به نام خدا

دستور کار آزمایشگاه شماره ۸-1

آشنایی با اشاره‌گرها

در این جلسه شما با اشاره‌گرها (pointer) و ارتباط آن‌ها با آرایه‌ها آشنا خواهید شد.

تعریف اشاره‌گر : اشاره‌گر یک متغیر است که حاوی آدرس یک متغیر دیگر در حافظه است. می‌توانیم به ازای هر نوع متغیر اشاره‌گر مخصوص به آن نوع متغیر را به صورت زیر تعریف کنیم :

```
Variable_Type *variable_name;
```

همچنین همانطور که می دانید می توانیم با استفاده از علامت & می توانیم به آدرس یک متغیر دسترسی پیدا کنیم.

```
int a = 1;
int *p;
p = &a;
```

و همچنین برای دسترسی به محتوای متغیری که اشاره‌گر به آن متغیر اشاره می‌کند از علامت * قبل از نام اشاره گر استفاده می‌کنیم.

```
//following the last code
int b = *p;
//the b value is equl to 1
```

۱. انجام دهید!

۱) قطعه کد مقابل را نوشته، کامپایل و اجرا نمایید. با قرار دادن breakpoint در برنامه در هر قسمت مقادیر خواسته شده را بر روی برگه بنویسید. **سپس هر کدام از این مقادیر را توجیه کنید.**

```
int main() {
	int x;
	int *ptr;
	int **ptr2;
		/* ۱. مقادیر x و ptr و *ptr را در این نقطه بنویسید. */
	x = 25;
	ptr = &x;
	ptr2 = &ptr;
```

```
        */ بنویسید.Debug را با استفاده از پنجره *ptr و ptr و ۲x. مقادیر /*
        *ptr = 2 * **ptr2;
        printf("x = %d and address of x = 0x%p = 0x%p = 0x%x = 0x%p \n", x, ptr, &x,
&x, *ptr2););
                */ ۳. مقدار خروجی را بنویسید. /*
        return 0;
}
```

برگه را به دستیاران آموزشی تحویل دهید.

نکته : هنگام کار با اشاره‌گرها باید بسیار دقت نمود زیرا ممکن است در صورت مقداردهی اشتباه به اشاره‌گر با خطای زمان اجرا مواجه شویم. قطعه کد زیر را اجرا کنید و نتیجه را مشاهده کنید.

```
int main(){

        int *ptr = 0x1;
        *ptr = 25;

        return 0;
}
```

آرایه‌ها و اشاره‌گرها:

رابطه ی نزدیکی بین اشاره‌گرها و آرایه‌ها وجود دارد. وقتی یک آرایه تعریف می کنید، آدرس اولین خانه‌ی آن در متغیر مربوطه ریخته می‌شود.[1] مثلاً اگر داشته باشیم int x[10] یک آرایه با ۱۰ خانه از نوع integer تعریف کرده‌ایم که آدرس اولین خانه‌ی آن در متغیر x ریخته شده است. حال به دو نکته زیر **توجه کنید**:

۱) برای دسترسی به محتوای یک خانه ی آرایه دو روش وجود دارد:
   a. از اندیس مربوطه استفاده کنیم. مثلاً x[6] (یعنی محتوای هفتمین خانه ی آرایه)
   b. به روش base + offset عمل کنیم: مثلاً *(x + 6) (یعنی محتوای هفتمین خانه ی آرایه)

۲) آدرس یک خانه ی آرایه نیز به طور مشابه به دو صورت می تواند بیان شود:
   a. از اندیس مربوطه استفاده کنیم. مثلاً &x[6] (یعنی آدرس هفتمین خانه ی آرایه)
   b. به روش base + offset عمل کنیم: مثلاً (x + 6) (یعنی آدرس هفتمین خانه ی آرایه)

۲. انجام دهید!

با توجه به کد داده شده در سمت چپ، دو قسمت جا افتاده در کد سمت راست را با استفاده از اشاره گرها کامل کنید.

[1] این آدرس ثابت و غیر قابل تغییر می باشد.

```c
#include <stdio.h>
#define SIZE 4
int main () {
    int i, sum = 0;
    int num[SIZE];
    printf("Enter %d numbers:\n", SIZE);
    for (i = 0; i < SIZE; i++)
        scanf("%d", &num[i]);
    for (i = 0; i < SIZE; i++)
        sum += num[i];
    printf("Sum: %d\n", sum);
    return 0;
}
```

```c
#include <stdio.h>
#define SIZE 4
int main () {
    int i, sum = 0;
    int num[SIZE];
    printf("Enter %d numbers:\n", SIZE);
    for (i = 0; i < SIZE; i++)
        scanf ("%d", ...); /* Complete this instruction */
    for (i = 0; i < SIZE; i++)
        sum += ...; /* Complete this instruction */
    printf("Sum: %d\n", sum);
    return 0;
}
```

نتیجه را به دستیاران آموزشی نشان دهید.

۳. فکر کنید!

در جلسه قبلی آزمایشگاه با این سوال که چگونه می‌توان دو آرایه را برابر یکدیگر قرار داد مواجه شدید. به نظر شما روش زیر پاسخ مناسبی برای این سوال است؟

```c
int main(){

      int arr[4] = { 1, 2, 3, 4 };
      int *arr_cpy;
      arr_cpy = arr;
      return 0;

}
```

پاسخ خود را توجیه کنید.

نتیجه را با دستیاران آموزشی در میان بگذارید.

نکته: رشته‌ها که با نام دیگر string در زبان c شناخته می‌شوند علاوه بر آرایه‌ای از متغیرهای char به صورت char* نیز می‌توانند نمایش داده شوند(که در وقاع معادل یکدیگرند). برای درک بیشتر این مطلب کد زیر را مشاهده کنید.

```c
char s1[10] = "Hello";
char* s2 = "Hello";
char* s3 = s1;
```

۴. انجام دهید!

نکته: همان‌طور که پیشتر نیز دیده‌اید یکی از مزایای استفاده از اشاره‌گرها پاس دادن متغیر ها به توابع به صورتی است که بتوان مقدار آن‌ها را در تابع تغییر داد.

نکته: یکی دیگر از مزایای استفاده از اشاره‌گرها پاس دادن آرایه ها به توابع است به صورتی که تنها نیاز است اشاره‌گر ابتدای آرایه را به تابع منتقل کرد. برای درک بهتر این مطلب به ساختار زیر توجه کنید. در هر دو ساختار زیر آرگومان ورودی تابع یک آرایه است.

```
int func(int *a);
int func(int a[]);
```

می خواهیم تابعی به نام compare بنویسیم که با گرفتن دو رشته ی first و second تعیین کند که آیا این دو رشته برابرند یا خیر؟ در صورت مساوی بودن مقدار true و در غیر این صورت مقدار false برگرداند.

برای این کار مراحل زیر را طی کنید:

۱) در تابع main دو رشته از کاربر بگیرید و آن ها را در آرایه هایی به طول ۷۰ بریزید.
یادآوری: برای خواندن رشته توسط تابع scanf به صورت زیر عمل کنید:

```
char my_string[70];

scanf ("%s", my_string);
```

۲) حال دو رشته را به عنوان ورودی به تابع compare دهید و مقدار بازگشتی را چاپ کنید.

راهنمایی:

```
bool compare(char first[], char second[]);
```

یک نمونه از اجرای برنامه فوق به صورت زیر است:

Input:
Hardware
Software
Output:

False

توجه: برای این که بررسی کنید ۲ آرایه برابرند یا نه، باید درون یک حلقه تمامی عناصر دو آرایه را نظیر به نظیر باهم مقایسه کنید. (در مورد رشته ها تا جایی پیش می رویم که به کاراکتر null یا پایان رشته برسیم.)

۳) در کتابخانه string.h مجموعه توابعی برای کار کردن با رشته ها نوشته شده اند که هم از نظر هزینه زمانی[۲] بهینه اند و هم کار شما در کار کردن با رشته ها را آسان می کند.تابعی معادل تابع compare که در این تمرین نوشتید در کتابخانه string.h پیدا کنید. مقدار بازگشتی این تابع چه تفاوتی با تابع شما دارد؟ (در Google عبارت string.h را جستجو کنید و توابع آن را بررسی کنید.)

نتیجه را به دستیاران آموزشی نشان دهید

۵. انجام دهید!(آرایه چند بعدی)

در جلسه گذشته با مفهوم آرایه‌ی چند بعدی آشنا شدید. در مورد رابطه‌ی آرایه‌های چند بعدی با اشاره‌گر مربوطه باید به این نکته توجه نمود که حافظه‌ی کامپیوتر مانند یک آرایه‌ی یک بعدی است. لذا برای شبیه‌سازی آرایه‌هایی با ابعاد بیش‌تر سطرهای آن را پشت سر هم قرار می‌دهد و با استفاده از اشاره‌گر به آن‌ها دسترسی پیدا می‌کند. به همین دلیل جنس (Type) یک آرایه‌ی دو بعدی از int معادل int** است. برای مثال کد زیر را در نظر بگیرید.

```
int a[2][2] = { {1,2},{3,4} };
printf("0x%p  0x%p", &a[0][0], &a[1][0]);
```

خروجی کد مورد نظر را مشاهده و توجیه کنید.

نتیجه را با دستیاران آموزشی در میان بگذارید.

۶. انجام دهید!

می‌خواهیم تابعی به نام Cyclic_Swap تعریف کنیم به صورتی که سه متغیر را دریافت کنید و به صورت چرخش مقادیر آن‌ها را جابه‌جا کند. (مقدار متغیر اول در متغیر دوم، مقدار متغیر دوم در متغیر سوم و مقدار متغیر سوم در متغیر اول قرار داده شود.) برای انجام این کار :

۱) تابع Cyclic_Swap را با خروجی void و سه ورودی از جنس int* تعریف کنید.

۲) عملیات جابه‌جایی گردشی را درون تابع انجام دهید.

[2] time complexity

۳) در تابع `main` کد آزمایش برنامه را بنویسید و با دریافت ورودی مناسب، خروجی مناسب را چاپ نمایید.

نتیجه را به دستیاران آموزشی نشان دهید.