به نام خدا

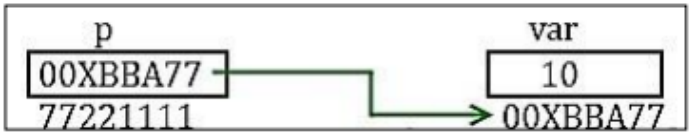
دانشکده‌ی مهندسی برق و کامپیوتر دانشکده فنی دانشگاه تهران

مبانی کامپیوتر و برنامه نویسی

نیمسال اول	عنوان:	اساتید:
1402-03	اشاره‌گرها	دکترمرادی، 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 می‌باشد.



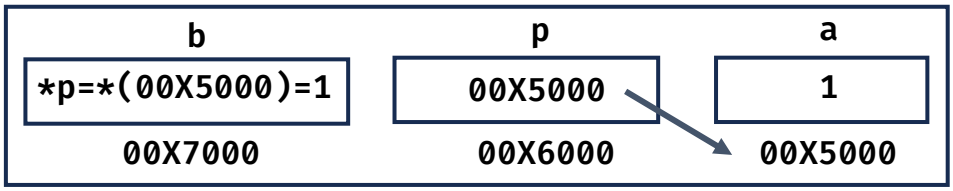
می‌توانیم به ازای هر نوع متغیر، اشاره‌گر مخصوص به آن نوع متغیر را به صورت زیر تعریف کنیم:

```
Variable_Type *variable_name;
```

همانطور که می‌دانید با استفاده از علامت & می‌توانیم 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
```

برای درک بیشتر مثال بالا، به شکل زیر توجه کنید:



1- انجام دهید!

- برنامه‌های زیر را کامپایل و اجرا کنید. سپس مقدار خروجی هر یک از برنامه‌ها را توجیه نمایید. علت را برای دستیاران آموزشی توضیح دهید.(قسمت 1)

```c
#include<stdio.h>
int main(){
    int *ptr = NULL;
    printf("%d\n", ptr);
    return 0;
}
```

```c
#include<stdio.h>
int main(){
    int *ptr = NULL;
    printf("%d\n", *ptr);
    return 0;
}
```

- حال قطعه کد زیر را در یک پروژه‌ی جدید اجرا کنید. چه خطایی دریافت می‌کنید؟ برنامه را طوری تغییر دهید تا مشکل رفع گردد. نتیجه را به دستیاران آموزشی نشان دهید.(قسمت 2)

```c
#include<stdio.h>
int main(){
    int *ptr ;
    ptr=20;
    printf("x= %d\n",*ptr);
    return 0;
}
```

- اکنون برنامه زیر را کامپایل کنید. به چه مشکلی برخوردید؟ مشکل را اصلاح کنید. چه نتیجه‌ای گرفتید؟ علت را برای دستیاران آموزشی توضیح دهید.(قسمت 3)

```c
#include<stdio.h>
int main(){
    float x,y;
    int *ptr ;
    x=12.34;
    ptr=&x;
    y=*ptr;
    printf("y= %f\n",y);
    return 0;
}
```

2- انجام دهید!

برنامه‌ی زیر را در یک پروژه‌ی جدید اجرا کنید. به طوری که هر بار، سطر خالی را که با هایلایت زرد مشخص شده‌است، با یکی از عبارات داده شده در جدول زیر تکمیل نموده و خروجی بدست آمده را تحلیل نمایید. علت را برای دستیاران آموزشی توضیح دهید.(قسمت 4)

```
#include<stdio.h>
int main() {
int x = 30 , y = 80 , *ptr1 , *ptr2;
ptr1=&x;
ptr2=&y;
//...
printf("x=%d , address_x=0x%p\ny=%d , address_y=0x%p\n
        *ptr1=%d , ptr1=0x%p\n*ptr2=%d , ptr2=0x%p\n",
        x , &x , y , &y , *ptr1 , ptr1 , *ptr2 , ptr2);
return 0;
}
```

الف	ب	ج	د	ه
ptr2 = ptr1;	*ptr2 = *ptr1;	x = y;	*ptr1 *= 3;	*ptr1 += *ptr2;

3- انجام دهید!

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.

```
#include<stdio.h>
int main() {
    int x= 25;
    int *ptr;
    int **ptr2;
    ptr = &x;
    ptr2 = &ptr;
    /* print the values of x , &x , *ptr , ptr , &ptr ,
     **ptr2 , *ptr2 , ptr2 and compare them! */
    *ptr = 2 * **ptr2;
    /* Now check the values of x , *ptr , **ptr2 */
    return 0;
}
```

مقادیر نمایش داده شده در خروجی را توجیه کنید.(قسمت 5)

❖ همانطور که در قسمت‌های قبل مشاهده کردید، یکی از روش‌های رفع خطاها در برنامه استفاده از تابع printf می‌باشد. توسط این تابع می‌توان مقادیر متغیرها و ... را در هربخش برنامه بررسی کرده و از بروز خطاهای احتمالی جلوگیری کرد. اما یک راه بهتر، استفاده از break points می‌باشد. در فیلم های آموزشی درس این روش شرح داده شده است. همچنین می‌توانید از این لینک استفاده کنید.

آرایه‌ها و اشاره‌گرها:

رابطه‌ی نزدیکی بین اشاره‌گرها و آرایه‌ها وجود دارد. وقتی یک آرایه تعریف می‌کنید، **آدرس اولین خانه‌ی** آن در متغیر مربوطه ریخته می شود. برای مثال اگر داشته باشیم int x[10] ، یک آرایه با 10 خانه از نوع integer تعریف کرده‌ایم که آدرس اولین خانه‌ی آن (&x[0]) در متغیر x ریخته شده است. حال به دو نکته زیرتوجه کنید:

1) فرض کنید می‌خواهیم به محتوای خانه‌ی سوم از یک آرایه دسترسی پیدا کنیم. برای این کار می توان از دورش زیر بهره گرفت:

   a. **استفاده از اندیس:** x[2] (یعنی محتوای سومین خانه‌ی آرایه)
   b. **استفاده از روش base + offset** : *(x+2) (یعنی محتوای سومین خانه‌ی آرایه)

2) حال فرض کنید قصد داریم به آدرس این خانه از آرایه دسترسی پیدا کنیم. به طور مشابه داریم:

   a. **استفاده از اندیس:** &x[2] (یعنی آدرس سومین خانه‌ی آرایه)
   b. **استفاده از روش base + offset** : (x+2) (یعنی آدرس سومین خانه‌ی آرایه)

نکته: رشته‌ها که با نام دیگر string در زبان c شناخته می‌شوند، علاوه بر آرایه ای از متغیرهای char به صورت char* نیز می‌توانند نمایش داده شوند. که در واقع اسم آرایه حاوی آدرس اولین خانه از آرایه می‌باشد. برای درک بیشتر این مطلب کد زیر را مشاهده کنید.

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

در مثال بالا، (s2+1) نشانگر آدرس خانه‌ی دوم این آرایه است ولی توسط *(s2+1) می‌توان به محتوای خانه‌ی دوم این آرایه یعنی حرف "e" دسترسی پیدا کرد.

نکته: 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 شبیه سازی 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.

4- انجام دهید!

با توجه به کد داده شده در سمت چپ، دو قسمت جا افتاده در کد سمت راست را با استفاده از اشاره‌گرها و به روش base + offset کامل کنید. سپس نتیجه را به دستیاران آموزشی نشان دهید.(قسمت 6)

```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",...);
    for(i=0; i<SIZE;i++)
        sum+= ...;
    printf("Sum: %d\n", sum);
    return 0;
}
```

در برنامه‌ی زیر قصد داریم نسخه‌ای دیگر از رشته‌ی str را در str_copy ذخیره سازی کرده و سپس در نسخه‌ی دوم، حرف 't' در کلمه‌ی "tehran" را به 'T' تبدیل کرده و در نهایت هر دو رشته را چاپ کنیم. ابتدا جای خالی را تکمیل کنید.

```c
#include<stdio.h>
int main() {
    char str[] = "University of tehran";
    char* str_copy;
    str_copy=str;
    ...='T'; //conversion
    printf("str= %s\nstr_copy= %s\n",str , str_copy );
    return 0;
}
```

آیا دو رشته‌ی نمایش داده شده در خروجی تفاوتی دارند؟ چرا؟ علت را به دستیاران آموزشی توضیح دهید.(قسمت 7)

5- انجام دهید!

حال می‌خواهیم برنامه‌ای بنویسیم که در آن توسط یک تابع، دو عدد از بزرگ به کوچک مرتب شوند. به برنامه‌ی زیر دقت کرده و آن را در یک پروژه جدید اجرا کنید.

```
#include<stdio.h>
void num_sorter(int first_num, int second_num){
    if (first_num<second_num){
        int temp=first_num;
        first_num=second_num;
        second_num=temp;
    }
}
int main(){
    int first_num=5, second_num=7;
    num_sorter(first_num,second_num);
    printf("first= %d\nsecond= %d", first_num, second_num);
    return 0;
}
```

- آیا این برنامه به‌درستی عمل می‌کند؟ چرا؟ علت را به دستیاران آموزشی توضیح دهید.(قسمت 8)
- این برنامه را با استفاده از اشاره‌گرها اصلاح کنید. نتیجه را به دستیاران آموزشی نشان دهید.(قسمت 9)

راهنمایی: لازم است تا آدرس دو متغیر را به صورت اشاره‌گر به عنوان ورودی به تابع بدهید و سپس ...

```
void num_sorter(int* first_num, int* second_num)
```

6- انجام دهید!

حال در این قسمت می‌خواهیم تابعی را پیاده سازی کنیم که آرایه‌ای به طول 10 از اعداد طبیعی را به عنوان ورودی دریافت کرده و هر یک از اعداد را با باقی مانده‌اش بر 2 جایگزین سازد. برنامه‌ی زیر را بدین منظور تکمیل کنید.

```
#include<stdio.h>
#define SIZE 10
void function(...){
    ...
}
int main(){
    float arr[SIZE]={1,2,3,4,5,6,7,8,9,10};
    function(...);
    for(int j=0;j<SIZE;j++){
        printf("%f ",arr[j]);
    } //printing the result
    return 0;
}
```

توجه: 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[]);
```

نتیجه را با دستیاران آموزشی در میان بگذارید.(قسمت 10)

موفق باشید.

تهیه و تنظیم: امیرمرتضی رضائی