



استاد : دکتر مرادی،
دکتر هاشمی

عنوان:
آرایه‌های پویا

نیمسال اول
۰۱-۰۲

آرایه‌ی پویا^۱:

آرایه‌هایی که تا به حال دیده‌اید و از آن‌ها استفاده کرده‌اید، آرایه‌های ایستا^۲ بوده‌اند. اگر یادتان باشد در تعریف این آرایه‌ها حتماً باید طول آن‌ها را با یک عدد ثابت مشخص می‌کردید. امروز می‌خواهیم با نوع دیگری از آرایه‌ها به نام آرایه پویا آشنا شویم. طول این آرایه‌ها در هنگام کامپایل نامشخص بوده و در هنگام اجرا تعیین می‌گردد.

دستور تخصیص حافظه^۳ (malloc):

شما می‌توانید توسط تابع malloc که از توابع کتابخانه‌ی `stdlib.h` می‌باشد، از سیستم عامل درخواست کنید که مقدار مشخصی حافظه در `heap` گرفته و آن را در اختیار شما قرار دهد. نحوه‌ی استفاده از این تابع به صورت زیر است:

```
<type>* pointer = (<type>*) malloc(number * sizeof(<type>));
```

<type> : نوع داده‌ای که می‌خواهید آرایه‌ای پویا از آن داشته باشید.

number : طول آرایه‌ای که می‌خواهید.

حال به نکات زیر توجه کنید:

۱. آرگومان تابع malloc مقدار حافظه درخواستی بر حسب بایت می‌باشد.
۲. sizeof از عملگرهای زبان C است که سایز هر type ای که به آن بدهید را بر حسب بایت برمی‌گرداند. چون سایز یک type (مثلاً int) در سیستم‌های مختلف ممکن است متفاوت باشد، بهتر است از عملگر sizeof استفاده کنید.
۳. مقدار برگشتی تابع malloc در صورتی که تخصیص حافظه موفقیت آمیز باشد، اشاره‌گر به سرآرایه‌ی پویا خواهد بود و در غیر این صورت NULL است. لذا بعد از فراخوانی این تابع حتماً باید بررسی کنید که اگر مقدار بازگشتی NULL بود، ضمن دادن پیغام خطا به کاربر از برنامه خارج شوید.
۴. همچنین مقدار برگشتی این تابع از جنس void* (یعنی اشاره‌گری که می‌تواند از هر نوعی باشد و لزوماً قرار نیست نوع خاصی، مثلاً integer، داشته باشد) بوده و برای همین آن را به type مورد نظر cast کرده‌ایم.
۵. هر حافظه‌ای که توسط تابع malloc می‌گیرید را در پایان باید توسط تابع free(pointer) آزاد کنید.

¹ dynamic

² static

³ memory allocation

۶. نحوه‌ی استفاده از آرایه‌های ایستا و پویا هیچ تفاوتی با هم ندارد. پس همان گونه که قبلاً با آرایه های ایستا کار می‌کردید، می‌توانید با آرایه های پویا نیز کار کنید.

برای درک نکات بالا به برنامه زیر توجه کنید:

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int arr_size, i;
    int* dynamic_arr;
    printf("Enter the size of array:\n");
    scanf("%d", &arr_size);
    /* Requesting an integer array with capacity of arr_size elements.
     * On success dynamic_arr will be a pointer to the beginning of the array.
     * On failure dynamic_arr will be null. */
    dynamic_arr = (int*) malloc (arr_size * sizeof(int));
    if (dynamic_arr == NULL) {
        printf("Oops! Memory allocation failed.\n");
        exit(EXIT_FAILURE);
    }
    /* From now on you can work with the dynamic array just like static arrays!*/
    printf("Enter %d numbers:\n", arr_size);
    for (i = 0; i < arr_size; i++)
        scanf ("%d", &dynamic_arr[i]);
    free(dynamic_arr); /* Do not forget to free the allocated memory! */
    return 0;
}
```

۱. انجام دهید

یکی از توابع کاربردی برای تخصیص حافظه‌های پویا تابع `realloc` است. از این تابع می‌توانید برای تغییر مقدار حافظه‌ای که قبلاً از سیستم گرفته بودید، استفاده کنید. تعریف این تابع به صورت زیر است:

```
<type>* pointer = (<type>*) realloc(pointer, number * sizeof(<type>));
```

این تابع به عنوان ورودی اشاره‌گر فعلی و اندازه‌ی جدید مورد نظر را گرفته و اشاره‌گر جدید را بر می‌گرداند. برای آشنایی با رفتار این تابع، قطعه کدهای زیر را اجرا کنید.

قطعه کد اول:

```
#include <stdio.h>
#include <stdlib.h>
#define INITIAL_ARRAY_SIZE 4
#define FINAL_ARRAY_SIZE 5
#define ZERO 0
#define ONE 1
```

```

#define FIVE 5

void print_array(int* array_of_int, int size) {
    for (int i = 0; i < size; i++)
        printf("%d ", array_of_int[i]);
    printf("\n");
}

void initislizing_the_array(int* array_of_int, int size) {
    for (int i = ZERO; i < size; i++)
        array_of_int[i] = ZERO;
}

int main() {
    int* array_of_int = (int*)malloc(INITIAL_ARRAY_SIZE *
sizeof(int));
    initislizing_the_array(array_of_int, INITIAL_ARRAY_SIZE);
    print_array(array_of_int, INITIAL_ARRAY_SIZE);
    /**/
    array_of_int = (int*)realloc(array_of_int, FINAL_ARRAY_SIZE *
sizeof(int));
    /**/
    initislizing_the_array(array_of_int, FINAL_ARRAY_SIZE);
    array_of_int[FINAL_ARRAY_SIZE - ONE] = FIVE;
    print_array(array_of_int, FINAL_ARRAY_SIZE);
    return 0;
}

```

حالا دستور زیر را با خط مشخص شده در قطعه کد بالا جایگزین کنید و آن را اجرا نمایید. تفاوت نتیجه‌ی این دو قطعه کد در چیست؟

```
array_of_int = (int*)realloc(NULL, FINAL_ARRAY_SIZE * sizeof(int));
```

قسمت ۱: نتیجه را به دستیاران آموزشی نشان دهید.

۲. انجام دهید (Free)



(۱) هنگام کار با تخصیص حافظه‌ی پویا به دو نکته باید توجه نمود:

۱. لزوم آزاد کردن حافظه‌های گرفته شده

۲. عدم استفاده از حافظه‌ی آزاد شده

۲) قطعه کد زیر را اجرا کنید.

```
int main() {
    int* p = (int*)malloc(10 * sizeof(int));
    int i;
    printf("P = 0x%p\n", p);
    for (i = 0; i < 10; i++)
    {
        p[i] = i;
    }
    free(p);
    printf("P = 0x%p\n", p);
    printf("P[0] = %d", *p);
    return 0;
}
```

پس از آزادسازی حافظه‌ی گرفته شده دقیقاً چه اتفاقی می‌افتد؟ خروجی‌ها را توجیه کنید.

قسمت ۲: نتیجه را با دستیاران آموزشی در میان بگذارید.

فکر کنید: چرا اگر در کد بالا اقدام به چاپ `p[15]` کنیم برنامه خطا نمی‌دهد؟

← ۳. انجام دهید (Memory Leak)

حافظه‌های گرفته شده از سیستم هنگام اجرای برنامه حتماً باید در انتهای برنامه آزاد شوند. در این قسمت مشاهده می‌کنیم که در صورتی حافظه‌ی گرفته شده آزاد نشود ممکن است چه مشکلاتی برای سیستم ایجاد شود.

۱) کد زیر را اجرا کنید.

```
void main() {
    int *p = NULL;
    int i = 500000;
    for(int j=0, j<5, j++) {
        p = realloc(p, i * sizeof(int)); /*put breakpoint here*/
        i+=500000;
    }
    free(p);
    return 0;
}
```

۲) با استفاده از کلیدهای `ctrl+shift+esc` پنجره‌ی task manager را باز نمایید.

۳) سربرگ Processes را انتخاب کنید.

- ۴) برنامه‌ی اجرایی `your_project_name.exe` را پیدا کنید و مقدار `memory` مورد استفاده‌ی آن را مشاهده کنید.
- ۵) حال مقدار نهایی متغیر `j` در دستور `for` را به عدد ۱۰ تغییر دهید. `for(int j=0, j<5, j++)`
- ۶) مقدار `memory` گرفته شده توسط برنامه را مجدداً مشاهده کنید.
- ۷) چه نتیجه‌ای می‌گیرید؟

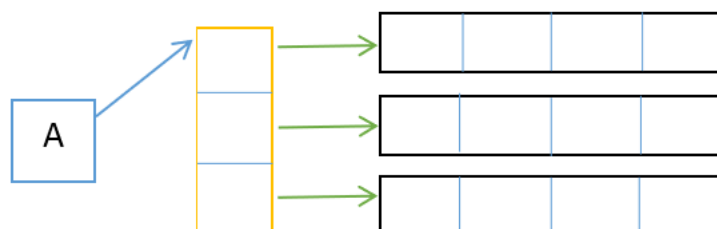
قسمت ۳: نتیجه را با دستیاران آموزشی در میان بگذارید.

فکر کنید: آیا آدرس `p` که به ابتدای حافظه‌ی مورد نظر ما اشاره می‌کند همیشه یکسان است؟

آرایه پویای دو بعدی (!):

قبلاً (اگر یادتان باشد :)) با آرایه‌های چند بعدی هم کار کرده بودیم. حال می‌خواهیم همان آرایه‌ها را نیز به شکل پویا درست کنیم. یک آرایه‌ی دوبعدی، عملاً آرایه‌ای از آرایه‌های یک بعدی است. مثلاً `int a[3][4]` یک آرایه‌ی ۳ تایی از آرایه‌هایی به طول ۴ می‌باشد. شکل زیر را نگاه کنید:

	Column 0	Column 1	Column 2	Column 3
Row 0	<code>a[0][0]</code>	<code>a[0][1]</code>	<code>a[0][2]</code>	<code>a[0][3]</code>
Row 1	<code>a[1][0]</code>	<code>a[1][1]</code>	<code>a[1][2]</code>	<code>a[1][3]</code>
Row 2	<code>a[2][0]</code>	<code>a[2][1]</code>	<code>a[2][2]</code>	<code>a[2][3]</code>



پس اگر ما ابتدا یک آرایه از جنس `int**` به طول ۳ بسازیم که `A` سر آن باشد، و سپس هر کدام از خانه‌های آن آرایه را برابر با سر یک آرایه‌ی ۴ تایی قرار دهیم، یک آرایه‌ی دوبعدی ساخته‌ایم.

← **۴. انجام دهید!**

قسمت‌های مشخص شده در کد زیر کامل کرده، سپس کد زیر را اجرا کنید. در این برنامه می‌خواهیم یک آرایه‌ی دو بعدی بسازیم، آن را مقدار دهی کرده، مقادیر را چاپ کرده، و سپس حافظه‌ی تخصیص داده شده را آزاد کنیم.

```

#include <stdio.h>
#include <stdlib.h>
#define ZERO 0

void print_array(int** 2d_array_of_int, int size1 , int size2) {
    for (int i = 0; i < size1; i++)
    {
        for (int j = 0; j < size2; j++)
            printf("%d ", array_of_int[i][j]);

        printf("\n");
    }
}

void initislizing_the_array(int** 2d_array_of_int, int size1, int
size2) {
    for (int i = ZERO; i < size1; i++)
        for (int j = ZERO; j < size2; j++)
            array_of_int[i][j] = ZERO;
}

int main() {
    int row, col, i;
    int** A;
    printf("Enter row and column:\n");
    scanf("%d %d", &row, &col);
    A=(...) malloc(...*sizeof(...)); /* 1. Complete this instruction */
    if (A == NULL) exit(EXIT_FAILURE);
    for(i = 0; i < row ; i++) {
        A[i]=(...) malloc(...*sizeof(...)); /* 2. Complete this instruction */
        if (A[i] == NULL)
            exit(EXIT_FAILURE);
    }
    /* Now you have a 2D integer array */

    // Your Program Goes Here.

    /* Don't forget to free the allocated memory when you don't need it any more */
    for (i = 0; i < row; i++)
        free(A[i]);
    free(A);
    return 0;
}

```

قسمت ۵ : نتیجه را به دستیاران آموزشی نشان دهید.

توجه: همه ی نکات ذکر شده به راحتی قابل تعمیم به یک آرایه ی n بعدی است

← ۵. فکر کنید.

کد زیر را اجرا کنید و درباره ی خطوط اشاره شده فکر کنید.

```
#include <stdio.h>
#include <stdlib.h>
void main() {
    int i;
    char* s;
    int *p=(int*)malloc(10*sizeof(int));
    for(i=0;i<10;i++)
        p[i]=i+48;
    s=(char*)p;
    for(i=0; i<40; i++) /* Pay attention to bound of for */
        printf("%c", s[i]); /* What's happening here? what is value of s[1]? why? */
    printf("\n");
    p++;
    printf("p[1] is %d\n",*p);
    free(p); /* What's happening here? */
}
```

قسمت ۶ : نتیجه را با دستیاران آموزشی در میان بگذارید.

موفق باشید