



به نام خدا

دانشکده‌ی مهندسی برق و کامپیوتر دانشکده فنی دانشگاه تهران

مبانی کامپیوتر و برنامه‌نویسی



استاد : دکتر مرادی

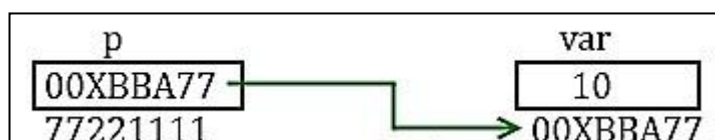
عنوان: آزمون‌گاه پنجم (اشاره‌گرها)

نیمسال دوم

99-98

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 فضای حافظه است.



می‌توانیم به ازای هر نوع متغیر اشاره‌گر مخصوص به آن نوع متغیر را به صورت زیر تعریف کنیم :

`Variable_Type *variable_name;`

همانطور که می‌دانید می‌توانیم با استفاده از علامت `&` به آدرس یک متغیر دسترسی پیدا کنیم. همچنین برای دسترسی به محتوای متغیری که اشاره‌گر به آن اشاره می‌کند، از علامت `*` قبل از نام اشاره‌گر استفاده می‌کنیم:

```
int a = 1;
int *p;
p = &a;
int b = *p; // b's value equals to 1
```

1. انجام دهید!



1) قطعه کد مقابل را نوشته، کامپایل و اجرا نمایید و با قرار دادن breakpoint در برنامه در هر قسمت مقادیر خواسته شده را مشاهده کنید:

```

int main() {
    int x;
    int *ptr;
    int **ptr2;
    /* مقدار موجود در اشاره گر ها و متغیر فوق را توجیه کنید */
    x = 25;
    ptr = &x;
    ptr2 = &ptr;
    /* اکنون مقادیر دو اشاره گر فوق نشان دهنده چه هستند؟ */
    *ptr = 2 * **ptr2;
    printf("x = %d and address of x = 0x%p = 0x%p = 0x%x = 0x%p \n",
x, ptr, &x, &x, *ptr2);
    /* مقدار خروجی را مشاهده کنید. */
    return 0;
}

```

نکته : هنگام کار با اشاره گر ها باید بسیار دقت کرد زیرا ممکن است در صورت مقداردهی اشتباه به اشاره گر با خطای زمان اجرا مواجه شویم.

به عنوان مثال، قطعه کد زیر را در debug mode اجرا و نتیجه را مشاهده کنید:

```

int main(){
    int *ptr = 0x1;
    *ptr = 25;
    return 0;
}

```

قسمت 1: نتیجه را برای دستیاران آموزشی توضیح دهید.

آرایه ها و اشاره گر ها:

رابطه ی نزدیکی بین اشاره گر ها و آرایه ها وجود دارد. وقتی یک آرایه تعریف می کنید، آدرس اولین خانه ی آن در متغیر مربوطه ریخته می شود. مثلاً اگر داشته باشیم `int x[10]` یک آرایه با 10 خانه از نوع `integer` تعریف کرده ایم که آدرس اولین خانه ی آن در متغیر `x` ریخته شده است. حال به دو نکته زیر توجه کنید:

1) 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)

2) 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)

2. انجام دهید!

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

```
#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;
}
```



```
#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;
}
```

قسمت 2: نتیجه را برای دستیاران آموزشی توضیح دهید.

3. فکر کنید!

چگونه می توان یک نسخه ی دیگر از یک آرایه داشت؟ به نظر شما روش زیر پاسخ مناسبی برای این سوال است؟ پاسخ خود را توجیه کنید.

```
int main(){

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

}
```

قسمت 3: 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 وقایع معادل یکدیگرند). برای درک بیشتر این مطلب کد زیر را مشاهده کنید.

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

4. انجام دهید!

می‌خواهیم 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:

1) در تابع **main** دو رشته از کاربر بگیرید و آن‌ها را در آرایه‌هایی به طول 70 بریزید.
یادآوری: برای خواندن رشته توسط تابع **scanf** به صورت زیر عمل کنید:

```
char first_array [70], second_array [70];  
scanf("%s", first_array);  
scanf("%s", second_array);
```

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

راهنمایی:

```
int compare(char* first_array, char* second_array);
```

یک نمونه از اجرای برنامه فوق به صورت زیر است:

Input:

Hardware

Software

Output:

False

توجه: برای این که بررسی کنید 2 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.)

به نظر شما، چگونه می توان تنها با یک پیمایش همزمان روی دو آرایه، هم طول و هم برابری کاراکترهای آنها را مقایسه کرد؟

قسمت 4: نتیجه را به دستیاران آموزشی نشان دهید.

نکته: در کتابخانه ای به نام `string.h` مجموعه توابعی برای کار کردن با رشته ها نوشته شده اند که هم از نظر هزینه زمانی¹ بهینه اند و هم کار شما در کار کردن با رشته ها را آسان می کند. (برای مطالعه ی بیشتر، می توانید عبارت `string.h` را در Google جستجو و توابع آن را بررسی کنید.)

← 5. انجام دهید!

نکته: همان طور که پیشتر نیز دیده اید یکی از مزایای استفاده از اشاره گرها پاس دادن متغیر ها به توابع به صورتی است که بتوان مقدار آن ها را در تابع تغییر داد.

نکته: یکی دیگر از مزایای استفاده از اشاره گرها پاس دادن آرایه ها به توابع است به صورتی که تنها نیاز است اشاره گر 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[]);
```

شما قبلا با مفهوم آرایه ی چند بعدی آشنا شده اید. در مورد رابطه ی آرایه های چند بعدی با اشاره گر مربوطه باید به این نکته توجه نمود که حافظه ی کامپیوتر مانند یک آرایه ی یک بعدی است. لذا برای شبیه سازی آرایه هایی با ابعاد بیش تر سطریهای آن را پشت سر هم قرار می دهد و با استفاده از اشاره گر به آن ها دسترسی پیدا می کند. به همین دلیل جنس (Type) یک آرایه ی دو بعدی از `int**` معادل `int` است.

در این قسمت بنا است تا معکوس یک ماتریس را بوسیله ی کار با آرایه های دو بعدی به این سبک حساب کنید. ابتدا فایل ضمیمه شده آزمایش را مشاهده نمایید. در این قسمت شما می بایست تا تابعی که در ابتدای فایل مذکور تنها اظهار آن آورده شده است، پیاده سازی کنید:

- تابع `calc_transposed_matrix`، با گرفتن یک آرایه به عنوان آرگومان اول و در نظر گرفتن آن به عنوان یک ماتریس، ترانپاده آن را وارد آرایه ای که به عنوان آرگومان دوم گرفته شده می نماید. توصیه می شود تمامی توابع را بررسی فرمایید.

¹ time complexity

قسمت 5: نتیجه را به دستیاران آموزشی نشان دهید.

6. 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 :

- 1) تابع `Cyclic_Swap` را با خروجی `void` و سه ورودی از جنس `int*` تعریف کنید.
- 2) عملیات جابه‌جایی گردشی را درون تابع انجام دهید.
- 3) در تابع `main` کد آزمایش برنامه را بنویسید و با دریافت ورودی مناسب، خروجی مناسب را چاپ نمایید.

قسمت 6: نتیجه را به دستیاران آموزشی نشان دهید.

موفق باشید