مبانی برنامه‌نویسی

مدرس: سعدون عزیزی

s.azizi@uok.ac.ir

مرکز آموزش‌های الکترونیکی

تابستان ۹۶

سرفصل مطالب

- آشنایی با کامپیوتر و الگوریتم
- مقدمه‌ای بر برنامه‌نویسی C
- محاسبات
- ورودی/خروجی
- حلقه ها
- دستورات شرطی
- توابع
- آرایه‌ها
- کاراکترها و رشته‌ها
- اشاره‌گرها
- ساختار
- فایل‌ها

مقدمه

- روش دسترسی مستقیم به حافظه برای ذخیره سازی و بازیابی، استفاده از نام متغیر است
- اما گاهی لازم می شود که به جای نام، آدرس متغیر در اختیار برنامه نویس قرار گیرد تا از طریق آن دستیابی به محل مربوطه صورت گیرد
- روش دسترسی غیرمستقیم به مکانی از حافظه توسط اشاره گر انجام می شود
- متغیر اشاره گر متغیری است که محتوای آن آدرس یک متغیر دیگر است

معرفی (اعلان) متغیر اشاره گر

- متغیر اشاره گر به صورت زیر معرفی می گردد:

نوع متغیر اشاره شونده نام متغیر اشاره‌گر* ;

علامت * نشان دهنده اشاره گر بودن متغیر است

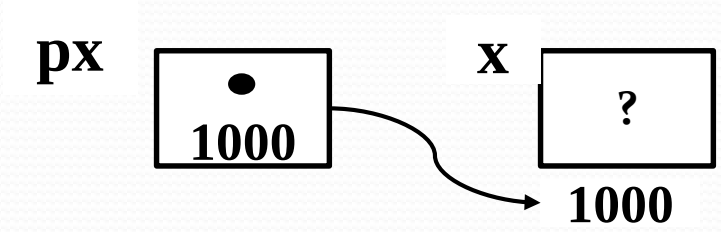
- **مثال:**

```
int   x, *px;
float  *p1, *p2;
double  y, *p3, z;
```

مقداردهی به اشاره گرها

- مثال:

```
int   x, *px;
px=&x;
```



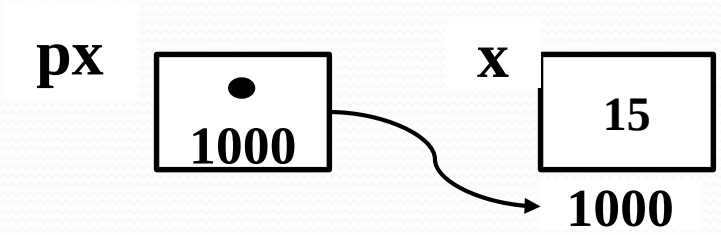
- مقداردهی اولیه

```
int   x, *px=&x;
```

کاربرد اشاره گرها

- مثال:

```
int   x, *px;
px = &x;
*px = 15;
```



کاربرد اشاره گرها

```
#include<stdio.h>
main()
{
      float a, b = 20, *pf;
      pf = &a;
      *pf = b + 5;
      b = *pf - 8;
      pf = &b;
      printf("a=%g , b=%g", a, *pf);
}
```

خروجی:

a=25 , b=17

اشاره گر و تابع

- فراخوانی با مقدار
- فراخوانی با ارجاع

توضیحات در محیط DeV-C++

ثابت اشاره گر و اشاره گر ثابت

توضیحات در محیط DeV-C++

جواب تابع از نوع اشاره گر

```
#include<stdio.h>
int *funct();
main()
{
	int *p;
	p=funct();
	printf("The first value of x inside funct is:%d",*p);
	p=funct();
	printf("\nThe second value of x inside funct is:%d",*p);
}
int *funct()
{
	static int x=1;
	x*=10;
	return &x;
}
```

خروجی:

```
The first value of x inside funct is:10
The second value of x inside funct is:100
```

استفاده از اشاره گر در عبارت

```
#include<stdio.h>
main()
{
        int *p1, *p2, x=12;
        p1=&x;
        p2=p1;
        printf("%d , %d",*p1,*p2);
}
```

خروجی:

12 , 12

استفاده از اشاره گر در عبارت

```
#include<stdio.h>
main()
{
        int x[5]={0,2,4,6,8};
        int *p1=&x[0], *p2=&x[3];
        p1++;
        p2-=2;
        printf("%d , %d",*p1,*p2);
}
```

خروجی:

2 , 2

رابطه میان اشاره گر و آرایه

- نام آرایه در واقع آدرس اولین عنصر آرایه را دربر دارد

```
int  a[6];
```

آنگاه a و &a[0] معادل هستند.

- **مثال:**

```
int  a[6], *p;

p = &a[0];  // p = a
```

- a[i] را می توان به صورت *(p+i) نیز نوشت.

اشاره گر و رشته

```
#include<stdio.h>
main()
{
        char *p = "The C Language";
        printf("%s", p);
}
```

خروجی:

The C Language

آرایه ای از اشاره گرها

```
#include<stdio.h>
main()
{
        int a[5][3]={11,12,13,21,22,23,31,32,33,41,42,43,51,52,53};
        int *p[5];
        int i;
        for(i=0; i<5; i++)
            p[i]=&a[i][0];
        printf("The elements of the second column are:\n");
        for(i=0; i<5; i++)
            printf("%d\t",*(p[i]+1));
}
```

خروجی:

12 22 32 42 52

اشاره گر به اشاره گر

```
#include<stdio.h>
main()
{
        float a=16.5;
        float *p=&a;
        float **q=&p;
        printf("The value of a is: %g\n",**q);
        **q=120.75;
        printf("The new value of a is: %g",*p);
}
```

خروجی:

```
The value of a is: 16.5
The new value of a is: 120.75
```

تخصیص حافظه به صورت پویا

- تابع malloc(n) به تعداد n بایت متوالی آزاد را در حافظه پیدا می کند و آن مجموعه را در اختیار گرفته و آدرس شروع آنها را تحویل می دهد
- تابع free(pointer) مجموعه بایت هایی که توسط تابع malloc در اختیار گرفته شده و اشاره گر pointer به ابتدای آن اشاره می کند را آزاد می نماید
- برای استفاده از توابع malloc و free باید فایل کتابخانه ای stdlib.h فراخوانی شود

تخصیص حافظه به صورت پویا (مثال)

```
#include<stdio.h>
#include<stdlib.h>
main()
{
      int *p, n, i;
      printf("Enter n:");
      scanf("%d",&n);
      p=(int *) malloc (n*sizeof(int));
      for(i=0; i<n; i++)
            *(p+i)=5*i;
      for(i=0; i<n; i++)
            printf("%d\t",*(p+i));
      free(p);
}
```

خروجی:

```
Enter n: 4
0        5        10       15
```