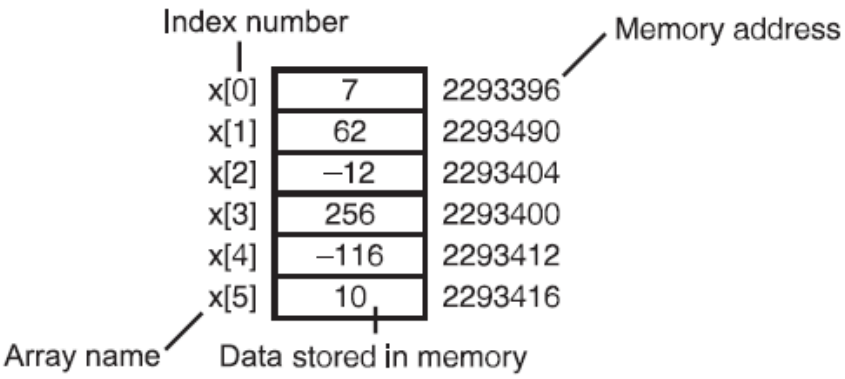
نیمسال دوم 99-98	عنوان: آزمایشگاه چهارم (آرایه‌ها)	استاد : دکتر مرادی

در این جلسه شما با آرایه[1] ها و نوع خاصی از آن ها یعنی رشته[2] ها آشنا خواهید شد.

تعریف آرایه: تعدادی داده های پشت سر هم در حافظه که همگی از یک نوع[3] هستند.



نکته1 : شماره اندیس آرایه به طول n از 0 شروع و تا n-1 ادامه دارد.

برای تعریف و مقدار دهی آرایه‌ها در زبان c روش های متعددی وجود دارد. در کد زیر تعدادی از این روش‌ها ذکر شده‌اند.

```
√ int a[] = { 5, 3, 2 };
√ int b[3];
√ int c[3] = { 6, 5 };
√ b[0] = 2;

√ int d[3];
× d = { 4, 5, 6 };
√ int e[3];
× e[3] = 4;
```

نکته2: مقداردهی آرایه با استفاده از {} تنها هنگام تعریف آرایه مجاز است.

نکته3: طول آرایه‌ها باید هنگام در زمان `compile` تعیین شود. بدین معنا که طول آرایه در زمان اجرای برنامه نمی توان تعریف یا تغییر داده شود. با توجه به این موضوع طول آرایه می‌تواند به صورت متغیر `const` تعریف شود.

1. انجام دهید!

[1] array
[2] string
[3] type

1. یک آرایه از نوع `int` به طول 5 تعریف کنید. طول آرایه را با استفاده از یک متغیر تعریف کنید. برنامه خود را کامپایل کرده و نتیجه را ببینید. چه خطایی رخ میدهد و چرا؟
2. برنامه بالا را با تغییر متغیر به یک متغیر `const` تغییر داده و مجددا کامپایل کنید. آیا خطا رخ نمی‌دهد؟ چرا؟
3. با استفاده از حلقه `for` و دستور `scanf` ، 5 عدد از کاربر دریافت نموده و در خانه‌های آرایه ذخیره کنید.
4. با استفاده از دستور `printf` مقادیر وارد شده را به ترتیب معکوس در خروجی چاپ کنید.

راهنمایی :

```
scanf("%d", &a[i]);
```

قسمت 1 : موارد خواسته شده را انجام دهید. نتایج به دست آمده و یافته‌های خود را در کادر زیر بنویسید.

دلیل بروز خطا در قسمت 1 این است که وقتی اندازه آرایه با متغیر تعریف می شود این متغیر می تواند مقدارش در طی برنامه عوض شده و ثابت نماند پس چون مقدار متغیر ثابت نیست و امکان تغییر مقدار متغیر وجود دارد نمی توان از آن در تعریف آرایه استفاده کرد.

در قسمت دوم نیز به خطا می خوریم و دلیل آن این است که چون اندازه آرایه باید در زمان کامپایل مشخص شود با تعریف متغیر const چون این متغیر می تواند مقدار ثابت خود را در زمان اجرا بگیرد باز برای آرایه مشکل ایجاد می شود پس دلیل این خطا از مقداردهی متغیر const ایجاد می شود چون در زمان اجرا ممکن است مقدار این متغیر مشخص شود و نه در زمان کامپایل.

```c
#include <stdio.h>

int main(){
	int a[5];
	for (int i = 0; i < 5; i++){
		scanf("%d", &a[i]);
	}

	for (int i = 4; i >= 0; i--){
		printf("%d ", a[i]);
	}
	printf("\n");
	return 0;
}
```

2. انجام دهید!

1) در تابع main یک آرایه از کاراکترها (به طول 5) به نام msg بسازید و سعی کنید به عنوان مقدار اولیه رشته ی "Hello" را در آن بریزید.
2) به compile error تولید شده دقت کنید. چرا با وجود ان که طول رشته و طول آرایه برابر است این خطا تولید شده است؟ درباره NULL charachter یا '\0' تحقیق کنید.
3) حال به وسیله ی یک حلقه for طول رشته ی msg را محاسبه و بر روی صفحه چاپ کنید.

قسمت 2 : موارد خواسته شده را انجام دهید. نتایج به دست آمده و یافته‌های خود را در کادر زیر بنویسید.

مشکل ایجاد شده به خاطر این است که وقتی که "Hello" را به آرایه می دهیم این آرایه از حروف این لغت که در خانه های 0 تا 4 آرایه قرار می گیرند تشکیل می شود و علاوه بر آن یک null character نیز در انتهای آرایه یعنی در خانه 5 ام قرار می گیرد پس در نتیجه به خاطر این که سایز آرایه 5 است ولی مقدار داده شده به آن اندازه ای برابر 6 دارد مشکل ایجاد می شود.

```
#include <stdio.h>

int main(){
	char msg[] = "Hello";
	int num;
	for (num = 0; msg[num] != '\0'; num++){
		printf("%c", msg[num]);
	}
	printf("\nSize of msg: %d\n" , num);
	return 0;
}
```

در این کد msg به همراه اندازه آن چاپ می شود.

3. انجام دهید!

یک آرایه را نمی توان برابر یک آرایه ی دیگر گذاشت. مثلاً برنامه ی زیر خطای کامپایلی دارد.

```
int arr[4] = {1, 2, 3, 4};
int arr_copy[4];
arr_copy = arr;
```

1. برنامه را کامپایل کرده و خطا را بنویسید.
2. چرا این خطا رخ میدهد؟
3. کد را بگونه ای تغییر دهید تا مشکل حل شده و محتوای یک آرایه را در یک آرایه دیگر ذخیره کنید.

قسمت 3 : موارد خواسته شده را انجام دهید. نتایج به دست آمده و یافته‌های خود را در کادر زیر بنویسید.

error c2106 '=' left operand must be l-value خطای ایجاد شده است.

دلیل بروز این خطا این است که آرایه امکان مقدار دهی مستقیم به شکل بالا را ندارد یعنی همان طور که در ارور گفته شده مقدار معتبری برای قرار گیری در سمت چپ تساوی نیست هر خانه آرایه باید به صورت جداگانه و مستقل مقدار دهی شود یعنی اول خانه صفر آرایه سپس خانه یک سپس خانه دو و به همین شکل تا پر شدن آرایه پیش می رویم و مقدار دهی می کنیم و نمی توان به شکل بالا برای مقداردهی عمل کرد.

```
#include <stdio.h>

int main(){
	int arr[4] = { 1, 2, 3, 4 };
	int arr_copy[4];
	for (int i = 0; i < 4; i++){
		arr_copy[i] = arr[i];
	}
	return 0;
}
```

همان طور که در کد نوشته شده برای کپی کردن، هر خانه آرایه را باید جداگانه در آرایه دیگر قرار داد.

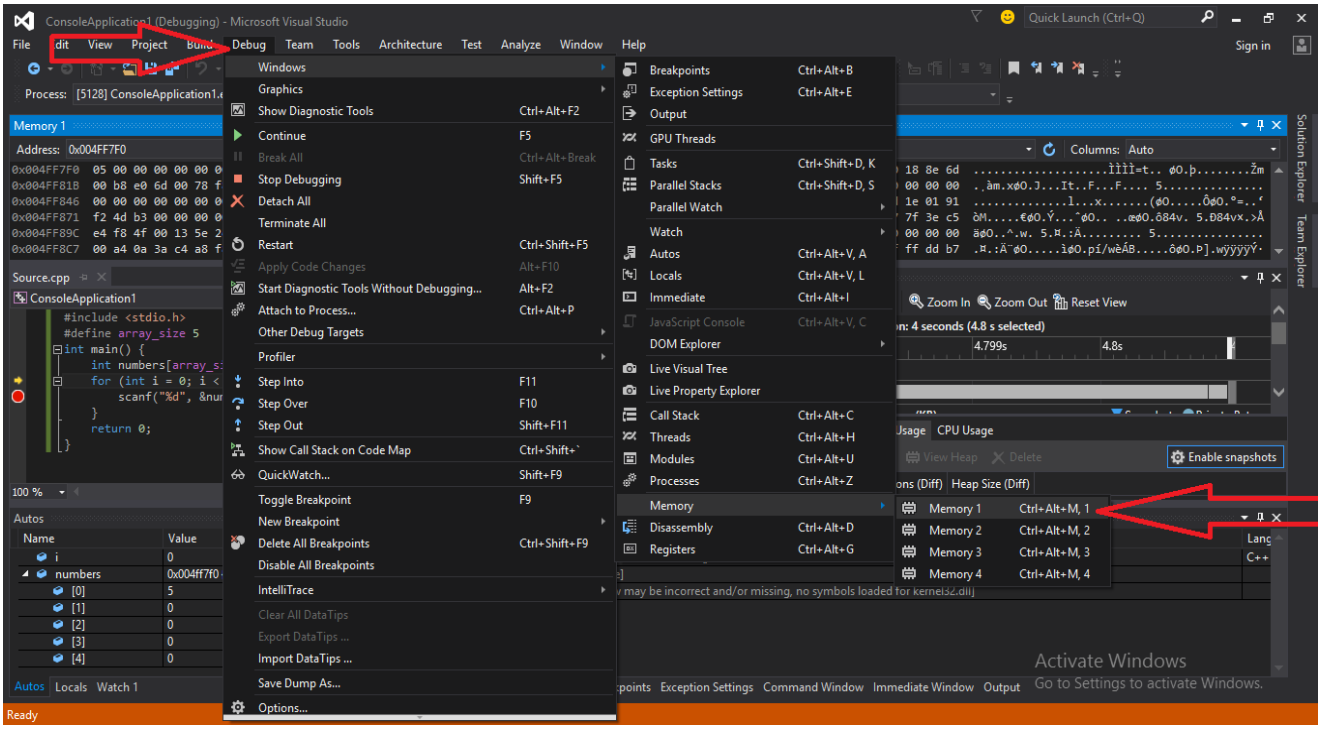
4. انجام دهید!

1) مشابه قطعه کد زیر، آرایه‌ای به طول 5 تعریف کرده و یک break point بر روی دستور scanf تعریف کنید:

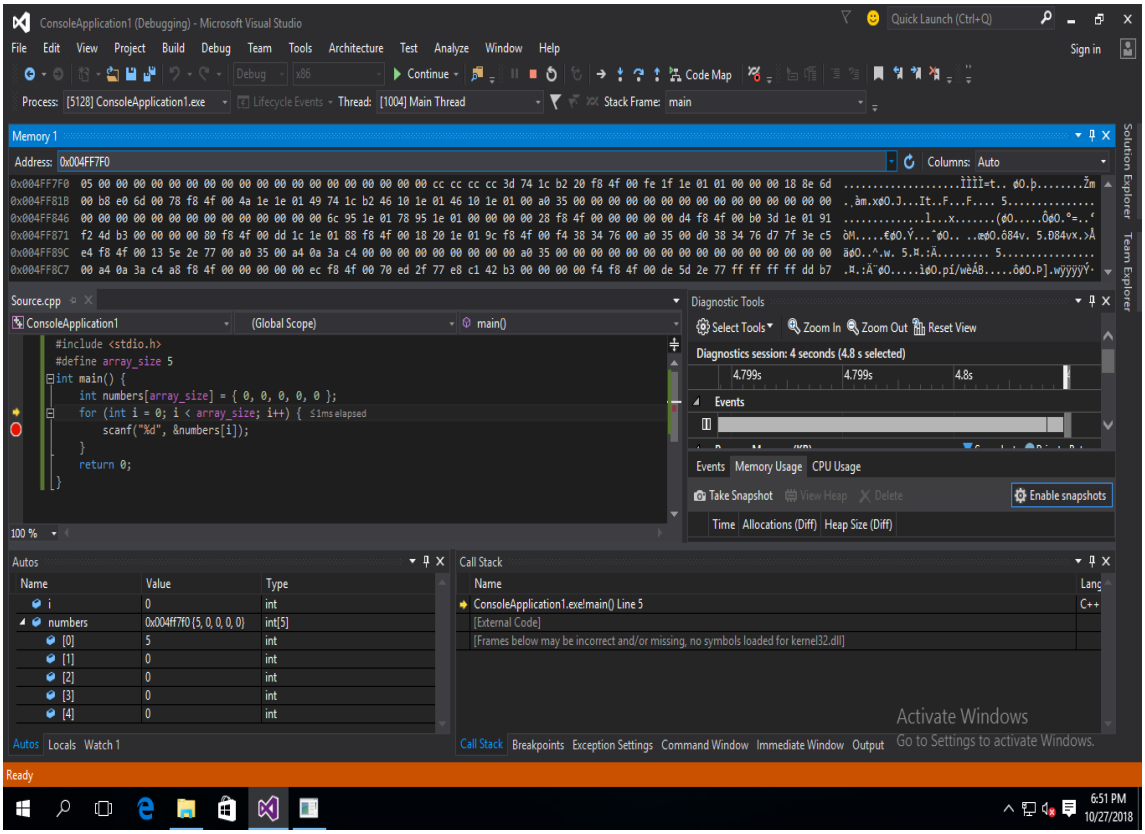
```
#include <stdio.h>
#define array_size 5

int main() {
	int numbers[array_size] = { 0, 0, 0, 0, 0 };
	for (int i = 0; i < array_size; i++) {
		scanf("%d", &numbers[i]);
	}
	return 0;

}
```

2) کد خود را با Debug mode اجرا کنید.

3) مطابق شکل زیر، پنجره‌ای که مقادیر موجود در حافظه را به شما نشان می دهد، فعال کنید.



4) کد را تا انتها اجرا کرده و با نحوه پر شدن خانه‌های حافظه آشنا شوید.



قسمت 4 : نتایج به دست آمده را در کادر زیر بنویسید.

در ابتدا آدرس آرایه را با add watch کردن در پیدا می کنیم در آن آدرس به اندازه آرایه جا خالی است یعنی در واقع عدد 00 در خانه ها قرار گرفته است هر بار با گرفتن مقدار جدید عدد گرفته شده در یک خانه از حافظه قرار می گیرد با دریافت عدد جدید در سه خانه جلو تر از مقدار قبل مقدار جدید قرار می گیرد و همین روند تا اتمام پر شدن آرایه ادامه می یابد با این کار آرایه ما با اعداد دریافتی از طرف کاربر در قسمتی از حافظه قرار می گیرد.

5. انجام دهید!

در این قسمت، به عنوان ورودی برنامه، آرایه‌ای از اعداد طبیعی به شما داده می‌شود و خواسته‌ی مسئله این است که برای هر یک از اعداد موجود در آرایه، اولین عددی را که در سمت چپ آن بوده و بزرگتر از آن است، بیابید. چنانچه چنین عددی وجود نداشت، مقدار پیش‌فرض صفر برای آن در نظر گرفته می‌شود. به عنوان مثال، رشته‌ی زیر را در نظر بگیرید:

{3, 2, 5, 4, 1, 2, 3, 8, 3, 6}

اگر از چپ به راست روی عناصر این رشته حرکت کنیم، به عنوان مثال برای عنصر سوم که عدد 5 است، پاسخ 1- خواهد بود. یا برای عنصر شماره‌ی 3 که عدد 4 است، پاسخ اندیس مربوط به عدد 5، یعنی 2 است. چنانچه در زیر مشاهده می‌شود، پاسخ کلی مثال بالا این است:

{-1, 0, -1, 2, 3, 3, 3, -1, 7, 7}

حال مراحل زیر را به ترتیب انجام دهید:

1) یک آرایه از اعداد صحیح و به طول 10، با مقادیر دلخواه خودتان ایجاد کنید.(می‌توانید با استفاده از تولید کردن اعداد تصادفی این کار را بکنید. در آزمایش‌های قبلی این کار را یاد گرفته‌اید :))
2) ابتدا مقدار میانگین اعداد آرایه را محاسبه و چاپ نمایید.(برای این کار با استفاده از حلقه for جمع اعداد آرایه را حساب کرده و بر 10 تقسیم کنید. برای دقیق‌تر بودن پاسخ از متغیر float برای نگه داری جمع اعداد استفاده کنید.)
3) اکنون، الگوریتمی را پیدا کنید که در آن با استفاده از حلقه for، بتوان برای هر یک از اعداد موجود در آرایه، اولین عددی را که در اندیس‌های کمتر از آن قرار داشته و از آن بزرگتر است، یافت. به نظر شما این روش به ازای n تا ورودی، ماکزیمم چند بار پیمایش روی آرایه خواهد داشت؟ آیا راه سریعتری وجود دارد؟

قسمت 5 : موارد خواسته شده را انجام دهید. نتایج به دست آمده و یافته‌های خود را در کادر زیر بنویسید.

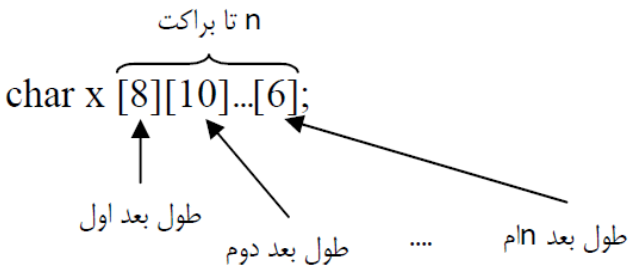
```c
#include <stdio.h>
#define array_size 10
int main() {
	int numbers[array_size] = { 3, 2, 5, 4, 1, 2, 3, 8, 3, 6 };
	float sum = 0;
	// Calculating mean
	for (int i = 0; i < array_size; i++) {
		sum += numbers[i];
	}
	printf("Mean: %f\n", sum / 10);
	// Finding answer indices
	int answer[array_size];
	int found = 0;
	for (int i = array_size - 1; i >= 0; i--){
		for (int j = i - 1; j >= 0; j--){
			if (numbers[j] > numbers[i]){
				answer[i] = j;
				found = 1;
				break;
			}
		}
		if (!found){
			answer[i] = -1;
		}
		found = 0;
	}
	printf("Answer: ");
	for (int i = 0; i < array_size; i++){
		printf("%d " , answer[i]);
	}
	printf("\n");
	return 0;
}
```

به ازای n تا ورودی در کد بالا در بدترین حالت با n^2 بار پیمایش جواب پیدا می شود. O(n^2)

راه سریع تری برای حل این سوال وجود دارد که در O(n) این کار را انجام می دهد یعنی با یکبار پیمایش آرایه جواب به دست می آید. این روش با استفاده از stack قابل پیاده سازی است.

آرایه های چند بعدی:

در زبان C می توان آرایه هایی با بیش از یک بعد نیز تعریف و استفاده کرد. نحوه ی تعریف یک آرایه از نوع کاراکتر با n بعد به صورت زیر است:



دقت کنید: حافظه ی کامپیوتر مانند یک آرایه ی یک بعدی است. لذا برای شبیه سازی آرایه هایی با ابعاد بیش تر سطرهای آن را پشت سر هم قرار می دهد و با استفاده از اشاره گر به آن ها دسترسی پیدا می کند.

به عنوان مثال اگر آرایه از جنس int است:

```
int a[3][4]={{11,12,14,15},{21,22,23,24}, {13,31,32,33};
```

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

همچنین می‌توانید به این شکل هم آرایه‌ی فوق را تعریف کنید:

```
int a[3][4]={11,12,14,15,21,22,23,24,13,31,32,33};
```

دقت کنید که همانند آرایه های یک بعدی شما فقط هنگام تعریف یک آرایه ی چند بعدی می توانید آن را به صورت فوق مقداردهی کنید.

6. انجام دهید!

مانند قسمت 4، برنامه را در debug mode اجرا کرده و عناصر حافظه را مشاهده کنید. در حلقه‌ی for دوم می‌بایست قطعه کد کوچکی بنویسید که عناصر آرایه‌ی با نام second_array را در آرایه‌ی با نام table بریزد. ضمن اینکه تفاوت هدف این قسمت با قسمت 4 در این است که قرار است که شما در قسمت 6 با نحوه‌ی ذخیره شدن داده در آرایه‌های دو بعدی آشنا شوید.

```
#include <stdio.h>
#define SIZE 5

int main() {
	int i, j;
	int table[SIZE][SIZE] = { { 1,2,3,4,5 },{ 6,7,8,9,10 },{ 11,12,13,14,15 },{
1,2,3,4,5 },{ 4,6,7,3,2 } };
	int second_table[SIZE][SIZE];

	for (int i = 0; i < SIZE; i++) {
		for (int j = 0; j < SIZE; j++) {
			second_table[j][i] = table[i][j];
		}
	} // transposing

	for (int i = 0; i < SIZE; i++) {
		for (int j = 0; j < SIZE; j++) {
			// Your code goes here
		}
	} // replacing

	for (i = 0; i < SIZE; i++) {
		for (j = 0; j < SIZE; j++)
			printf("%d ", table[i][j]);
		printf("\n");
	} // printing the result

	return 0;
}
```

قسمت 6 : نتایج به دست آمده را در کادر زیر بنویسید.

```c
#include <stdio.h>
#define SIZE 5
int main() {
	int i, j;
	int table[SIZE][SIZE] = { { 1, 2, 3, 4, 5 }, { 6, 7, 8, 9, 10 }, { 11, 12, 13,
14, 15 }, {
		1, 2, 3, 4, 5 }, { 4, 6, 7, 3, 2 } };
	int second_table[SIZE][SIZE];
	for (int i = 0; i < SIZE; i++) {
		for (int j = 0; j < SIZE; j++) {
			second_table[j][i] = table[i][j];
		}
	} // transposing
	for (int i = 0; i < SIZE; i++) {
		for (int j = 0; j < SIZE; j++) {
			table[i][j] = second_table[i][j];
		}
	} // replacing
	for (i = 0; i < SIZE; i++) {
		for (j = 0; j < SIZE; j++)
			printf("%d ", table[i][j]);
		printf("\n");
	} // printing the result
	return 0;
}
```

در آرایه های دوبعدی چند آدرس (خط) از حافظه در اختیار آرایه قرار داده می شود حال در این کد در حلقه اول خانه های آرایه به صورت عمودی یعنی ستونی پر می شوند ولی در حلقه دوم خانه های آرایه به صورت سطری پر می شوند. در کل پر شدن خانه های آرایه دوبعدی به شکل آرایه های یک بعدی است ولی با این تفاوت که به جای یک آرایه یک بعدی چند تا آرایه یک بعدی در کنار هم قرار گرفته اند و هر بار یکی از این آرایه های یک بعدی پر شده و به آرایه یک بعدی بعدی می رویم. مانند قسمت 4 با 3 خانه (00) فاصله عدد های جدید در خانه های حافظه قرار می گیرند ولی در اینجا به جای یک خط یا آدرس از حافظه چندین خط یا آدرس ازحافظه در اختیار آرایه قرار گرفته است.

بخش های اختیاری در یک فایل جداگانه بر روی سایت قرار گرفته است. توصیه می شود که حتماً آن ها را انجام دهید.

موفق باشید