به نام خدا

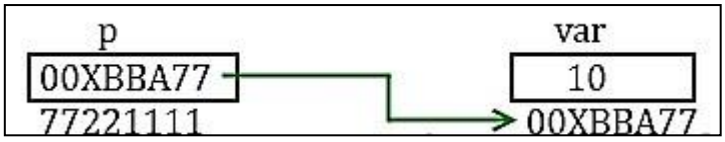
دانشکده‌ی مهندسی برق و کامپیوتر دانشکده فنی دانشگاه تهران

مبانی کامپیوتر و برنامه‌نویسی

نیمسال اول	عنوان:	استاد : دکتر مرادی،
۱۴۰۱-۱۴۰۲	اشاره‌گرها	دکتر هاشمی

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

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

۱. انجام دهید!

۱) قطعه کد مقابل را مشاهده کنید،سپس کامپایل و اجرا نمایید، مشکل کجاست؟ آن را اصلاح کنید.

```
int main() {
      int *ptr ;
      ptr = 20

printf("x = %d",*ptr);
      return 0;
}
```

نکته : هنگام کار با اشاره‌گرها باید بسیار دقت نمود زیرا ممکن است در صورت مقداردهی اشتباه به اشاره‌گر با خطای زمان اجرا مواجه شویم.

کد را به حالت زیر تغییر میدهیم، چه اتفاقی میافتد؟آن را توضیح دهید.

```
int main() {
      int *ptr ;
printf("x = %d", ptr);
      return 0;
}
```

قطعه کد مقابل را نوشته، کامپایل و اجرا نمایید و با قرار دادن دستور printf در برنامه در هر قسمت مقادیر خواسته شده را مشاهده کنید.

```
int main() {
      int x;
      int *ptr;
      int **ptr2;
      int **ptr3;

       /*مقدار موجود در اشاره‌گرها  و متغیر فوق را توجیه کنید */
      x = 25;
      ptr = &x;
      ptr2 = &ptr;
         /*اکنون مقادیر دو اشاره‌گر فوق نشان‌دهنده 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;
}
```

قسمت ۱ : نتیجه را برای دستیاران آموزشی توضیح دهید.

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. 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.

```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;
}
```

قسمت ۲ : نتیجه را برای دستیاران آموزشی توضیح دهید.

۳. فکر کنید!

چگونه می‌توان یک نسخه‌ی دیگر از یک آرایه داشت؟ به نظر شما روش زیر پاسخ مناسبی برای این سوال است؟ پاسخ خود را توجیه کنید.

```c
int main(){

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

}
```

قسمت ۳ : 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.

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

۴. انجام دهید!

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

۲) 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 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.)

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

قسمت ۴ : نتیجه را به دستیاران آموزشی نشان دهید.

۵. انجام دهید!

در کتابخانه‌ای به نام string.h مجموعه توابعی برای کار کردن با رشته ها نوشته شده اندکه هم از نظر هزینه زمانی[1] بهینه اند و هم کار شما در کار کردن با رشته ها را آسان می کند. (برای مطالعه‌ی بیشتر، می‌توانید عبارت string.h را در Google جستجو و توابع آن را بررسی کنید.)

حال میخواهیم برنامه ی قبل را با استفاده از کتابخانه string.h بنویسیم، و همچنین در صورتی که دو رشته برابر بودند طول آن را نمایش دهیم و در صورتی که برابر نبودند مقدار رشته ی دوم را برابر با رشته ی اول قرار دهیم،با استفاده از توابع کتابخانه string.h این برنامه را بنویسید.

راهنمایی:

برای مقایسه دو رشته:

int strcmp(const char *str1, const char *str2)

برای کپی کردن یک رشته:

char *strcpy(char *dest, const char *src)

برای بدست آوردن طول رشته:

size_t strlen(const char *str)

قسمت ۵ : نتیجه را به دستیاران آموزشی نشان دهید

۶. انجام دهید!

[1] time complexity

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[]);
```

شما قبلا با مفهوم آرایه‌ی چند بعدی آشنا شده‌اید. 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.

در این قسمت بنا است تا ترانهاده یک ماتریس را بوسیله‌ی کار با آرایه‌های دو بعدی حساب کنید.

- تابع `calc_transposed_matrix`، با گرفتن یک آرایه به عنوان آرگومان اول و در نظر گرفتن آن به عنوان یک ماتریس، ترانهاده آن را وارد آرایه‌ای که به عنوان آرگومان دوم گرفته شده می‌نماید.

قسمت ۶ : نتیجه را به دستیاران آموزشی نشان دهید.

موفق باشید