengajar

KOM120B – Algoritme dan Dasar Pemrograman

Departemen Ilmu Komputer - FMIPA

Nilai Latihan : Pointer dan Fungsi

Jumlah yang berlatih : 144 (72%)

Partisipasi Mahasiswa dalam Latihan 3

PARALEL	MHS	ATTEMPT	PERSEN	RATAAN
K1	27	22	81.48	60.55
K2	25	18	72.00	63.33
K3	29	17	58.62	61.76
K4	28	24	85.71	60.42
K5	45	30	66.67	52.33
K6	45	42	93.33	66.55
TOTAL	199	153	76.88	61.03

Banyak yang Salah Jawab

Fungsi untuk membaca array 1 dimensi yang tepat adalah .....

```
#define N 100
void readArray(int n, int arr[N])
{
     scanf("%d", &n);
     for (int i=0;i<*n; i++)
        scanf("%d", arr[i]);
}
```

```
#define N 100
void readArray(int *n, int arr[N])
{
     scanf("%d", n);
     for (int i=0;i<*n; i++)
        scanf("%d", &arr[i]);
}
```

```
#define N 100
void readArray(int *n, int arr[N])
{
     scanf("%d", &n);
     for (int i=0;i<n; i++)
        scanf("%d", &arr[i]);
}
```

SALAH

```
#define N 100
void readArray(int *n, int arr[N])
{
     scanf("%d", n);
     for (int i=0;i<n; i++)
        scanf("%d", &arr[i]);
}
```

```
#define N 100
void readArray(int n, int arr[N])
{
     scanf("%d", &n);
     for (int i=0;i<*n; i++)
          scanf("%d", &arr[i]);
}
```

Mengapa Dynamic Memory Allocation?

- Ukuran sebuah array yang dideklarasikan harus diketahui sebelumnya.
- Jika tidak didefinisikan, maka program akan menampilkan pesan error pada saat kompilasi program.
- Ukuran array akan efisien jika disesuaikan dengan kebutuhan.

Butuh teknik alokasi memori secara dinamis
(*Dynamic Memory Allocation*)

Dynamic Memory Allocation

- Memory akan dialokasikan pada saat run time
- Mengurangi beban pada memory stack karena data yang dimasukkan akan disimpan pada area lainnya (segmentation)

`void *malloc(size_t size);`

```
int i, n;
int *dt;
scanf("%d", &n);

// menyiapkan array int berukuran n → array 1D
dt=(int*)malloc(n*sizeof(int));

// memasukkan data ke array seperti biasa
for (i=0; i<n; i++)
        scanf("%d", &dt[i]);
```

memory allocation

dada

Inisialisasi Nilai Memory

Inisialisasi setiap elemen memory (array) dengan nilai 0 menggunakan fungsi:

- memset()
- calloc()

```
void *memset(void *str, int c, size_t n);
```

```
int i, n;
int *dt, *d2;
scanf("%d", &n);

// menyiapkan array int berukuran n → array 1D
dt=(int*)malloc(n*sizeof(int));
memset(dt, 0, (n*sizeof(int)));   // inisialisasi nilai 0 pada tiap sel

// menggunakan fungsi calloc()
d2 = (int*)calloc(n, sizeof(int));   // contiguous allocation
```

Reallocation

- Memory yang pernah dialokasikan menggunakan malloc() atau calloc() dapat di-re-alokasi Kembali (mengubah ukuran memory) menggunakan fungsi realloc()
- Lebih dinamis.

```
void *realloc(void *ptr, size_t size);
```

```
int i, n;
int *dt;
scanf("%d", &n);

dt=(int*)malloc(n*sizeof(int));
for (i=0; i<n; i++) scanf("%d", &dt[i]);

// menambah ukuran array menjadi n+3
dt=(int*)realloc(dt, (n+3)*sizeof(int));   // ukuran array sekarang: (n+3)

// membaca 3 data tambahan
for (; i<n+3; i++) scanf("%d", &dt[i]);
```

Deallocates

- Memory yang pernah dialokasikan menggunakan malloc(), calloc(), atau realloc() dapat di-dealokasi menggunakan fungsi free()
- Bermanfaat untuk “membersihkan” memory dari “sampah”.

void free(void *ptr);

```
int i, n;
int *dt;
scanf("%d", &n);

// menyiapkan array int berukuran n → array 1D
dt=(int*)malloc(n*sizeof(int));

// memasukkan data ke array seperti biasa
for (i=0; i<n; i++)
   scanf("%d", &dt[i]);
...
free(dt);
```

Contoh Array 1D

```
10
5 7 6 4 3 2 8 9 1 10
```

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

void print1d(int n, int a[n]);

int main()
{
   int i, n;
   int *dt;
   scanf("%d", &n);

   dt=(int*)malloc(n*sizeof(int));

   for (i=0; i<n; i++)
      scanf("%d", &dt[i]);

   // cetak isi array
   print1d(n, dt);

   return 0;
}
```

```c
void print1d(int n, int a[n])
{
    int i;
    for(i=0;i<n;i++)
    {
        printf("%d", a[i]);
        if (i==n-1) printf("\n");
        else printf(" ");
    }
}
```

Return Type: Pointer

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

// Fungsi membalik elemen array 1D
int* balik(int n, int* a)
{
    int *t;
    t=(int*)malloc(n*sizeof(int));
    for(int i=0;i<n;i++)
        t[i]=a[n-i-1];
    return t;
}

// Fungsi mencetak elemen array 1D
void print1d(int n, int *a);
```

```
int main()
{
    int i, n;
    int *dt, *res;
    scanf("%d", &n);

    dt=(int*)malloc(n*sizeof(int));
    res=(int*)malloc(n*sizeof(int));

    // input data
    for (i=0; i<n; i++)
        scanf("%d", &dt[i]);

    res = balik(n, dt);

    print1d(n, res);

    return 0;
}
```

Dynamic Memory Allocation 2D

1. Single pointer
2. Array of pointer
3. Pointer to a pointer

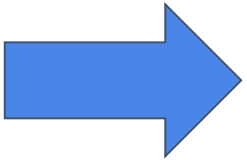
Misalnya kita akan menangani data 2 dimensi berukuran 3x4 berikut:

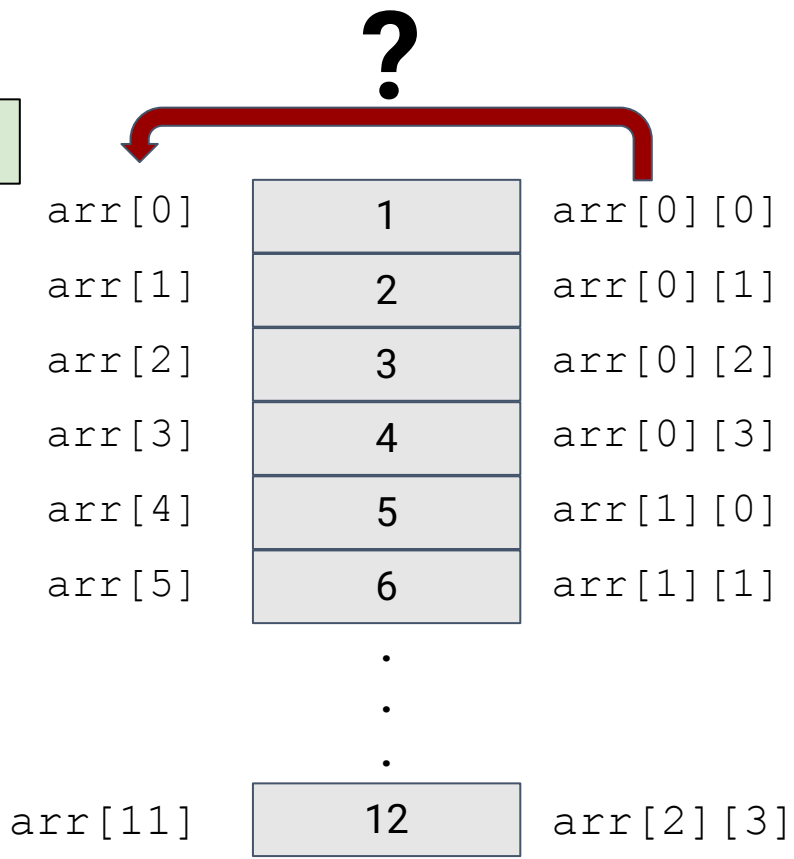
```
3  4
1  2  3  4
5  6  7  8
9  3  4  7
```

Single Pointer

Ingat, penyimpanan data array 2D dalam memory !

```
3 4
1 2 3 4
5 6 7 8
9 3 4 7
```





Fungsi Mencetak Array

```
void print1d(int n, int *a)
{
   int i;
   for(i=0;i<n;i++)
   {
      printf("%d", a[i]);
      if (i==n-1) printf("\n");
      else printf(" ");
   }
}
```

```
void print2d(int m, int n, int *a)
{
   int i,j;
   for(i=0;i<m;i++)
   {
      for (j=0;j<n;j++)
      {
         printf("%d", a[i*n+j]);
         if (j==n-1) printf("\n");
         else printf(" ");
      }
   }
}
```

(1) Single Pointer

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

int main()
{
    int r, c, i, j, nilai;
    int *arr;

    scanf("%d %d", &r, &c);
    arr = (int *)malloc(r * c * sizeof(int));

    for (i = 0; i < r; i++)
       for (j = 0; j < c; j++)
       {
          scanf("%d", &nilai);
          arr[i*c + j] = nilai;
       }

    print2d(r, c, arr);
    return 0;
}
```

(2) Array of Pointers

- Array 2D → setiap baris merupakan array 1D
- Sehingga dapat dibuat single pointer untuk tiap baris
- Berlaku untuk C99 atau yang lebih baru.

```
int r, c, i, j, nilai;

scanf("%d %d", &r, &c);

int *arr[r];
for (i=0; i<r; i++)
   arr[i] = (int *)malloc(c*sizeof(int));

for (i = 0; i < r; i++)
for (j = 0; j < c; j++)
{
    scanf("%d", &nilai);
    arr[i][j] = nilai;
}
```

(3) Pointer to a Pointer

- Array 2D → setiap baris merupakan array 1D
- Sehingga dapat dibuat single pointer untuk tiap baris
- Bisa juga dibuat single pointer to single pointer

```
int r, c, i, j, nilai;

scanf("%d %d", &r, &c);

int **arr = (int **)malloc(r*sizeof(int *));
for (i=0; i<r; i++)
    arr[i] = (int *)malloc(c*sizeof(int));


for (i = 0; i < r; i++)
    for (j = 0; j < c; j++)
    {
        scanf("%d", &nilai);
        arr[i][j] = nilai;
    }
```

Latihan 1 : Merge → Gunakan DMA

Diketahui 2 deretan bilangan yang sudah terurut ascending, masing-masing sebanyak n dan m bilangan. Buat program mencetak seluruh bilangan yang ada secara terurut ascending juga. [TIDAK BOLEH ADA PROSES SORTING ARRAY]

Contoh Input:

```
3 5          → banyaknya bilangan (n dan m)
2 5 9
4 8 10 15 20
```

Contoh Output:

```
2 4 5 8 9 10 15 20
```

Latihan 2 : Susunan Nomor Map → Gunakan DMA

Oh Yoon-seo, seorang sekretaris di kantornya Kwon Jung-rok sedang mengatur susunan map tebal bernomor yang berisi arsip sidang pengadilan agar mudah mencarinya nanti, yaitu agar nomornya terurut dari kecil ke besar. Dalam mengatur susunan map, dilakukan prosedur: (a) cabut satu map, (b) geser beberapa map ke kiri atau ke kanan, (c) kemudian dia masukkan map tadi ke posisi yang seharusnya. Prosedur (a)-(b)-(c) ini dihitung sebagai 1 langkah. Sebagai contoh, jika susunan map adalah 1 3 2 5 4, maka dibutuhkan 2 langkah sehingga susunannya menjadi 1 2 3 4 5. Buat program C untuk menentukan banyaknya langkah minimum agar nomor map terurut. Maksikum map sebanyak 10 ribu.

Contoh Input:

```
8
10 22 9 33 21 50 41 60
```

Contoh Output:

```
3
```