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

تابع چیست؟

- یکی از خصوصیات مطلوب در نوشتن یک برنامه ساخت یافته، آن است که سعی شود برنامه به صورت مجموعه ای از بخش های مستقل نوشته شود به گونه ای که هر بخش وظیفه پیاده سازی قسمتی از برنامه را بر عهده داشته باشد

- در زبان C هر یک از این بخش های مستقل را در قالب یک **تابع** ارائه می دهند

- در واقع هر تابع تکه برنامه ای است که برای حل بخشی از مسئله نوشته می شود

مزایای استفاده از تابع

- نوشتن برنامه را آسان تر می کند
- خطایابی و اشکال زدایی برنامه ساده تر می شود
- در وقت برنامه نویسی صرفه جویی می شود
- از نوشتن دستورات تکراری جلوگیری می شود و حجم برنامه کاهش می یابد
- همکاری میان گروه برنامه نویسان را ممکن می سازد

دسته بندی توابع

در حالت کلی، توابع به دو دسته تقسیم می شوند:

- **توابع کتابخانه ای:** توابعی که از قبل نوشته شده اند مانند sin(x)، exp(x)، printf(...)، scanf(...) و غیره
- **توابع غیرکتابخانه ای:** تابع که توسط برنامه نویس طراحی و پیاده سازی می شود

اجزای یک تابع

- **الگوی تابع:** شامل مشخص کردن نوع خروجی تابع، اسم تابع و آرگومان های تابع
- **تعریف تابع:** تکه برنامه مستقلی که وظیفه انجام قسمتی از برنامه را بر عهده دارد
- **فراخوانی تابع:** جهت اجرای تکه برنامه مورد نظر

```
#include<….>
function prototype          الگوی تابع
main()
{
        …
        function call       فراخوانی تابع
        …
}
function definition         تعریف تابع
```

اجزای یک تابع (مثال)

```
#include<stdio.h>
int fact(int);
int main()
{
        int k;
        printf("Enter an integer:");
        scanf("%d",&k);
        printf("The fact. of (%d) is: %d",k,fact(k));
        return 0;
}
int fact(int k)
{
        int i,f=1;
        for(i=1; i<=k; i++)
            f*=i;
        return f;
}
```

تابع بدون آرگومان (پارامتر)

```
#include<stdio.h>
int fact();
int main()
{
        printf("The fact. is: %d",fact());
        return 0;
}
int fact()
{
        int k,i,f=1;
        printf("Enter an integer:");
        scanf("%d",&k);
        for(i=1; i<=k; i++)
            f*=i;
        return f;
}
```

تابع بدون ارسال مقدار

```
#include<stdio.h>
void fact(int);
main()
{
	int k;
	printf("Enter an integer:");
	scanf("%d",&k);
	fact(k);
}
void fact(int k)
{
	int i,f=1;
	for(i=1; i<=k; i++)
		f*=i;
	printf("The fact. of (%d) is: %d",k,f);
	return;
}
```

تابع بدون پارامتر و بدون ارسال مقدار

```
#include<stdio.h>
void fact();
main()
{
	fact();
}
void fact()
{
	int k,i,f=1;
	printf("Enter an integer:");
	scanf("%d",&k);
	for(i=1; i<=k; i++)
		f*=i;
	printf("The fact. of (%d) is: %d",k,f);
	return;
}
```

متغیرهای محلی و سراسری

- متغیرهایی که در بدنه تابع معرفی می شوند، **متغیرهای محلی (local)** نامیده می شوند
- این گونه متغیرها فقط در همان تابعی که معرفی شده اند قابل استفاده هستند
- به طور کلی، متغیرهایی که در بلوک معرفی می شوند، متغیرهای محلی می باشند و فقط در همان بلوکی که معرفی شده اند و یا در بلوک های داخل آن بلوک، قابل رجوع هستند.
- متغیرهایی که خارج از توابع معرفی می شوند، توسط تمامی توابعی که از آن به بعد تعریف شده اند قابل استفاده اند. به این متغیرها، **متغیرهای سراسری (global)** می گویند

متغیرهای محلی و سراسری (مثال)

```
#include<stdio.h>
int fact();
int k;
main()
{
        printf("The fact. of (%d) is: %d",k,fact());
}
int fact()
{
        int i,f=1;
        printf("Enter an integer:");
        scanf("%d",&k);
        for(i=1; i<=k; i++)
                f*=i;
        return f;
}
```

کلاس های حافظه

- هر متغیر دارای یک کلاس حافظه می باشد که توسط آن حوزه دسترسی به آن و طول عمرش مشخص می گردد
- حوزه دسترسی به متغیر آن قسمت از برنامه است که در آن متغیر قابل استفاده است و منظور از طول عمر متغیر، مدت زمانی است که متغیر در حافظه وجود دارد (از زمان به وجود آمدن تا زمان از بین رفتن)
- در C چهار نوع کلاس حافظه وجود دارد:
  - **کلاس حافظه اتوماتیک (automatic)**
  - **کلاس حافظه ثبات (register)**
  - **کلاس حافظه خارجی (external)**
  - **کلاس حافظه ایستا (static)**

کلاس حافظه اتوماتیک

متغیری که دارای کلاس حافظه اتوماتیک است، خصوصیات زیر را دارد:

- فقط در همان بلوکی که معرفی شده است قابل دسترسی می باشد
- طول عمر آن به اندازه طول عمر بلوکی است که در آن معرفی شده است
- چنانچه هنگام معرفی به آن مقدار اولیه داده نشده باشد، مقدار آن نامشخص است
- هنگام معرفی متغیر از کلمه کلیدی **auto** استفاده می شود

```
auto int  m;
```

- متغیرهای محلی به صورت پیش فرض دارای کلاس حافظه اتوماتیک هستند

کلاس حافظه ثبات

برای افزایش سرعت انجام محاسبات، می توان متغیرها را در ثبات ها نگهداری کرد. متغیری که دارای کلاس حافظه ثبات است، خصوصیات زیر را دارد:

- فقط در همان بلوکی که معرفی شده است قابل دسترسی می باشد
- طول عمر آن به اندازه طول عمر بلوکی است که در آن معرفی شده است
- چنانچه هنگام معرفی به آن مقدار اولیه داده نشده باشد، مقدار آن نامشخص است
- هنگام معرفی متغیر از کلمه کلیدی **register** استفاده می شود

```
register int  count;
```

- معمولا نیازی به تعیین حافظه ثبات نیست، زیرا کامپایلرهای امروزی دارای این هوشمندی هستند که در مورد کلاس حافظه ثبات متغیرها تصمیم مناسب بگیرند

کلاس حافظه خارجی

کلاس حافظه متغیرهایی می تواند خارجی باشد که بیرون تابع معرفی شده باشند. متغیری که دارای کلاس حافظه خارجی است، خصوصیات زیر را دارد:

- در تمامی توابعی که بعد از معرفی آن قرار دارند، قابل دسترسی می باشد
- طول عمر آن تا پایان عمر کل برنامه است
- چنانچه هنگام معرفی به آن مقدار اولیه داده نشده باشد، مقدار آن صفر خواهد بود
- هنگام معرفی متغیر از کلمه کلیدی **extern** استفاده می شود

```
extern int  result;
```

- متغیرهای سراسری همین که معرفی شوند، خود به خود دارای کلاس حافظه خارجی هستند

کلاس حافظه خارجی (مثال)

file 1	file 2

#include<stdio.h>
void power();
int result=1;
main()
{
int n=10,i;
for(i=0; i<=n; i++) {
printf("%d ",result);
power();
}
}
	extern int result;
	void power()
	{
	result *= 2;
	}

خروجی:

1 2 4 8 16 32 64 128 256 512 1024

کلاس حافظه ایستا

متغیری که دارای کلاس حافظه خارجی است، خصوصیات زیر را دارد:

- چنانچه متغیر محلی باشد تنها در تابعی که معرفی شده است قابل دسترسی می باشد. اگر متغیر سراسری باشد، فقط در فایلی که معرفی شده است آن هم توسط توابعی که پس از آن تعریف شده اند قابل دسترسی است
- طول عمر آن تا پایان عمر کل برنامه است
- چنانچه هنگام معرفی به آن مقدار اولیه داده نشده باشد، مقدار آن صفر خواهد بود
- هنگام معرفی متغیر از کلمه کلیدی **static** استفاده می شود

```
static int  result=1;
```

- این نوع متغیرها آخرین مقدار خود را حفظ می کنند.

کلاس حافظه ایستا (مثال)

```
#include<stdio.h>
void power();
main()
{
        int n=10,i;
        for(i=0; i<=n; i++)
                power();
}
void power()
{
        static int result=1;
        printf("%d  ",result);
        result*=2;
}
```

خروجی:

1 2 4 8 16 32 64 128 256 512 1024

توابع بازگشتی (خودفراخوانی)

- خود فراخوانی به فرآیندی گفته می شود که طی آن از درون یک تابع همان تابع فراخوانی شود
- یک تابع بازگشتی تا زمانی که شرط خاصی برقرار نشده باشد، مرتبا خودش را فرا می خواند
- این روش محاسبه معمولا در محاسبات تکراری که هر محاسبه بر اساس جواب قبلی انجام می شود، به کار می رود
- برنامه بسیاری از مسائل پیچیده را که به صورت تکرار فرآیند حل می شوند، از این طریق می توان به فرم ساده تر و روان تر پیاده سازی کرد

توابع بازگشتی (مثال)

```
int fact(int  k)
{
    if (k<=1)
        return 1;
    return k*fact(k-1);
}
```