زبان برنامه‌سازی

C++

مؤلف : فرشید شیرافکن

دانشجوی دکترای بیوانفورماتیک دانشگاه تهران

ناشر: سازمان علمی آموزش فرادرس

بزرگترین پلتفرم آموزش آنلاین ایران

وب: www.faradars.org

تقدیم به:

روح پاک پدرم

- فرشید شیرافکن

سخن ناشر

در عین تمام نقدهای وارد شده به کنکور، هنوز راه حلی عملی که در جمیع جوانب، بهتر از سبک کوتاه و چند گزینه‌ای سؤالات باشد؛ ارائه نشده است. همین موضوع، کنکور را به ویژه کنکور کارشناسی ارشد به عنوان یک آزمون متمرکز و سراسری، از اهمیت دوچندانی برخوردار می‌کند.

یکی از آسیب‌های همراه با این آزمون سراسری این است که فضای رقابتی آن با ایجاد مؤسسات گوناگون، به سرعت از فضای یک رقابت علمی تبدیل به فضای رقابت اقتصادی می‌شود؛ به گونه‌ای که هزینه سرسام آور کلاس‌ها، دوره‌ها و منابع مرتبط با آزمون، از عهده بسیاری از دانشجویان خارج می‌شود. دانشجویانی که در عین استعداد تحصیلی بالا، در میدان رقابت مالی تحمیلی، در عین تمام شایستگی‌های خود، قدرت ادامه مسیر را از دست می‌دهند یا به نتیجه‌ای که در فضای مساوی مالی برای همه باید به آن می‌رسیدند، دست نمی‌یابند.

یکی از اهداف و آرمان‌های فرادرس به عنوان بزرگ‌ترین پروژه آموزش دانشگاهی اجرا شده بر بستر وب کشور، ایجاد دسترسی همگانی و یکسان به آموزش و دانش؛ مستقل از جغرافیا، زمان و سطح مالی دانشجویان بوده است. سیاست کاری فرادرس در راستای این آرمان، انتشار آموزش‌های ویدئویی تخصصی و دانشگاهی رایگان و یا بسیار کم هزینه، با تدریس مجرب‌ترین اساتید داخل و خارج کشور بوده است.

ما با انتشار رایگان این کتاب (به همراه نزدیک به ده کتاب رایگان دیگر) یکی از گام های دیگر خود را در راستای آرمان فرادرس برداشتیم. کتاب حاضر که حاصل نزدیک به یک دهه تدریس و پژوهش و تألیف مؤلف و مدرس فرادرس می‌باشد؛ در عین هزینه‌های بالای تألیف و آماده‌سازی، به جای انتشار و فروش؛ با تأمین مالی و سرمایه‌گذاری فرادرس به عنوان ناشر، به صورت کاملاً رایگان منتشر می‌شود. ما در گام‌های بعدی نیز تلاش خواهیم کرد که تا هر جا بتوانیم، حتی شده یک کتاب مرجع دیگر و بیشتر را با پرداخت هزینه، آزادسازی کرده و به صورت رایگان منتشر کنیم.

مؤلفین و ناشرینی که تمایل به واگذاری حق انتشار کتاب خود به فرادرس را دارند، می توانند با ایمیل ebooks@faradars.org مکاتبه نمایند. ما این کتاب‌ها را با پرداخت هزینه تألیف به مدرس و ناشر، به صورت رایگان منتشر خواهیم کرد تا همه دانشجویان مستقل از سطح مالی، به منابع مفید آزمون دسترسی داشته باشند. همچنین اگر ایده و نظری در خصوص کتاب‌های رایگان فرادرس داشته باشید، خوشحال می‌شویم که آن را با ایمیل ebooks@faradars.org مطرح نمایید.

سازمان علمی آموزش فرادرس

بزرگترین پلتفرم آموزش آنلاین ایران

وب: www.faradars.org

منبع مطالعاتی تکمیلی مرتبط با این کتاب

آموزش ویدئویی برنامه نویسی C++

عمومیت زبان C++ در میان زبان‌های برنامه‌نویسی بسیار بالا است و می تواند به عنوان اولین زبان نیز یاد گرفته شود و به پیش نیاز دیگر احتیاج نباشد.

مجموعه فیلم‌های آموزشی برنامه‌نویسی C++، با این فرض تهیه شده است که مخاطب هیچ دانش و تجربه قبلی در زمینه برنامه‌نویسی ندارد و در این مجموعه آموزشی، همه مباحث با بیان و تشریح مبانی نظری و سپس با پیاده‌سازی گام به گام مثال‌های عملی آموزش داده می‌شوند و از این نظر، در ایجاد یک دانش عمیق در زمینه برنامه‌نویسی، بسیار کارآمد است.

مدرس: مهندس فرشید شیر افکن

مدت زمان: ۲۰ ساعت

جهت مشاهده آموزش ویدئویی این آموزش – کلیک کنید faradars.org /fvcp9504

درباره مدرس

مهندس فرشید شیرافکن کارشناس ارشد مهندسی کامپیوتر گرایش نرم‌افزار است و در حال حاضر دانشجوی دکترای بیوانفورماتیک دانشگاه تهران هستند. ایشان از مدرسین نمونه در زمینه ارائه و آموزش دروس دانشگاهی انتخاب شده‌اند.

ایشان مشاور کنکور هستند و بیش از ۳۰ کتاب در زمینه کنکور رشته کامپیوتر تألیف نموده‌اند. ایشان در حال حاضر به عنوان یکی از برترین مدرسین فرادرس از جهت کمیت و کیفیت دروس ارائه شده، نزدیک به ۲۰ عنوان درسی را در قالب آموزش ویدئویی از طریق فرادرس منتشر کرده‌اند. این مجموعه دروس تا کنون مورد استفاده ده‌ها هزار دانشجوی سراسر کشور قرار گرفته اند.

مشاهده همه آموزش های تدریسی و تالیفی توسط مؤلف کتاب - کلیک کنید.

کتب رایگان دیگر از این مجموعه آموزشی

۱. آموزش نظریه زبان‌ها و ماشین - کلیک کنید (+)
۲. آموزش شیءگرایی در سی پلاس پلاس - کلیک کنید (+)
۳. آموزش پایگاه داده‌ها - کلیک کنید (+)
۴. آموزش ساختمان داده‌ها - کلیک کنید (+)
۵. آموزش سیستم عامل - کلیک کنید (+)
۶. آموزش ذخیره و بازیابی اطلاعات - کلیک کنید (+)

برای دانلود رایگان این مجموعه کتب، به لینک زیر مراجعه کنید:

http://faradars.org/computer-engineering-exam

دسته‌بندی موضوعی آموزش‌های فرادرس، در ادامه آمده است:

			
برنامه نویسی – کلیک (+)	آموزش های دانشگاهی و تخصصی – کلیک (+)	هوش مصنوعی و یادگیری ماشین – کلیک (+)	مهندسی برق الکترونیک و روباتیک – کلیک (+)
			
دروس دانشگاهی – کلیک (+)	مباحث مشترک – کلیک (+)	مهارت های دانشگاهی – کلیک (+)	نرم افزارهای تخصصی – کلیک (+)
			
مهندسی نرم افزار – کلیک (+)	نرم افزارهای عمومی – کلیک (+)	طراحی و توسعه وب – کلیک (+)	آموزش های عمومی – کلیک (+)

فهرست مطالب

فصل ۱ :

انواع داده ها – انواع عملگرها – دستورات ورودی و خروجی

مقدمه

زبان C در سال ۱۹۷۰ توسط دنیس ریچی طراحی گردید. این زبان، عناصر زبانهای سطح بالا را با خصوصیات تابعی زبان اسمبلی درهم آمیخته است و به همین علت زبان C را یک زبان میانی می‌نامند.

زبان c++ از شیوه برنامه نویسی شی ء گرا (OOP) ، استفاده می کند. در این روش از شیء (object) استفاده می شود. یک شیء دارای یک سری متغیر و توابع است که برای شی ء های دیگر شناخته شده نیست. زبان c++ در سال ۱۹۸۳ توسط استروستراپ ابداع شد و در سال ۱۹۹۴ توسط ANSI استاندارد شد. هر زبان oop دارای سه ویژگی مهم به قرار زیر است که در فصل های بعدی به آنها می پردازیم :

۱- **پلی مورفیسم** (Polymorphism)

استفاده از یک نام در چندین مورد مربوط به هم

۲- **کپسوله سازی** (Encapsulation)

کنار هم قرار دادن کد و داده هایی که این کدها با آنها کار می کنند و حفظ آنها از دخالتهای خارجی.

۳- **ارث بری** (Inheritance)

به دست آوردن خصوصیات یک شیء توسط شیء دیگر.

انواع داده ها

در زبان C پنج نوع داده اصلی وجود دارد که عبارتند از: void, double, float, int, char . برای ذخیره مقادیر صحیح از نوع int و برای مقادیر اعشاری از نوع double,float و برای ذخیره کاراکترها از نوع char استفاده می‌شود. در رابطه با void در فصل تابع و اشاره گر صحبت خواهد شد.

تذکر: با استفاده از کلماتی مانند signed, unsigned, long , short می توان نوع های دیگری نیز ساخت. تمامی این اصلاح کننده ها می توانند با نوع int به کار روند. بر روی نوع char اصلاح کننده های signed وunsigned و بر روی double فقط long را می توان به کار برد.

در زبان C++ داده ای به نام bool برای ذخیره مقادیر منطقی وجود دارد.

در c++ از نوع wchar_t برای ذخیره کاراکترهای ۱۶ بیتی می توان استفاده کرد.

جدول زیر انواع داده ها را نمایش می‌دهد:

نوع	اندازه(بایت)	بازه قابل قبول
char	1	127 تا -128
unsigned char	1	0 تا 255
int	2 یا 4	32767+ تا -32768
unsigned int	2 یا 4	0 تا 65535
long int	4	تقریباً ۲- میلیارد تا ۲+ میلیارد
float	4	
double	8	
long double	10	

- اندازه int در محیطهای ۳۲ بیتی برابر ۴ بایت است.
- نوع signed int همان int است.
- نوع signed char همان char است.
- نوع float دارای ۶ رقم دقت و نوع double دارای ۱۰ رقم دقت است.
- می توان چند اصلاح کننده را با هم ترکیب کرد. مثلا نوع unsigned long int یک عدد ۴ بایتی بدون علامت است.
- می توان کاری کرد که عدد کوچکی مانند ۲۰ به عنوان یک long محسوب شود. کافی است که از حرف L بعد از عدد به صورت long int a=20L; استفاده کنیم.

اعلان ثوابت

ثابت مقداری است که در طول اجرای برنامه قابل تغییر نمی باشند. برای اعلان ثابت به دو روش عمل می شود:

۱- توسط دستور const : const int x =5;

۲- توسط دستور #define : #define Y 2

مشاهده کردید که در انتهای دستور define نیازی به سمی کالون نمی باشد. دستور define باعث می شود که قبل از ترجمه برنامه توسط کامپایلر، مقدار ۲ به جای ثابت Y در برنامه قرار بگیرد و این دستور در زمان اجرا وجود ندارد.

تذکر: به ثوابتی که با دستور define تعریف می شوند، ماکرو می گویند و برای تفکیک آنها از متغیرهای برنامه، آنها را با حروف بزرگ نمایش می دهیم.

تبدیل نوع

وقتی که متغیرهایی با نوعهای مختلف در یک عبارت با یکدیگر ترکیب می‌شوند، نوع هایی با طول کوچکتر به نـوع هـایی بـا طول بزرگتر تبدیل می‌شوند. تبدیل نوع هم در عبارات و هم در احکام انتسابی ممکن است رخ دهـد. در تبـدیل نـوع ممکـن است اطلاعاتی از بین بروند. مثلاً وقتی یک متغیرint به یک متغیر char انتساب داده می‌شود، بایت کم ارزش متغیـر int بـه یک متغیر char منتقل شده ولی بایت با ارزش از بین می‌رود.

برای ذخیره سازی کاراکترها، کد اسکی آنها ذخیره می شود. مثلا با تعریف char ch='5'; مقدار ۳۵ (کد اسکی کاراکتر ۵) در متغیر ch ذخیره می شود.

*نوع نتیجه عبارت زیر را مشخص نمایید؟

```
float     f;
char      ch;
double  d;
(f+ch)-(f*d)
```

حل: نوع نتیجه عبارت، double است:

$$\underbrace{\underbrace{(f+ch)}_{\text{float}}-\underbrace{(f\times d)}_{\text{double}}}_{\text{double}}$$

* بعد از اجرای دستورات زیر چه مقداری در ch قرار می‌گیرد؟

```
int     i=65; char  ch;
ch= i;
```

حل: کاراکتر A قرار می‌گیرد.(کد اسکی کاراکتر A برابر ۶۵ است.)

عملگرها

عملگرها نمادهایی هستند که اعمال خاصی را انجام می دهند. انواع عملگرها عبارتند از :

۱- محاسباتی (+ , -, *, /, ٪ , -- , ++)

۲- رابطه ای (> , < , >=, <=, ==, !=)

۳- منطقی (&& , || , !)

۴- بیتی (& , | , ~, ^, >> , <<)

۵- عملگر کاما (,)

۶- عملگر شرطی (?)

۷- عملگر ترکیبی (ترکیب عملگرهای محاسباتی و عملگر =)

تذکر: البته عمگرهای دیگری مانند & ، * وجود دارد، که در مبحث اشاره گر ها بررسی می شوند.

مقدار صفر به منزله FALSE و هر مقدار غیر صفر به منزله TRUE در نظر گرفته می شود.

حاصل and (عملگر &&) دو عبارت وقتی درست است، که هر دو عبارت درست باشند و حاصل or (عملگر ||) دو عبارت وقتی درست است، که حداقل یکی از آنها درست باشد.

اولویت عملگرهای رابطه ای بالاتر از عملگرهای منطقی است.

جدول تقدم عملگرها

```
[ ]   ( )                                   بالاترین اولویت
!     ~     ++    --    *    &    sizeof
*     /     %
+     -
<<    >>
<     <=    >     >=
==    !=
&
^
|
&&
||
?
=     +=    -=    *=    /=
,                                           کمترین اولویت
```

مثالهایی از کاربرد عملگرهای محاسباتی

✱ بعد از اجرای دستورات زیر مقدار x و y چه خواهد شد؟

```
x  = 5;
y = --x;
```

حل: ابتدا مقدار x یک واحد کم شده و سپس در متغیر y ذخیره می شود. بنابراین در هر دو متغیر مقدار ۴ ذخیره می شود.

■

✱ بعد از اجرای دستورات مقدار x و y چه خواهد شد؟

```
x  = 5;
y = x--;
```

حل: ابتدا مقدار x به y منتقل شده و سپس یک واحد از x کم می‌شود. در نتیجه مقدار y برابر ۵ و مقدار x برابر ۴ خواهد شد.

✱ بعد از اجرای دستورات زیر مقدار x وy چه خواهد شد؟

```
x  = 5;
y = ++ x * x;
```

حل: ابتدا مقدار x برابر ۶ شده و سپس حاصل ۶×۶ در y قرار می گیرد. در نتیجه مقدار x برابر ۶ و مقدار y برابر ۳۶ خواهد شد.

✱ بعد از اجرای دستورات زیر مقدار x وy چه خواهد شد؟

```
x  = 5;
y = ++ x * ++x;
```

حل: ابتدا مقدار x برابر ۶ شده و سپس برابر ۷ شده و در نهایت حاصل ۷×۷ یعنی ۴۹ در y قرار می گیرد.

✱ بعد از اجرای دستورات زیر مقدار x وy چه خواهد شد؟

```
x  = 5;
y =  ++ x * x++;
```

حل: ابتدا مقدار x برابر ۶ شده و سپس حاصل ۶×۶ در y قرار گرفته و در نهایت مقدار x برابر ۷ خواهد شد.

✱ حاصل اجرای دستورات زیر در مقابل آنها نوشته شده است:

```
4 % 3 = 1  , 4 % -3 = 1 , -4 % 3 = -1 , -4 % -3 = -1
```

تذکر: علامت نتیجه عملگر % ، با علامت عملوند اول یکسان است و به علامت عملوند دوم بستگی ندارد.

مثالهایی از کاربرد عملگرهای بیتی

✱ حاصل 3 | 129 را بدست آورید؟(عملگر or بیتی)

```
129 = 10000001
  3 = 00000011
--------------------
      10000011=131
```

حل: نتیجه or برای ۲ بیت وقتی یک است که حداقل یکی از آنها یک باشد.

✱ حاصل 3 ^ 129 را بدست آورید؟ (عملگر xor بیتی)

```
129 = 10000001
  3 = 00000011
--------------------
      10000010=130
```

حل: نتیجه xor برای ۲ بیت وقتی یک است که آن دو بیت مانند هم نباشند.

■

✱ حاصل 3 & 129 را بدست آورید؟ (عملگر and بیتی)

```
129 = 10000001
  3 = 00000011
      ------------
      00000001=1
```

حل: نتیجه and برای ۲ بیت وقتی یک است که هر دو بیت یک باشند.

■

✱ بعد از اجرای دستورات زیر مقدار x چه خواهد شد؟ (عملگر شیفت به راست)

```
x = 96;
x=x  >> 2 ;
```

حل: هر شیفت به راست معادل تقسیم بر ۲ است. بنابراین ۹۶ بر ۴ تقسیم شده و حاصل برابر ۲۴ خواهد شد.

■

✱ بعد از اجرای دستورات زیر مقدار x چه خواهد شد؟(عملگر شیفت به چپ)

```
x= 14;
x=x<<3;
```

حل: هر شیفت به چپ معادل ضرب در ۲ است و سه شیفت به چپ معادل ضرب در ۸ است. بنابراین ۱۴ در ۸ ضرب شده و x برابر ۱۱۲ می شود.

■

عملگر شرطی

این عملگر با تست یک شرط، مقداری را به یک متغیر نسبت می‌دهد:

```
x = exp1 ? exp 2: exp3;
```

یعنی ابتدا عبارت exp1 ارزیابی می‌شود و اگر دارای ارزش درست بود، مقدار exp2 بعد از ارزیابی در x قرار می‌گیرد و اگر نادرست بود، مقدار exp3 بعد از ارزیابی در x قرار می گیرد.

✱ بعد از اجرای دستورات زیر چه مقداری در x قرا ر می‌گیرد؟

```
x = 5 > 2 ?  100: 20 ;
```

حل: حاصل عبارت 5>2 ، درست است و مقدار عبارت اول یعنی ۱۰۰ در x قرار می گیرد.

■

عملگر کاما

برای انجام چند عمل در یک دستور از عملگر کاما باید استفاده کرد :

```
x = (exp1 , exp2);
```

ابتدا exp1 ارزیابی شده و سپس نتیجه ارزیابی exp2 به متغیر x منتقل می‌شود. در این نوع موارد معمولاً exp1 و exp2 با یکدیگر ارتباط دارند. بعد از اجرای دستور زیر، مقدار ۵ در متغیر x قرار می‌گیرد.

```
x = (a=2 , a+3);
```

✱ حاصل ارزیابی عبارت زیر چیست؟

```
9 > 4 || !(5 <7 )  &&  8 <=9
```

حل: حاصل ارزیابی عبارت ، (True) است:

$$\underbrace{9>4}_{T} \;\|\; \underbrace{!(5<7)}_{F} \;\&\&\; \underbrace{8<=9}_{T}$$

$$\underbrace{\qquad\qquad}_{F}$$

$$T$$

■

کوتاه نویسی

می‌توان از کوتاه نویسی استفاده کرد. مثلاً به جای دستور x= x+3; از دستور x+ =3; استفاده کرد.

✱ با فرض x=5 و y=2 ، مقدار متغیر y بعد از اجرای دستور ; y *= x-2 چه خواهد شد؟

حل: در واقع عبارت به صورت زیر ارزیابی می‌شود و حاصل برابر ۶ خواهد شد:

```
y = y* (x-2) = 2*(5-2)=6
```

■

دستورات cin , cout

در c++ ، برای خواندن اطلاعات از ورودی از دستور cin و برای نمایش اطلاعات در خروجی از cout استفاده می شود. به عبارتی به جای استفاده از دستور scanf از دستور cin و به جای استفاده از دستور printf از دستور cout استفاده می شود.

دو دستور خواندن زیر معادل می باشند:

```
scanf("%d%d",&x,&y);
cin>>x>>y;
```

دو دستور چاپ زیر معادل می باشند:

```
printf("%d%d",x,y);
cout<<x<<y;
```

دستور cin تا رسیدن به space ، داده ها را از ورودی می خواند.

دستور cin.get( ) تا رسیدن به enter و یا کاراکتر مشخص شده، داده ها را از ورودی می خواند. تابع get( ) ، عضو شیء cin است.

مثالهایی برای cin :

✱اگر از ورودی ↵ ali reza وارد شود، فقط رشته ali در متغیر s ذخیره می شود:

```
char s[10]; cin>>s;
```

اگر بخواهید کل رشته ali reza در متغیر s ذخیره شود، باید از دستور cin.get استفاده کرد.

✱ اگر از ورودی ↵ abcd وارد شود، فقط رشته ab در متغیر s ذخیره می شود:

```
char  s[10]; cin.get(s,10,'c');
```

✱ در صورت ورود کاراکتر A ، کد اسکی آن یعنی 65 در x ذخیره می شود:

```
int  x;  x=cin.get( );
```

✱در صورت ورود رشته ↵ ali، فقط کاراکتر اول آن یعنی a در متغیر s ذخیره می شود:

```
char  s; s=cin.get( );
```

✱ در صورت وارد کردن Enter خروجی 10 است :

```
cout<<cin.get( );
```

تذکر: پس از وارد کردن ctrl+z ، خروجی 1- می باشد.

■

✱ در برنامه ی روبرو، پس از وارد کردن سه Enter خروجی کدام است؟

```
cout<<(cin.get( ) +cin.get( ) +cin.get( ) );
```

حل : خروجی 30 می باشد.

تذکر: بعد از وارد کردن یک ctrl+z ، خروجی 3- می باشد.

■

مثالهایی برای cout :

* توسط دستور زیر یک رشته ali چاپ شده و در خط بعد رشته reza می باشد:

```
cout<<"ali"<<endl<<"reza";
```

تذکر: به جای endl می توان از "\n" استفاده کرد.

* توسط دستور زیر رشته ok با ۴ فاصله قبل از آن چاپ می شود:

```
cout.width(6);
cout<<"ok";
```

تذکر: اگر از دستور ;('-')cout.fill قبل از دستورات بالا استفاده کنید، به جای فاصله خالی، کاراکتر خط تیره چاپ خواهد شد.

* خروجی دستور زیر 1.20000 می باشد:

```
cout.setf(ios::showpoint);
cout<<1.2;
```

تذکر: اگر از دستور اول استفاده نشود، خروجی 1.2 خواهد بود.

* برای چاپ عدد ۱۰ در مبنای ۱۶ (یعنی A) از دستور زیر استفاده می کنیم:

```
cout<<hex<<10;
```

البته می توان از دستورات زیر نیز استفاده کرد:

```
cout.setf (ios::hex);
cout<<10;
```

* برای چاپ عدد ۱۰ در مبنای ۸ (یعنی ۱۲) از دستور زیر استفاده می کنیم:

```
cout<<oct<<10;
```

* خروجی دستور زیر 1- می باشد:

```
cout<<EOF;
```

* خروجی چه می باشد؟

```
char  c=EOF;
cout<<c<<'a';
```

حل: چاپ کاراکتر **a** با یک فاصله خالی قبل .

* خروجی چه می باشد؟

```
cout<<EOF<<'a';
```

حل: خروجی 1a- می باشد.

تذکر: تابع ( )put ، عضو شیء cout است. خروجی دستور ;('a')cout.put برابر a می باشد.

دستور cin، شیئی از کلاس istream و دستور cout، شیئی از کلاس ostream می باشد. هنگام استفاده از این دستورات باید فایل <iostream.h> را در ابتدای برنامه include کرد.

عبارت دستور cout از راست به چپ پردازش شده و از چپ به راست نوشته می شود.

✱ خروجی چیست؟

```
int  x=3;
cout<<(x=0);
```

حل: خروجی 0 است.(اگر پرانتز برداشته شود، خطا دارد.)

■

✱ خروجی چیست؟

```
int  x=3;
cout<<(x=0)<<(x==0);
```

حل: خروجی 00 است.(اگر پرانتزها برداشته شود، خطا دارد.)

■

✱ خروجی چیست؟

```
int  x=0;
cout<<(x=0)<<(x==0);
```

حل: خروجی 01 است.

■

✱ خروجی چیست؟

```
int  x=4;
cout<<(x==0)<<(x=0);
```

حل: خروجی 10 است.

■

✱ خروجی چیست؟

```
int  x=0;
cout<<(x==0)<<(x=0);
```

حل: خروجی 10 است.

■

در جدول زیر کاراکترهای کنترلی که در cout می توان از آنها استفاده کرد، آورده شده است:

کاراکتر	عملکرد	کاراکتر	عملکرد
\n	انتقال کنترل به خط بعد	**\b**	حذف کاراکتر قبلی
\r	انتقال کنترل به اول خط	\"	چاپ دابل کوتیشن (")
\t	انتقال کنترل به ۸ محل بعدی	\\	چاپ کاراکتر \
\v	انتقال کنترل به ۸ سطر بعد	\:	چاپ کاراکتر :

✱ خروجی هر دستور در مقابل آن نوشته شده است:

```
cout<<"abc"'<<'\r'<<'d';     ⇒  dbc
cout<<"abc"'<<'\b'<<'d';     ⇒  abd
cout<<'a'<<'\t'<<'b';        ⇒  a    b
```

■

✱ خروجی چه می باشد؟

```
cout<<strlen("'\n'");
```

حل: تابع strlen طول رشته ورودی را محاسبه می کند. بنابراین خروجی 3 می باشد. (\n یک کاراکتر محسوب می شود.)

■

توابع کتابخانه‌ای

از توابع برای انجام عملیات ریاضی، کاراکتری، رشته ای، مقایسه ای، تبدیل نوع و غیره استفاده می‌شود.

نام تابع	عملکرد
abs(x)	محاسبه قدر مطلق عدد صحیح x
fabs(x)	محاسبه قدر مطلق عدد اعشاریx
sqrt(x)	محاسبه $\sqrt{x}$
pow(x,y)	محاسبه x^y
ceil	محاسبه کوچکترین عدد صحیح بزرگتر یا مساوی با عدد ورودی (سقف)
floor	محاسبه بزرگترین عدد صحیح کوچکتر یا مساوی با عدد ورودی (کف)
exp(x)	محاسبه e^x (که e همان عدد نپر است)
log(x)	محاسبه لگاریتم در مبنای e یعنی lnx
log10(x)	محاسبه لگاریتم در مبنای 10 یعنی logx
fmod(x,y)	محاسبه x mod y
modf	جدا کردن قسمت صحیح و اعشاری یک عدد
ldexp(a,b)	محاسبه $a \times 2^b$
tolower(ch)	تبدیل کاراکتر ch به حروف کوچک انگلیسی
toupper(ch)	تبدیل کاراکترch به حروف بزرگ انگلیسی
isalnum(ch)	اگر کاراکتر ch از حروف الفیا یا ارقام باشد، مقداری غیر صفر را بر می گرداند.
isalpha(ch)	اگر کاراکتر ch از حروف الفبا باشد، مقداری غیر صفر را بر می‌گرداند.
isascii(ch)	اگر کاراکتر ch در بازه صفر تا ox7f باشد، مقداری غیر صفر را بر می‌گرداند
isdigit(ch)	اگر کاراکتر ch یکی از ارقام ۰ تا ۹ باشد، مقداری غیر صفر را برمی‌گرداند
islower(ch)	اگر کاراکتر ch از حروف کوچک a تا z باشد، مقداری غیر صفر را برمی‌گرداند.
isupper(ch)	اگر کاراکتر ch از حروف بزرگ A تا Z باشد، مقداری غیر صفر را برمی‌گرداند.
isspace(ch)	اگر کاراکتر ch از فضاهای خالی باشد، مقداری غیر صفر را برمی‌گرداند.

atof(s)	تبدیل رشته عددی s به float
atoi(s)	تبدیل رشته عددی s به integer
atol(s)	تبدیل رشته عددی s به long

*خروجی هر تابع، در مقابل آن نوشته شده است:

pow (10,2) =100 ceil (9.8) =10 ceil (-9.8) =-9 floor (9.8) =9

floor (-9.8) =-10 log10(100) =2 log(exp(x)) =x ldexp(4,3)=4* 2^3 =32

مثال های حل شده در مجموعه آموزشی فرادرس

```
#include <iostream>
#include <conio.h>
using namespace std;
main( )
{
      cout<<"hello"<<'\a';
}
//////////////////////////
main()
{
 int  a;
 cout<<"enter a:";
 cin>>a;
 cout<<2*a;
}
//////////////////////////
main()
{
 int a,b;


 cout<<"enter a:";
 cin>>a;


 cout<<"enter b:";
 cin>>b;


 cout<<a+b;
}
//////////////////////////
main()
{
 char  ch;
 ch='A';
 cout<<ch+1;
 }
//////////////////////////
main()
{
```

```
 int a,b;
 a=5;
 b=4;
 cout<<++a*--b;
}
//////////////////////////
main()
{
 int a,b;
 a=2;
 b= ++a * a++;
 cout<<b;
}
//////////////////////////
main()
{
 int a=2;
 int b;
 b=++a * a;
 cout<<a;
 cout<<endl;
 cout<<b;
}
//////////////////////////
main()
{
 int  a=4;
 int b;
 b= --a * a--;
 cout<<b;


}
//////////////////////////
main()
{
 int  a=2;
 int  b=3;
 cout<<(a<b);
}
//////////////////////////
```

```
main()
{
 bool  a,b;
 a=true;
 b=false;
 cout<<(a || b );
 }
//////////////////////
main()
{


 int  x,y;
 x=8;
 y=x<<1;
 cout<<y<<endl;
 int z;
 z=x>>1;
 cout<<z;
 }
//////////////////////
main()
{
 int x=6;
 int y=0;
 int z=x && y;
 cout<<z;
 }
//////////////////////
main()
{
 int  x=8;
 int  y=4;
 z= x>y  ?  x : y;
 cout<<y;
 }
//////////////////////
main()
{
 int x=2,y;
 y=(x=10,x+4,9);
 cout<<x;
```

```
 cout<<endl;
 cout<<y;
 }
//////////////////////
main()
{
 int  i;
 char  ch;
 float  f;
 double  d;
 result=(i+ch) *(f-d);
 }
//////////////////////
main()
{
 int  r;
 float  x;


 cout<<"enter the radius:";
 cin>>r;


 x=PI*r*r;
 cout<<"area="<<x;
 }
//////////////////////
main()
{
 int x,y;
 float ave;


 cin>>x;
 cin>>y;


 ave=(float)(x+y)/2;
 cout<<ave;
 }
//////////////////////
main()
```

```
{
  int x=5;
  cout.width(3) ;
 // cout.setf(ios::left);
  cout.fill('-');
  cout<<x;


  cout<<endl;


  cout.width(8) ;
  cout.setf(ios::right);
  cout.fill('*');
  cout<<"ali";
 }
//////////////////////////
main()
{
  cout<<INT_MAX;
  cout<<endl;
  cout<<INT_MIN;


  cout<<endl<<endl;


  cout<<SHRT_MAX;
  cout<<endl;
  cout<<SHRT_MIN;
  cout<<endl<<endl;
  cout<<LONG_MAX;
  cout<<endl;
  cout<<LONG_MIN;
  }
//////////////////////////
main()
{
  int x=9;
  int y=2;
  cout<< x/y;
```

```
 cout<<endl;


 float a=9;
 float b=2;
 cout<< a/b;


cout<<endl;


 int  c=9;
 float d=2;
 cout<< c/d;
 }
//////////////////////////
main()
{
 bool  x,y;
 x=true;
 y=false;
 cout<<(x && y);
 }
//////////////////////////
main()
{
 cout<<pow(2,3)<<endl;
 cout<<pow(2,-2)<<endl;


 cout<<endl<<endl;


 cout<<fmod(6,4)<<endl;
 cout<<fmod(-20,7)<<endl;


 cout<<endl<<endl;


 cout<<abs(-2)<<endl;
 cout<<fabs(-3.5)<<endl;
```

```
 cout<<endl<<endl;


 cout<<floor(3.1)<<endl;
 cout<<ceil(3.1)<<endl;


 cout<<endl<<endl;


cout<<log10(100)<<endl;
 cout<<log(2.7)<<endl;


 cout<<tolower('A')<<endl;
cout<<toupper('a')<<endl;


 cout<<endl<<endl;


 cout<<islower('B')<<endl;
 cout<<isupper('B')<<endl;


 cout<<endl<<endl;



 cout<<int('a')<<endl;
 cout<<char(97)<<endl;
}
//////////////////////
main()
{
 int i;


 i=1;
 cout<<++i<<"   "<<++i<<endl;
```

```
i=1;
cout<<++i<<"  "<<--i<<endl;

i=1;
cout<<i<<"  "<<++i<<endl;

 i=1;
cout<<i<<"  "<<i++<<endl;

i=1;
cout<<++i<<"  "<<i<<endl;

i=1;
cout<<++i<<"  "<<i++<<endl;

i=1;
cout<<i++<<"  "<<++i<<endl;

i=1;
cout<<i++<<"  "<<++i<<endl;

i=1;
cout<<i<<"  "<<++i<<"  "<<++i<<endl;

i=1;
cout<<++i<<"  "<<i++<<"  "<<--i<<endl;

i=1;
cout<<i<<"  "<<i++<<"  "<<--i<<endl;
```

```
    i=1;
    cout<<++i<<"   "<<i++<<"   "<<--i<<endl;
}
```

فصل ۲ :

دستورات شرطی و تکرار

ساختارهای کنترلی عبارتند از : ترتیب ، انتخاب وتکرار ، که ساختارهای انتخاب خود بر ۳ نوع می باشند:

۱- تک انتخابی (if)

۲- دو انتخابی(if/else)

۳- چند انتخابی (switch)

توسط ابزار ساختار تصمیم می‌توان تحت شرایطی ، مجموعه ای از دستورات اجرا شوند و مجموعه ای دیگر از دستورات اجـرا نشده و کنترل اجرای برنامه به جای دیگر منتقل شود.

ساختار تصمیم if

این ساختار به صورتهای زیر می‌تواند مورد استفاده قرار گیرد:

۱)

```
if   (شرط)
    دستور ;
```

۲)

```
if   (شرط)
{
  مجموعه دستورات
}
```

۳)

```
if   (شرط)
{
  مجموعه دستورات ۱
}
else
{
  مجموعه دستورات ۲
}
```

* دستور ضرب تنها وقتی اجرا می‌شود که x برابر صفر نباشد:

```
if (x!= 0)
```

```
   p=p*x;
```

■

✱ توسط دستور زیر اگر x> = 0 عبارت positive و اگر x<0 ، عبارت negative چاپ می‌شود:

```
if  (x >=0)
    cout<< "positive";
else
    cout<< "negative";
```

■

شرط مقابل دستور if باید داخل پرانتز قرار بگیرد.

به جای دستور if(a==0) می توان از دستور if(!a) استفاده کرد.

✱ در صورتی که a برابر ۱۵ باشد، چه مقداری در b قرار می‌گیرد؟

```
b = 25;
if (a != (b-10))
     b=b–10;
else
     b = b%2;
```

حل : با قرار دادن مقادیر a و b در عبارت مقابل if متوجه می‌شویم که شرط برقرار نیست و دستور قسمت else اجرا می‌شود. بنابراین مقدار b برابر ۱ خواهد شد.

■

✱ در دستور if زیر، بعد از شرط و بعد از کلمه else، دستورهای مرکب وجود دارد:

```
if(x > y){
    x=x+1;
   cout<<"x  Bigger \ n";
}
else
{
  cout<<"x smaller \ n";
  cout<<y;
}
```

✱ خروجی برنامه زیر چیست؟

```
main( ){
   int x=0;
   if (x)       cout<<'A';
   if (!x)      cout<<'B';
   if (x!=0)   cout<<'C';
   if (x==0)  cout<<'D';
```

}

حل: خروجی BD است.دستور if اول و سوم اجرا نمی شوند، چون نتیجه شرط داخل if نادرست است.(عدد صفر معادل false و هر عدد غیر صفر معادل true می باشد.)

■

✱ در دستورات زیر اگر مقدار b برابر ۲ باشد، چه مقداری در a ذخیره می‌شود؟

```
if (b<10)
   if (b>=5)
      a=2*b;
   else
      a=3*b;
else
      a=4*b;
```

حل: شرط if اول برقرار است. ولی شرط if دوم برقرار نیست. بنابراین دستور a=3*b; اجرا می‌شود و مقدار a برابر ۶ خواهد شد. می‌توان دستورات را بصورت زیر بیان کرد:

$$a=\begin{cases}2*b & 5\le b<10\\ 3*b & b<5\\ 4*b & b\ge 10\end{cases}$$

■

✱ دستورات if زیر را با if تودرتو بنویسید.

```
if(x>0)
   a=a+1;
if(x<0)
    b=b+1;
if(x==0)
    c=c+1;
```

حل: استفاده از دستورات if تودرتو باعث بالا رفتن کارایی و خوانایی برنامه خواهد شد.

```
if(x>0)
    a=a+1;
else
     if(x<0)
        b=b+1;
     else
        c=c+1;
```

■

✱ اگر مقدار اولیه x برابر ۱ باشد، آنگاه مقدار نهایی x را در دوحالت الف و ب پیدا کنید:

```
if (x>=0)              if  (x>=0)
    x=x+1;                 x=x+1;
```

```
if(x>=1)                else  if (x>=1)
  x=x+2;                        x=x+2;
```

(الف) (ب)

حل: مقدار نهایی x در قسمت الف برابر ۴ و در قسمت ب برابر ۲ خواهد بود.

■

*جدول زیر را به کمک دستور if پیاده سازی کنید.

خروجی	محدوده x
A	۵۰ یا کمتر
B	۵۱ -۷۰
C	۷۱ – ۹۰
D	۹۱ به بالا

حل:

```
if (x <=50)
     cout<<'A';
else if  (x <=70)
         cout<<'B;
     else if  (x <=90)
              cout<<'C';
           else
              cout<<'D';
```

■

* دستور if زیر را به کمک عملگر شرطی بنویسید.

```
if(a>0  &&  a<10)
 {
   a=a+1;
   c=c+a;
 }
else
   c=c+a/b;
```

حل:

```
c += (a>0 &&a<10) ? ++a : a/b;
```

■

دستور switch

ساختار این دستور به صورت زیر است :

```
switch  (عبارت کنترلی)
 {
```

```
    case  'ثابت۱ ' : statement1;  break;
    case  'ثابت۲ ' : statement2;  break;
    :
    defult : statement n;
}
```

* در دستورات زیر، اگر مقدار متغیر کاراکتری color برابر 'R' باشد، Red، اگر برابر 'B' باشد، Bule و اگر برابـر 'Y' باشـد، Yellow چاپ می‌شود.

```
switch (color)
 {
    case 'R'  :  cout<< "Red";      break;
    case 'B'  :  cout<< "Blue";     break;
    case 'Y'  :  cout<< "Yellow"; break;
 }
```

■

استفاده از مقادیر double یا رشته ، برای برچسب های case، مجاز نمی باشد.

اگر مقدار عبارت کنترل کننده دستور switch ، در خارج از مجموعه مقادیر بر حسب case قرار داشته باشـد، تعریـف بر چسب default باعث می‌شود ردیابی برنامه ساده تر شود.

* دستور if زیر را با switch پیاده سازی کنید.

```
if (x ==1)
    cout<<'A';
else if (x ==2)
         cout<<'B';
      else if (x ==3)
              cout<<'C';
           else
              cout<<'D';
```

حل:

```
switch (x)
{
   case 1  : cout<<'A'; break;
   case 2  : cout<<'B'; break;
   case 3  : cout<<'C'; break;
   default : cout<<'D';
 }
```

■

اگر چند case زیر هم ذکر شوند، به طوری که فقط آخری شامل مجموعه دستورات باشد، در این صورت شرط آنها بـا هم or می‌شوند.

* توسط دستورات زیر، کاراکتری را از کاربرگرفته و در صورت اینکه از حروف صدا دار زبان انگلیسی باشد، پیام OK و در غیر اینصورت پیام NO چاپ می‌شود.

```
char  ch;
scantf ("%c",&ch);
switch (ch)
{
  case  'a' :
  case  'e' :
  case  'i'  :
  case  'o :
  case  'u': cout<< "OK"; break;
  default : cout<< "NO";
}
```

* اگر مقدار x برابر ۲ باشد، خروجی چه خواهد بود؟

```
switch (x)
{
 case  1 : cout<<'A';
 case  2 : cout<<'B';
 case  3 : cout<<'C';
}
```

حل: کارکترهای BC چاپ می شود.چون در انتهای دستورات از break استفاده نشده است، دستور پایین تر نیز اجرا می‌شود. یعنی به ازای x=2 ابتدا کاراکتر B چاپ شده و سپس کاراکتر C .

اگر مقابل case بیش از یک عبارت وجود داشته باشد، فقط عبارت سمت راست در نظر گرفته خواهد شد.

* اگر مقدار x برابر ۲ باشد، کاراکتر A و در غیر اینصورت کاراکتر B چاپ خواهد شد:

```
switch  (x)
{
   case  1,2  :  cout<<'A';  break;
   default     :  cout<<'B';  break;
}
```

```
main()
{
  int x;
  cin>>x;
  if(x>=0)
     cout<<"positive";
  else
     cout<<"negative";
}

////////////////////////////


main()
{
  int x,y;
  cin>>x>>y;
  if (x%y==0)
     cout<<"yes";
  else
     cout<<"no";
}

////////////////////////////
main()
{
   int a,b;
   cout<<"enter two number:";
   cin>>a>>b;
   if(a<b)
      cout<<"min="<<a;
   else
      cout<<"min="<<b;
}

////////////////////////////
main()
{
```

```
  int n;
  cin>>n;
  if(n==13)
    cout<<"yes";
  else
    cout<<"no";
}

//////////////////////////

main()
{
  int a,b,c;
  cin>>a>>b>>c;
   if(b<min)
     min=b;
  if (c<min)
     min=c;
  int k;
   k=(a<b)   ? a : b;
   min= (c<k) ? c : k;
  cout<<"min=" <<min;
  if(a<=b && a<=c)
      cout<<a;
  if(b<=a && b<=c)
     cout<<b;
  if(c<=a && c<=b)
     cout<<c;
}

//////////////////////////

main()
{
  int x,y,temp;
  cin>>x>>y;
  if(x>y)
  {
        temp=x;
        x=y;
        y=temp;
}
```

```
    cout<<x<<'\t'<<y;
}

////////////////////////////

main()
{
  int a,b;
  cin>>a>>b;
  if(b!=0  &&  a%b==0)
      cout<<"yes";
   else
      cout<<"no"  ;
}

////////////////////////////

main()
{
  int a,b;
  cin>>a>>b;


  if(b!=0)
    if(a%b==0)
       cout<<"yes";
    else
          cout<<"no";
  else
     cout<<"no";
}

////////////////////////////

main()
{
  int a,b,c;
 cin>>a>>b>>c;
   if(a<b)
    if(a<c)
       cout<<a;
    else
          cout<<c;
```

```
  else
    if(b<c)
       cout<<b;
    else
       cout<<c;
}

//////////////////////////

main()
{
  int grade;
  cin>>grade;


  if(grade>=17 && grade<=20)
        cout<<'A';
  else if (grade>=15  && grade<17)
         cout<<'B';
         else  if (grade>=10 && grade<15)
               cout<<'C';
               else
                  cout<<'D';
}

//////////////////////////

main()
{
  int grade;
  cin>>grade;
    if(grade>=17 && grade<=20)
        cout<<'A';
  else if (grade>=15  && grade<17)
         cout<<'B';
         else  if (grade>=10 && grade<15)
               cout<<'C';
               else
                  cout<<'D';
}

//////////////////////////

main()
```

```
{
 char  ch;
 cin>>ch;
  switch(ch)
{
        case 'r': cout<<"red";
        case 'g': cout<<"green";
        case 'b': cout<<"blue";
        default : cout<<"error";
   }
}
////////////////////////////
main()
{
 char  ch;
 cin>>ch;
 switch(ch)
{
       case 'a':
    case 'e':
    case 'i':
       case 'o':
       case 'u': cout<<"ok"; break;
       default : cout<<"no";
  }
}
////////////////////////////
main()
{
  int a,b;
  char  ch;


  cout<<"enter a,b:";
  cin>>a>>b;
```

```
   cout<<"enter ch:";
  cin>>ch;
  switch(ch)
{
      case '+' : cout<<a+b; break;
      case '-' : cout<<a-b; break;
      case '*':  cout<<a*b; break;
}
}
```

دستورات تکرار

برای تکرار یک دستور، می‌توان از ساختارهای تکرار استفاده کرد. این ساختارها در زبان C عبارتند از:

for , while , do –while

ساختار تکرار for

برای ایجاد حلقه می‌توان از دستور for استفاده کرد. بطور مثال دستورات زیر موجب چاپ سه مرتبه کاراکتر A می‌شود:

```
for (i = 1;i <=3 ; i ++)
     cout<<'A';
```

که می‌توان این عمل را با دستورات زیر نیز پیاده سازی کرد:

```
for (i=3;i >=1;i --)
    cout<<'A';
```

یعنی حلقه for را به دو نوع افزایشی و کاهشی می‌توان پیاده سازی کرد. در واقع ساختار کلی حلقه for به صورت زیر است :

for (مقدار دهی اولیه به متغیر حلقه ; شرط حلقه ; گام حرکت)

دستور ;

که اگر به جای یک دستور، چند دستور در داخل حلقه قرار داشته باشند، مجموعه دستورات را در بین دو آکولاد قرار می‌دهیم.

✱خروجی دستورات چیست؟

```
c = 4;
for (i =0;i <3;i ++)  {
    cout<<i;
    c ++;
 }
cout<< c;
```

حل: خروجی 0127 می باشد. حلقه for مقادیر ۰و۱و۲ را چاپ می‌کند و چون حلقه سه مرتبه چاپ می‌شود، مقدار c ، سه واحد اضافه می‌شود و دستور چاپ خارج از حلقه مقدار ۷ را چاپ می‌کند.

■

✱خروجی حلقه زیر کدام است؟

```
for (i =0; i <=5;i ++);
cout<<i;
```

حل: در آخر حلقه از سمی‌کالون استفاده شده و مقدار شمارنده i به ۶ می رسد و حلقه پایان می یابد. بنابراین در دستور چاپ، مقدار ۶ چاپ خواهد شد.

■

✱خروجی تکه برنامه زیر چیست؟

```
a=1;
```

```
for ( i =2; i !=a+1; i += 2)
    cout<< i ;
cout<< i+a;
```

حل: عدد ۳ چاپ می شود. بعد از ورود به حلقه مقدار i برابر ۲ شده و چون شرط برقرار نیست، حلقه پایان می‌یابد بدون اینکه مقدار شمارنده تغییری کند.

■

*خروجی دستورات زیر چیست؟

```
for (i =1; i<3; i ++)
    cout<<++ i;
```

حل: عدد ۲ چاپ می شود. ابتدا مقدار i برابر ۱ شده و چون شرط برقرار است، در دستور چاپ ابتدا به علت وجود ++i، مقدار i برابر ۲ شده و سپس چاپ می‌شود و بعد از آن توسط حلقه for یک واحد به i اضافه شده و مقدار آن برابر ۳ می شود و چون شرط حلقه برابر نیست، حلقه پایان می‌یابد.

■

*خروجی قطعه برنامه زیر کدام است؟

```
for (i =1,j=3;i<=j; i ++,j --)
    cout<< i* j;
```

حل: ۳۴ چاپ می شود. ابتدا مقدار i برابر ۱ و j برابر ۳ شده و چون شرط برقرار است، مقدار ۳ چاپ می‌شود و سپس یک واحد به i اضافه و یک واحد از j کم می‌شود و چون در این حالت باز هم شرط حلقه برقرار است، مقدار ۴ چاپ می‌شود و با تغییر شمارنده در مرحله بعدی، شرط برقرار نبوده و حلقه پایان می‌یابد.

■

- می‌توان با استفاده از دستور for (; ;) ، یک حلقه بی نهایت ایجاد کرد.
- مقدار شمارنده حلقه در زبان C می‌تواند صحیح، کاراکتری یا حتی اعشاری باشد.
- مقدار نهایی در حلقه را می توان تغییر داد.
- می‌توان هر یک از سه قسمت داخل حلقه for را حذف کرد.
- بعد از پایان حلقه for، مقدار شمارنده بیشتر از مقدار نهایی خواهد بود.

*خروجی دستورات زیر چیست؟

```
for (i =0;i <6; i += 2)
   cout<< i++;
```

حل: 03 چاپ می شود. مقدار شمارنده حلقه ابتدا صفر است و چون شرط برقرار است، مقدار صفر چاپ شده و در هنگام خروج از دستور cout<< مقدار شمارنده یک واحد افزایش می‌یابد و توسط حلقه نیز دو واحد افزایش می‌یابد و مقدار آن ۳ می‌شود و چون شرط حلقه برقرار است، مقدار ۳ چاپ شده و مجدداً بعد از خروج از دستور چاپ شمارنده ۴ شده و توسط حلقه نیز مقدار آن ۶ می‌شود و شرط حلقه دیگر برقرار نیست.

■

*خروجی دستورات زیر چیست؟

```
for (i =2 ;i <7; i ++)
   if (i%2 == 0)
      cout<<i;
```

حل: توسط دستور if مقادیر i هایی که بر ۲ قابل قسمت هستند، چاپ می‌شود، یعنی خروجی عبارت است از:

246

■

*عملکرد برنامه زیر چیست؟

```
main( ){
   char   ch;
   int     i ;
   for (i=0 ; (ch=getchar ( ) )! ='.' ; i++);
   cout<<i;
}
```

حل: شمارش تعداد حروف یک جمله که انتهای جمله با نقطه مشخص شده است.

■

ساختار تکرار while

از ساختار while در مواقعی استفاده می‌شود که تعداد دفعات تکرار مجموعه دستورات مورد نظر مشخص نباشد.

ساختار while به صورت زیر است:

```
while(شرط) {
   مجموعه دستورات
}
```

* دستور زیر اعداد ۱ تا ۳ را چاپ می‌کند.

```
i =1;
while(i<=3)
  {
     cout<< i;
     i ++;
  }
```

✱ عملکرد دستورات زیر چیست؟

```
long  int  f=1;
while(n>1)
    f *=n--;
```

حل: محاسبه فاکتوریل عدد n .

✱ عملکرد دستورات زیر چیست؟

```
int  c=0;
while(getchar( ) != '\r')
      c++;
```

حل: شمارش تعداد کاراکترهای یک جمله (انتهای جمله به کلید Enter ختم می‌شود)

✱ خروجی چیست؟

```
int   i=3;
while(i){
   cout<< i;
   i--;
 }
```

حل: حلقه تا صفر شدن i تکرار می‌شود و 321 چاپ خواهد شد.

✱حلقه زیر تا چه موقع تکرار می شود؟

```
while((ch=getcher( )) !='A' );
```

حل: تا وارد کردن کاراکتر A .

✱ اگر از ورودی ali ^z وارد شود، چه چیزی در صفحه نمایش نشان داده می شود؟

```
void  main( ){
  char   c;
  while((c=cin.get( ) ) !=EOF)
       cout<<c;
}
```

حل: با وارد کردن ali و سپس ctrl+z ، رشته ali نمایش داده خواهد شد. در واقع رشته aliali مشاهده خواهد شد.(یک رشته ali که کاربر وارد کرده و یک رشته ali که توسط cout نمایش داده شده است.)

✱ اگر از ورودی ali ^z وارد شود، چه چیزی در صفحه نمایش نشان داده می شود؟

```
void main( )
{
   char c;
  cout<<cin.eof( );
  while((c=cin.get( ) ) !=EOF);
  cout<<cin.eof( );
}
```

حل: با اجرای برنامه عدد صفر چاپ شده و با ورود رشته ali و سپس وارد کردن ctrl+z ، عدد ۱ نیز نمـایش داده مـی شـود. بنابراین در خروجی 0ali1 نمایش داده می شود.

■

ساختار تکرار do –while

حلقه do – while مشابه حلقه while است. با این تفاوت که شرط حلقه در انتهای حلقه تست می‌شود. بنـابراین مجموعـه دستورات داخل ساختار do –while، حداقل یکبار انجام می‌شوند. ساختار این دستور عبارت است از :

```
do {
                مجموعه دستورات
}while (شرط)
```

* حلقه زیر اعداد 5 تا 7 را چاپ می‌کند:

```
i =5;
do{
      cout<< i;
      i++;
   }while (i <=7);
```

*خروجی برنامه زیر به ازای ورود عدد 123 چیست؟

```
main( ){
  int   n,d ;
  scanf ("%d",&n);
  do{
       d = n%10;
       cout<< d;
       n/= 10;
    }while(n!=0);
}
```

حل: برنامه عددی را از ورودی خوانده و معکوس آن را چاپ می‌کند. بنابراین خروجی 321 است.

* در چه صورت حلقه زیر پایان می یابد؟

```
do{
   cin>>ch;
}while(ch <'A' ||  ch >'D');
```

حل: در صورت وارد کردن کاراکتری بین حروف A تا D ، حلقه پایان می‌یابد.

دستورات انتقال کنترل غیر شرطی

دستورات انتقال کنترل غیر شرطی عبارتند از : break ، continue، goto , exit.

دستور break

در صورت استفاده از این دستور در داخل حلقه ها ، کنترل اجرای برنامه به اولین دستور پس از حلقه تکرار منتقل می‌شوند.

* خروجی دستورات زیر چیست؟

```
for(i=1; i <8; i ++)
```

```
{
   if (i==4)  break;
   cout<< i;
}
```

حل: اعداد ۱ تا ۳ را چاپ می‌کند (مقدار شمارنده وقتی به ۴ می رسد، حلقه پایان می‌یابد.)

■

دستور break موجود در داخلی ترین حلقه، فقط موجب خروج از همین حلقه می‌شود.

دستور continue

در صورت استفاده از دستور continue در داخل حلقه ها ، کنترل به ابتدای حلقه می‌رود و بعد از تغییر مقدار شمارنده شرط حلقه تست می‌شود.

✱ خروجی دستورات زیر چیست؟

```
for (i =1; i <=5;i ++)
{
 if (i ==2)  continue;
 cout<<i;
}
```

حل: اعداد ۱ تا ۵ به غیر از عدد ۲ چاپ می‌شود.به عبارتی خروجی 1345 می باشد.

■

✱ در صورتی که از ورودی اعداد ۱۲ و ۳- و۵و ۱۰۰و ۸۰ به ترتیب وارد شوند، خروجی چه خواهد بود؟

```
do {
    cin>>x;
    if (x< 0)
        countinue;
    cout<< x ;
 } while (x!=100);
```

حل: ابتدا ۱۲ و سپس ۵ و در نهایت ۱۰۰ چاپ می شود.

دستورات داده شده، اعداد مثبت خوانده شده از ورودی را نمایش می دهند و در صورت وارد کردن عدد ۱۰۰ حلقه پایان می‌یابد. عدد ۸۰ دیگر خوانده نمی شود.

■

دستور goto

توسط دستور goto می‌توان کنترل را به مکانی برد که توسط برچسب مشخص شده است. البته از این دستور به ندرت استفاده می‌شود.

* خروجی دستورات زیر چاپ اعداد ۱ تا ۳ می‌باشد :

```
x =1;
loop1:
     cout<< x;
     x ++;
     if (x <= 3)
         goto  loop1;
```

از کاربردهای خوب goto خروج از حلقه تودر تو را می‌توان نام برد.

```
for (…){
   for (…){
      for (…){
        if (…) goto stop;
             ⋮
      }
  }
}
stop : cout<< "ERROR";
```

تابع exit( )

توسط تابع exit ( ) می‌توان به طور کامل از برنامه خارج شد. استفاده از عدد صفر در داخل پرانتز به صورت exit (0) باعث خروج عادی و استفاده از عدد یک به معنای خروج بر اثر بروز خطا می‌شود.

الگوی تابع exit( ) در فایل stdlib.h می‌باشد.

حلقه های متداخل (تو در تو)

حلقه های تودر تو از یک حلقه خارجی با یک یا چند حلقه داخلی تشکیل می‌شوند. هربار که حلقه خـارجی تکـرار می‌شـود، حلقه های داخلی از نو اجرا می‌شوند و عبارت کنترل حلقه آنها از نو ارزیابی شده و تمام تکرارهای مورد نیاز انجام می‌شوند.

✱ خروجی حلقه های تودرتو زیر چیست؟

```
for (i=1; i <= 3; i ++ )
   for (j=4;j <=5; j ++ )
         cout<<j;
```

حل: خروجی 454545 می باشد. حلقه خارجی ۳ مرتبه و حلقه داخلی ۲ مرتبه تکرار می شود.

■

✱ خروجی حلقه های تودرتو زیر چیست؟

```
for (i=1;i<=3;i ++)
   for (j=4;j <=5;j ++)
        cout<<i;
```

حل: خروجی 112233 می باشد.

■

✱ چند مرتبه رشته farshid چاپ خواهد شد؟

```
for (i =6 ; i >2 ; i --)
    for (j =0; j <=2 ; j ++)
       for (k=-3 ; k!=0 ; k ++)
         cout<< "farshid";
```

حل: ۳۶ مرتبه. تعداد تکرار حلقه اول برابر (۶-۲)، حلقه دوم برابر (۱+ ۲-۰) و حلقه سوم برابر ۳ می باشـد. بنـابراین دسـتور چاپ در کل ۳× ۳×۴ مرتبه اجرا می شود.

■

✱خروجی دستورات زیر چیست؟

```
for (i=1; i<=3 ; i++)
{
 for (j=1; j <= i ; j++)
       cout<<j;
  cout<< "\n";
}
```

حل:

```
1
1     2
1     2     3
```

بعد از پایان هر بار تکرار حلقه داخلی، دستور cout<< "\n"; موجب انتقال مکان نما به خط بعدی می‌شود.

منتخبی از عناوین آموزشی منتشر شده بر روی فرادرس

برنامه‌نویسی	
عنوان آموزش	مدت زمان تقریبی
مبانی برنامه نویسی - کلیک کنید (+)	۳ ساعت
برنامه نویسی C - کلیک کنید (+)	۱۳ ساعت
آموزش برنامه نویسیC++	۲۰ ساعت
برنامه نویسی کاربردی سی شارپ - کلیک کنید (+)	۱۴ ساعت
آموزش جامع شی گرایی در سی شارپ - کلیک کنید (+)	۱۴ ساعت
برنامه نویسی جاوا - کلیک کنید (+)	۲۳ ساعت
آموزش برنامه نویسی - PHP کلیک کنید (+)	۲۸ ساعت
آموزش فریمورک PHP کدایگنایتر - (CodeIgniter) کلیک کنید (+)	۷ ساعت
آموزش اسکریپت برنامه نویسی - jQuery کلیک کنید (+)	۷ ساعت
آموزش ویژوال بیسیک دات نت - (Visual Basic.NET) کلیک کنید (+)	۱۳ ساعت
آموزش تکمیلی ویژوال بیسیک دات نت - (Visual Basic.NET) کلیک کنید (+)	۱۶ ساعت
آموزش برنامه نویسی با روش سه لایه به زبان - VB.Net کلیک کنید (+)	۴ ساعت
برنامه نویسی اسمال بیسیک یا - Small Basic کلیک کنید (+)	۱۶ ساعت
آموزش ساخت بازی ساده در ویژوال بیسیک - کلیک کنید (+)	۲ ساعت
آموزش کاربردی - SQL Server کلیک کنید (+)	۱۱ ساعت
آموزش آشنایی با LINQ to SQL در - C# کلیک کنید (+)	۲ ساعت
آموزش برنامه نویسی با روش سه لایه به زبان سی شارپ - کلیک کنید (+)	۴ ساعت
آموزش برنامه نویسی تحت شبکه با سی شارپ در قالب پروژه - کلیک کنید (+)	۱ ساعت
آموزش Cryptography در دات نت - کلیک کنید (+)	۳ ساعت

برنامه‌نویسی (ادامه از صفحه قبل)	
عنوان آموزش	مدت زمان تقریبی
آموزش قفل نرم افزاری در سی شارپ از طریق رجیستری	۴ ساعت
آموزش ساخت اپلیکیشن کتاب و کار با داده‌ها در اندروید - کلیک کنید (+)	۱۳ ساعت
آموزش ارتباط با دیتابیس سمت سرور در اندروید - کلیک کنید (+)	۱۴ ساعت
آموزش ساخت روبات و کنترل آن با اندروید - کلیک کنید (+)	۱۶ ساعت
آموزش ساخت اپلیکیشن دیکشنری صوتی دو زبانه با قابلیت تشخیص صوت کاربر - کلیک کنید (+)	۷ ساعت
آموزش مدیریت بانک اطلاعاتی اوراکل - کلیک کنید (+)	۹ ساعت
آموزش مدیریت بانک اطلاعاتی اوراکل پیشرفته - کلیک کنید (+)	۷ ساعت
آموزش راه اندازی اوراکل ۱۲ cدر لینوکس	۱ ساعت
آموزش دیتاگارد در اوراکل - کلیک کنید (+)	۳ ساعت
برنامه نویسی متلب - کلیک کنید (+)	۹ ساعت
متلب برای علوم و مهندسی - کلیک کنید (+)	۱۴ ساعت
برنامه نویسی متلب پیشرفته - کلیک کنید (+)	۷ ساعت
طراحی رابط های گرافیکی (GUI) در متلب - کلیک کنید (+)	۸ ساعت
آموزش برنامه نویسی R و نرم‌افزار - RStudioکلیک کنید (+)	۷ ساعت
آموزش تکمیلی برنامه نویسی R و نرم‌افزار - RStudio کلیک کنید(+)	۵ ساعت
آموزش برنامه نویسی پایتون ۱ - کلیک کنید (+)	۲۰ ساعت
آموزش برنامه نویسی پایتون ۲ - کلیک کنید (+)	۵ ساعت
آموزش گرافیک کامپیوتری با - OpenGL کلیک کنید(+)	۱۶ ساعت

راه اندازی و مدیریت وبسایت‌ها و سرورها	
عنوان فرادرس	**مدت زمان تقریبی**
آموزش برنامه نویسی - PHP کلیک کنید (+)	۲۸ ساعت
آموزش فریمورک PHP کدایگنایتر - (CodeIgniter) کلیک کنید (+)	۷ ساعت
آموزش طراحی وب با - HTML کلیک کنید (+)	۳ ساعت
آموزش طراحی وب با - CSS کلیک کنید (+)	۵ ساعت
آموزش پروژه محور HTML و - CSS کلیک کنید - کلیک کنید (+)	۴ ساعت
آموزش جاوا اسکریپت - (JavaScript) کلیک کنید (+)	۹ ساعت
آموزش کار با - cPanel کلیک کنید (+)	۱ ساعت
آموزش مدیریت هاست با - DirectAdmin کلیک کنید - کلیک کنید (+)	۱ ساعت
راه اندازی سایت و کار با وردپرس - کلیک کنید (+)	۷ ساعت
راه اندازی فروشگاه دیجیتال با وردپرس و - Easy Digital Downloads کلیک کنید (+)	۱ ساعت
آموزش راه اندازی سایت شخصی با وردپرس - کلیک کنید (+)	۱ ساعت
آموزش ترجمه قالب وردپرس - کلیک کنید (+)	۲ ساعت
آموزش راه اندازی سایت خبری با وردپرس - کلیک کنید (+)	۲ ساعت

علوم کامپیوتر	
عنوان آموزش	مدت زمان تقریبی
ساختمان داده‌ها -کلیک کنید (+)	۱۰ ساعت
آموزش ساختمان داده‌ها (مرور – تست کنکور ارشد) - کلیک کنید (+)	۲۰ ساعت
آموزش نظریه زبان‌ها و ماشین‌ها -کلیک کنید (+)	۹ ساعت
آموزش نظریه زبان‌ها و ماشین (مرور – تست کنکور ارشد) - کلیک کنید (+)	۸ ساعت
آموزش سیستم های عامل - کلیک کنید (+)	۱۱ ساعت
آموزش سیستم عامل (مرور اجمالی و تست کنکور) - کلیک کنید (+)	۱۲ ساعت
آموزش پایگاه داده‌ها -کلیک کنید (+)	۸ ساعت
آموزش پایگاه داده‌ها (مرور – تست کنکور ارشد) - کلیک کنید (+)	۵ ساعت
آموزش طراحی و پیاده سازی زبان های برنامه سازی - کلیک کنید (+)	۱۰ ساعت
آموزش طراحی و پیاده سازی زبان های برنامه سازی (مرور – تست کنکور ارشد) - کلیک کنید (+)	۱۲ ساعت
آموزش روش های حل روابط بازگشتی - کلیک کنید (+)	۴ ساعت
آموزش روش تقسیم و حل در طراحی الگوریتم - کلیک کنید (+)	۲ ساعت
آموزش ذخیره و بازیابی اطلاعات - کلیک کنید (+)	۸ ساعت
آموزش ساختمان گسسته با رویکرد حل مساله - کلیک کنید (+)	۱۶ ساعت
آموزش جامع مدارهای منطقی - کلیک کنید (+)	۱۰ ساعت
آموزش معماری کامپیوتر با رویکرد حل مسأله - کلیک کنید (+)	۲۰ ساعت
آموزش ساختمان گسسته (مرور و حل تست های کنکور کارشناسی ارشد) - کلیک کنید (+)	۱۲ ساعت
آموزش طراحی الگوریتم - کلیک کنید (+)	۸ ساعت
آموزش شبکه های کامپیوتری ۱ - کلیک کنید (+)	۱۹ ساعت

علوم کامپیوتر (ادامه از صفحه قبل)	
عنوان آموزش	مدت زمان تقریبی
آموزش نظریه گراف و کاربردها - کلیک کنید (+)	۱۴ ساعت
آموزش نتورک پلاس - (Network+) کلیک کنید (+)	۱۰ ساعت
آموزش مدل سازی UML با نرم‌افزار - Rational Roseکلیک کنید (+)	۳ ساعت
آموزش پردازش ویدئو - کلیک کنید (+)	۳ ساعت
پردازش تصویر در متلب - کلیک کنید (+)	۱۶ ساعت
آموزش پردازش تصویر با - OpenCV کلیک کنید (+)	۱۰ ساعت

هوش مصنوعی	
عنوان آموزش	مدت زمان تقریبی
الگوریتم ژنتیک در متلب - کلیک کنید (+)	۱۴ ساعت
الگوریتم PSO در متلب - کلیک کنید (+)	۱۰ ساعت
الگوریتم ازدحام ذرات (PSO) گسسته باینری - کلیک کنید (+)	۲ ساعت
ترکیب الگوریتم ژنتیک و PSO در متلب - کلیک کنید (+)	۱ ساعت
حل مسأله فروشنده دوره گرد با استفاده از الگوریتم ژنتیک - کلیک کنید (+)	۲ ساعت
الگوریتم مورچگان در متلب - کلیک کنید (+)	۶ ساعت
الگوریتم رقابت استعماری در متلب - کلیک کنید (+)	۱۳ ساعت
طراحی سیستم های فازی عصبی یا ANFIS با استفاده از الگوریتم های فرا ابتکاری و تکاملی - کلیک کنید (+)	۲ ساعت
الگوریتم فرهنگی یا Cultural Algorithm در متلب - کلیک کنید (+)	۲ ساعت

هوش مصنوعی (ادامه از صفحه قبل)	
عنوان آموزش	مدت زمان تقریبی
شبیه سازی تبرید یا Simulated Annealing در متلب - کلیک کنید (+)	۴ ساعت
جستجوی ممنوع یا Tabu Search در متلب - کلیک کنید (+)	۲ ساعت
الگوریتم کرم شب تاب یا Firefly Algorithm در متلب - کلیک کنید (+)	۱ ساعت
بهینه سازی مبتنی بر جغرافیای زیستی یا BBO در متلب - کلیک کنید (+)	۲ ساعت
جستجوی هارمونی یا Harmony Search در متلب - کلیک کنید (+)	۲ ساعت
کلونی زنبور مصنوعی یا Artificial Bee Colony در متلب - کلیک کنید (+)	۳ ساعت
الگوریتم زنبورها یا Bees Algorithm در متلب - کلیک کنید (+)	۲ ساعت
الگوریتم تکامل تفاضلی - کلیک کنید (+)	۱ ساعت
الگوریتم بهینه سازی علف هرز مهاجم یا IWO در متلب - کلیک کنید (+)	۲ ساعت
الگوریتم بهینه سازی مبتنی بر و یادگیری یا - TLBO کلیک کنید (+)	۱ ساعت
الگوریتم بهینه سازی جهش قورباغه یا SFLA در متلب - کلیک کنید (+)	۴ ساعت
بهینه سازی چند هدفه در متلب - کلیک کنید (+)	۱۹ ساعت
بهینه سازی مقید در متلب - کلیک کنید (+)	۹ ساعت
شبکه های عصبی مصنوعی در متلب - کلیک کنید (+)	۲۸ ساعت
آموزش کاربردی شبکه های عصبی مصنوعی - کلیک کنید (+)	۹ ساعت
آموزش استفاده از شبکه عصبی مصنوعی با نروسولوشن - کلیک کنید (+)	۳ ساعت
شبکه عصبی GMDH در متلب - کلیک کنید (+)	۴ ساعت
شبکه های عصبی گازی به همراه پیاده سازی عملی در متلب - کلیک کنید (+)	۳ ساعت
طبقه بندی و بازشناسی الگو با شبکه های عصبی LVQ در متلب - کلیک کنید (+)	۳ ساعت
آموزش پیاده سازی الگوریتم های تکاملی و فراابتکاری در سی شارپ - کلیک کنید (+)	۸ ساعت

آمار و داده کاوی	
عنوان آموزش	مدت زمان تقریبی
گنجینه فرادرس های یادگیری ماشین و داده کاوی - کلیک کنید (+)	۸۸ ساعت
گنجینه فرادرس های محاسبات هوشمند - کلیک کنید (+)	۷۱ ساعت
آموزش یادگیری ماشین - کلیک کنید (+)	۲۴ ساعت
داده کاوی یا Data Mining در متلب - کلیک کنید (+)	۲۴ ساعت
آموزش داده کاوی در - RapidMiner کلیک کنید (+)	۲ ساعت
آموزش وب کاوی - کلیک کنید (+)	۱۷ ساعت
شبکه های عصبی مصنوعی در متلب - کلیک کنید (+)	۲۸ ساعت
آموزش کاربردی شبکه های عصبی مصنوعی - کلیک کنید (+)	۹ ساعت
شبکه عصبی GMDH در متلب - کلیک کنید (+)	۴ ساعت
شبکه های عصبی گازی به همراه پیاده سازی عملی در متلب - کلیک کنید (+)	۳ ساعت
طبقه بندی و بازشناسی الگو با شبکه های عصبی LVQ در متلب - کلیک کنید (+)	۳ ساعت
خوشه بندی با استفاده از الگوریتم های تکاملی و فراابتکاری - کلیک کنید (+)	۳ ساعت
تخمین خطای کلاسیفایر یا - Classifier کلیک کنید (+)	۲ ساعت
انتخاب ویژگی یا - Feature Selection کلیک کنید (+)	۲ ساعت
انتخاب ویژگی با استفاده از الگوریتم های فرا ابتکاری و تکاملی - کلیک کنید (+)	۴ ساعت
کاهش تعداد رنگ تصاویر با استفاده از روش های خوشه بندی هوشمند - کلیک کنید (+)	۱ ساعت
آموزش پردازش سیگنال های واقعی در متلب - کلیک کنید (+)	۴ ساعت
مبانی و کاربردهای راهبرد تلفیق داده یا - Data Fusion کلیک کنید (+)	۹ ساعت
آمار و احتمال مهندسی - کلیک کنید (+)	۱۳ ساعت

آزمون‌های فرض مربوط به میانگین جامعه نرمال در - SPSS کلیک کنید (+)	۳ ساعت
آمار و داده کاوی (ادامه از صفحه قبل)	
عنوان آموزش	**مدت زمان تقریبی**
آموزش محاسبات آماری در اکسل - کلیک کنید (+)	۲ ساعت
آموزش کنترل کیفیت آماری - کلیک کنید (+)	۵ ساعت
آموزش کنترل کیفیت آماری با - SPSS کلیک کنید (+)	۲ ساعت
آموزش مدل سازی معادلات ساختاری با - Amos کلیک کنید (+)	۷ ساعت
تجزیه و تحلیل اطلاعات با نرم‌افزار - SASکلیک کنید (+)	۴ ساعت

مهندسی برق	
عنوان آموزش	**مدت زمان تقریبی**
طراحی دیجیتال با استفاده از وریلوگ یا - Verilog کلیک کنید (+)	۷ ساعت
آموزش جامع مدارهای منطقی - کلیک کنید (+)	۱۰ ساعت
آموزش مروری طراحی و پیاده سازی مدارات منطقی - کلیک کنید (+)	۴ ساعت
آموزش میکروکنترلر AVR و نرم‌افزار - CodevisionAVRکلیک کنید (+)	۴ ساعت
آموزش تکمیلی میکروکنترلر AVR و نرم‌افزار - CodevisionAVRکلیک کنید (+)	۴ ساعت
آشنایی باPLC های ساخت شرکت های Omron و - Keyence کلیک کنید (+)	۶ ساعت
میکروکنترلر PIC با کامپایلر - CCS کلیک کنید (+)	۹ ساعت
آموزش تحلیل و طراحی مدارات الکترونیکی با - Proteus کلیک کنید (+)	۳ ساعت
آموزش شبیه سازی و تحلیل مدارهای الکتریکی و الکترونیکی با پی اسپایس (PSpice) -کلیک کنید (+)	۳ ساعت

آموزش مقدماتی - ADS کلیک کنید (+)	۳ ساعت
آموزش تکمیلی آنالیز مدار با نرم‌افزار - ADSکلیک کنید (+)	۲ ساعت
مهندسی برق (ادامه از صفحه قبل)	
عنوان آموزش	**مدت زمان تقریبی**
آموزش تحلیل ریاضی مدارات الکتریکی با - OrCAD کلیک کنید (+)	۲ ساعت
آموزش شبیه سازی مدارات الکترونیکی با - Orcad Capture کلیک کنید (+)	۲ ساعت
آموزش برنامه نویسی آردوینو (- (Arduinoکلیک کنید (+)	۸ ساعت
آموزش تکمیلی برنامه نویسی آردوینو - (Arduino) کلیک کنید (+)	۷ ساعت
آموزش طراحی برد مدار چاپی به کمک نرم‌افزار - Altium Designer کلیک کنید (+)	۷ ساعت
آموزش مبانی ربات های برنامه پذیر - کلیک کنید (+)	۵ ساعت
آموزش ساخت روبات و کنترل آن با اندروید - کلیک کنید (+)	۱۶ ساعت
آموزش مدارهای الکتریکی ۱ - کلیک کنید (+)	۹ ساعت
آموزش مدارهای الکتریکی ۲ - کلیک کنید (+)	۱۱ ساعت
آموزش سیستم های کنترل خطی - کلیک کنید (+)	۱۰ ساعت
آموزش مکاترونیک کاربردی ۱ - کلیک کنید (+)	۱۳ ساعت
آموزش کامسول (مباحث منتخب) - کلیک کنید (+)	۳ ساعت
آموزش سینماتیک مستقیم و معکوس ربات‌ها -کلیک کنید (+)	۳ ساعت
آموزش تجزیه و تحلیل سیگنال‌ها و سیستم ها - کلیک کنید (+)	۲۷ ساعت
آموزش متلب با نگرش تحلیل آماری، تحلیل سری های زمانی و داده های مکانی - کلیک کنید (+)	۸ ساعت
پردازش سیگنال های دیجیتال با استفاده از نرم‌افزار متلب - کلیک کنید (+)	۴ ساعت
شبیه سازی سیستم با سیمیولینک - کلیک کنید (+)	۴ ساعت
آموزش سیستم های قدرت در سیمولینک و متلب - کلیک کنید (+)	۱۱ ساعت
آنالیز پایداری و کنترل سیستم های قدرت با استفاده از جعبه ابزارهای نرم‌افزار متلب -	۲ ساعت

کلیک کنید (+)	
آشنایی با SimPowerSystems در شبیه سازی سیستم های قدرت - کلیک کنید (+)	۳ ساعت
مهندسی برق (ادامه از صفحه قبل)	
عنوان آموزش	**مدت زمان تقریبی**
شبیه سازی ماشین های الکتریکی در تولباکس های Simulink و SimPowerSystem در نرم‌افزار متلب - کلیک کنید (+)	۴ ساعت
آموزش الکترونیک قدرت – شبیه سازی در متلب و سیمولینک - کلیک کنید (+)	۸ ساعت
آموزش شبیه سازی عملکرد انواع ماشین های الکتریکی در سیمولینک متلب - کلیک کنید (+)	۱۰ ساعت
برنامه های پاسخگویی بار - کلیک کنید (+)	۴ ساعت
آموزش نرم‌افزار ETAP برای تحلیل سیستم های قدرت - کلیک کنید (+)	۲۱ ساعت
آموزش مقدماتی نرم‌افزار GAMSبرای حل مسائل بازار برق - کلیک کنید (+)	۵ ساعت
آموزش پخش بار اقتصادی (دیسپاچینگ اقتصادی) در GAMS - کلیک کنید (+)	۲ ساعت
کاربرد فازی در سیستم های قدرت - کلیک کنید (+)	۲ ساعت
آموزش نرم‌افزار HFSS - کلیک کنید (+)	۵ ساعت
طراحی آنتن مایکرواستریپ به کمک نرم‌افزار HFSS - کلیک کنید (+)	۱ ساعت
آموزش طراحی و شبیه سازی آنتن های SIW با HFSS - کلیک کنید (+)	۱ ساعت
آموزش بررسی کامل آنتن های مایکرواستریپ و طراحی آن توسط CST - کلیک کنید (+)	۳ ساعت
آموزش تجزیه سیگنال به مولفه های مود ذاتی یا Empirical Mode Decomposition -کلیک کنید (+)	۴۰ دقیقه
نمونه برداری و بازسازی اطلاعات در سیستم های کنترل دیجیتال - کلیک کنید (+)	۳ ساعت
بررسی پاسخ ورودی پله در شناسایی فرآیندهای صنعتی - کلیک کنید (+)	۱ ساعت
مدل سازی و شناسایی سیستم های دینامیکی با استفاده از مدل ARX و شبکه فازی عصبی - ANFIS کلیک کنید (+)	۱ ساعت

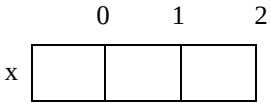
طراحی و تنظیم ضرایب کنترل کننده PID با منطق فازی - کلیک کنید (+)	۲ ساعت
آموزش کنترل سیستم چهار تانک - کلیک کنید (+)	۲ ساعت

فصل ۳ :

آرایه – رشته

آرایه

آرایه، محل نگهداری عناصر هم نوع تحت یک نام است. توسط اندیس آرایه می توان به هرکدام از عناصر دسترسی داشت. به طور نمونه تعریف ; int x[3] ، یک آرایه با ۳ خانه با اندیسه های ۰ تا ۲را تعریف می کند که در هر خانه آرایه می توان یک عدد صحیح قرار داد:



برای دسترسی به خانه دوم این آرایه از عبارت x[1] استفاده می شود. مثلاً با دستور زیر عدد ۱۰ در خانه دوم آرایه قرار خواهد گرفت:

```
x[1]=10;
```

آرایه x دارای ۳ خانه ۲ بایتی است و ۶ بایت از حافظه را اشغال می کند.

در زبان C ، bound checking وجود ندارد. یعنی در صورت استفاده از دستوری مانند ;x[5]=10 در مثال بالا توسط کامپایلر خطایی گرفته نمی شود. ولی بدیهی است که نتیجه مطلوبی نخواهد داشت.

✱ آرایه زیر چند بایت فضا می خواهد؟

```
float  x[5];
```

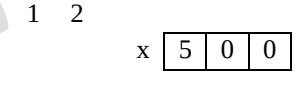
حل : آرایه دارای ۵ خانه است که در هر خانه آن می توان یک داده از نوع float که ۴ بایتی است را قرار داد. یعنی به ۲۰ = ۴ × ۵ بایت فضا نیاز دارد.

■

✱ عملکرد دستور زیر چیست؟

```
int  x[3] ={5};
```

حل : مقدار خانه اول برابر ۵ شده و بقیه خانه ها صفر هستند.



■

✱ اجرای دستور زیر موجب بروز پیام خطا خواهد شد. چون بیشتر از تعداد خانه های آرایه ، مقداردهی شده است.

```
int  a[2] ={5, 3, 6};
```

■

* عملکرد دستور زیر چیست؟

```
float   x[4]={2.6, 'a',5};
```

حل : اجرای دستور موجب مقدار دهی آرایه به صورت زیر می شود:

	0	1	2	3
X	2.6	97.0	5.0	0.0

■

* آرایه x دارای چند خانه است؟

```
int  x[ ]={3,1,6};
```

حل: آرایه دارای ۳ خانه است.

	0	1	2
X	3	1	6

تذکر: می توان بعد آرایه را مشخص نکرد، که در این حالت عناصر آرایه به تعداد مقدار دهی انجام شده می باشد.

■

* خروجی دستورات زیر چیست؟

```
double   x[6]={16.0,12.0, 6.0, 8.0, 2.5, 12.0};
cout<<x[(int) x[4] ] ;
```

حل : خروجی 6.0 است. چون مقدار x[4] برابر ۲/۵ است که int آن برابر ۲ است و مقدار x[2] برابر ۶/۰ است.

■

* توسط حلقه زیر عناصر آرایه با مربع اعداد صحیح ۰ تا ۳ پر می شود:

```
int   square[4],i;
for (i=0;i<4 ; i ++)
   square[i]=i*i ;
```

به صورت زیر :

0	1	2	3
0	1	4	9

■

✱ بعد از اجرای دستورات زیر، محتویات آرایه چه تغییری خواهد کرد؟

```
int   x[4]={1,2,3,4};
int   i,j;
for (i=0,j=3;i<2;i++,j--)
        x[i]=x[j];
```

حل: عناصر آرایه متقارن می شود :

i	j	عملکرد
0	3	4,2,3,4
1	2	4,3,3,4

تذکر: البته اگر شرط i<2 به صورت i<4 بود، همین نتیجه حاصل می شد. در واقع کارهای بعدی بی تاثیر بود.

■

✱ بعد از اجرای دستورات زیر، محتویات آرایه چه تغییری خواهد کرد؟

```
int   x[4]={1,2,3,4};
int   i,j,temp;
for (i=0,j=3; i <2; i++,j--)
  {
       temp=x[i];
       x[i]=x[j];
       x[j]=temp;
  }
```

حل: توسط دستورات داخل حلقه، محتویات x[i]وx[j] تعویض می شود. بنابراین با اجرای حلقه عناصر آرایه وارونه می شود:

i	j	عملکرد
0	3	4,2,3,1
1	2	4,3,2,1

تذکر: اگر شرط حلقه به جای i<2 ، برابر i<4 باشد، آنگاه محتویات آرایه هیچ تغییری نمی کند.

■

✱ بعد از اجرای دستورات زیر، محتویات آرایه چه تغییری خواهد کرد؟

```
int   x[4]={1,2,3,4};
int   i,j,temp;
for (i=0,j=3; i <4; i++,j--)
  {
       temp=x[i];
       x[i]=x[j];
       x[j]=temp;
  }
```

حل: محتویات آرایه هیچ تغییری نمی کند.

■

آرایه دو بعدی

آرایه دو بعدی یا ماتریس را می توان به صورت زیر تعریف کرد:

```
int   x[3][2] ;
```

آرایه تعریف شده دارای ۶ خانه ۲ بایتی است. بنابراین ۱۲ بایت از فضا را اشغال می کند.

✱تعریف زیر چند بایت از حافظه را اشغال می کند؟

```
int  a[3][2]={{50,12},{13,36},{15,20}};
```

حل: در واقع آرایه دارای ۶ خانه ۲ بایتی است:

	0	1
0	50	12
1	13	36
2	15	20

تذکر: می توان آرایه a را به صورت زیر هم مقدار دهی کرد:

```
int   a[3][2] ={50,12,13,36,15,20};
```

در آرایه های دوبعدی ، بعد اول را می توان مشخص نکرد که با توجه به مقداردهی اولیه مشخص خواهد شد.

✱ بعد اول آرایه زیر را مشخص نمایید؟

```
int   x[ ][3]={1,2,3,4,5,6,7};
```

حل: بعد اول آرایه از رابطه $\left\lceil \frac{۷}{۳} \right\rceil$ قابل محاسبه است.(صورت، تعداد مقداردهی های انجام شده و مخرج ، بعد دوم می باشد.)

✱ با توجه به تعریف زیر، در هر خانه چه عددی قرار می گیرد؟

```
int   a[3][3]={{1,2,3},{(4,5),6,7},8,9};
```

حل: چون 4,5 با کاما جدا شده اند و در بین پرانتز قرار دارند، فقط عدد ۵ در نظر گرفته می شود و تعریف به صورت زیر می باشد:

```
int   a[3][3]={{1,2,3},{5,6,7},8,9};
```

	0	1	2
0	1	2	3
1	5	6	7
2	8	9	0

✱ خروجی کدام است؟

```
int  a[2][3]={{1,2},{3,(4,5)}};
for(int  i=0;i<2; i++)
    for(int  j=0;j<3; j++)
        cout<<a[i,j];
```

}

حل: برنامه داده شده ، ۶ آدرس را چاپ می کند. چون دستور a[i,j] به معنی a[j] می باشد(طبق خاصیت کاما). البته اگر دستور به صورت a[i][j] بود، خروجی 120350 بود. ■

آرایه های چند بعدی

می توان آرایه با ابعاد بیشتر از ۲ بعد، تعریف کرد. آرایه زیر ۳ بعدی می‌باشد:

```
int  x[2][4][3];
```

این آرایه دارای ۲۴ خانه دوبایتی است.

✱ مقداردهی اولیه آرایه a :

```
int   a[3][4][2] = {{{1, 2},{3,4},{5, 6}{7,8}} ,
                    {{9,10},{11,12},{13, 14},{15,16}},
                     {17,18},{19,20},{21, 22},{23,24}},     };
```

■

✱ آرایه زیر دارای چند عضو است؟

```
int   a[ ][2][3]={1, 2, 3, 4, 5, 6, 7, 8};
```

حل : تعداد عناصر آرایه را بر ۶ تقسیم کرده تا بعد اول آرایه مشخص شود:

$$\left\lceil \frac{۸}{۲\times۳} \right\rceil = \left\lceil \frac{۸}{۶} \right\rceil = ۲$$

بنابراین آرایه دارای ۳×۲×۲ خانه است.■

✱ آرایه زیر دارای چند عضو است؟

```
int   x[2 ][ ][3]={1, 2, 3,4, 5, 6,7,8};
```

حل : تعریف نادرست است. چون در آرایه های n بعدی، در صورتی که مقداردهی اولیه شده باشد، فقط بعد اول آرایه را می توان محاسبه کرد.

■

✱ تعداد عناصر آرایه زیر را بدست آورید؟

```
float   k[ ][2][4][3]={1,2,3,4,5,6,7,8,9,10,11,12,13,14};
```

حل: بعد اول آرایه برابر یک است.

$$\left\lceil \frac{۱۴}{۲\times۴\times۳} \right\rceil = ۱$$

بنابراین تعداد عناصر آرایه برابر است با

$$1\times2\times4\times3=24$$

■

✱ عملکرد برنامه زیر چیست؟

```
main( ){
   float  x[10],min;
   int      i,j;
```

```
 for (i=0; i<10; i++)   cin>>x[i];
  min= x[0];
  j =1;
  for (i=1; i<10 ; i++)
    if (x[i]<min )   min= x[i];
  cout<<min;
}
```

حل: ۱۰ عدد را از ورودی خوانده و در آرایه x قرار می دهد. سپس کوچکترین عدد را چاپ می کند.

■

رشته

برای ذخیره متن ها از نوعی داده‌ای به نام رشته استفاده می شود. اما در زبان C، رشته ها نوعی داده نیستند و به صورت آرایه ای از کاراکتر می باشند که به رشته تهی (NULL) ختم می شوند. بنابراین طول رشته ها را باید به اندازه یک واحد بیش از مقدار لازم در نظر گرفت. به طور نمونه برای ذخیره یک نام حداکثر سه حرفی، می توان از تعریف زیر استفاده کرد:

```
char  x[4];
```

توابع کار با رشته ها

نام تابع	عملکرد
s=gets( )	خواندن یک رشته از ورودی و ذخیره آن در s
puts(s)	نمایش رشته s در مانیتور
strlen(s)	محاسبه طول رشته s
strupr(s)	تبدیل حروف کوچک رشته s به حروف بزرگ
strlwr(s)	تبدیل حروف بزرگ رشته s به حروف کوچک
strchr(s,ch)	جستجو کاراکتر ch در رشته s
strstr(s1,s2)	جستجو رشته s2 در رشته s1
strcmp(s1,s2)	مقایسه دو رشته s1وs2
strcmpi(s1,s2)	مانند strcmp ( )، با این تفاوت که بین حروف کوچک و بزرگ تفاوتی قائل نمی‌شود.
strcat(s1,s2)	اتصال رشته s2 به انتهای رشته s1
strncat(s1,s2,n)	اتصال حداکثر n کاراکتر رشته s2 به انتهای رشته s1
strcpy(s1,s2)	کپی رشته s2 در رشته s1
strncpy(s1,s2,n)	کپی حداکثر n کاراکتر رشته s2 به رشته s1
memset(s,ch,c)	کپی کاراکتر ch به تعداد c در رشته s
strset(s,ch)	جایگزینی کاراکتر ch با همه کاراکترهای رشته s

✱ در صورت ورود ALL ↵ ، خروجی دستورات زیر چه خواهد بود؟

```
char  s[10];
gets(s);
cout<<strlen(s);
```

حل : تابع strlen طول رشته را برمی‌گرداند. بنابراین خروجی ۳ است.

■

* در مثال زیر رشته ای از ورودی خوانده شده و در آن به جای کاراکتر A ، کاراکتر B قرار می گیرد:

```
main( ){
  char   s[50];
  int     i=5;
  gets(s);
  for (i=0; s[i]; i++)
   if (s[i] =='A')
      s[i] ='B';
  puts(s);
}
```

* در صورت ورود رشته AB و CDE، خروجی دستورات زیر چه خواهد بود؟

```
char   s1[10] ,s2[10];
gets(s1);
gets(s2);
strcat(s1,s2);
puts(s1);
```

حل : توسط تابع strcat، می‌توان دو رشته را به یکدیگر متصل کرد. بنابراین خروجی ABCDE خواهد بود.

تابع strchr(s,ch) در صورت بودن کارکتر ch در رشته s، محل وجود اولین وقوع آن را مشخص می‌کند و در صورت نبودن کاراکتر در رشته، مقدار صفر را برمی گرداند.

* خروجی دستور زیر چیست؟

```
if (strchr ("ABCD" ,'B' ))
  cout <<"found";
```

حل: خروجی چاپ رشته found است. چون کاراکتر B در رشته ABCD وجود دارد.

* خروجی دستورات زیر چیست؟

```
char   *p;
p=strchr("ABCDEF",'C');
cout<<p;
```

حل : خروجی CDEF است.

تابع strstr (s1, s2) ، در صورت بودن رشته s2 در رشته s1، محل اولین وقوع را برمی‌گرداند و در صورت عدم وجود، مقدار صفر را برمی‌گرداند.

* خروجی دستورات زیر چیست؟

```
if (strstr ("ABCD" , 'BC') )
```

```
cout <<"found";
```

حل: چون رشته BC در رشته ABCD وجود دارد، پیغام found چاپ می‌شود.

■

در تابع strcmp(s1,s2)، دو رشته s1 و s2 کاراکتر به کاراکتر با هم مقایسه می‌شوند و در صورت برابر بودن آنها، مقدار صفر برگردانده می‌شود و در صورت بزرگ بودن s1 ، خروجی مثبت و در غیراینصورت خروجی منفی خواهد بود.

* عملکرد دستورات زیر چیست؟

```
char   s1[4]="ABC";
char   s2[4]="ADE";
int     i=strcmp(s1,s2);
```

حل : اختلاف کد اسکی کارکترهای B وD یعنی مقدار ۲- در متغیر i ذخیره می‌شود. (۲- = ۶۶-۶۸)

■

* خروجی دستورات زیر، چاپ رشته GGCDEF خواهد بود:

```
char   s[ ]= "ABCDEF";
memset (s,'G',2);
puts(s);
```

■

* خروجی دستورات زیر، چاپ رشته DDD خواهد بود:

```
char  s[ ] ="ABC";
strset(s,'D');
puts(s);
```

■

* خروجی برنامه زیر در صورت ورود رشته ALI چیست؟

```
main( ){
  char    s[50]; char    temp ; int   i ,j;
  gets(s);
  for(j=0; s[j]; j++);
  j--;
  for(i=0; i<j/2; i++)
        { temp = s[i];  s[i] = s[j-i];   s[j-i] = temp; }
   puts(s);
}
```

حل: برنامه یک رشته را از ورودی خوانده و معکوس آن را چاپ می‌نماید. بنابراین خروجی ILA است.

تذکر: برنامه برای رشته با طول فرد نوشته شده است.

■

*در برنامه زیر کدام خط رشته ali را چاپ نمی کند؟

```
void   main( ) {
   char  s[4];
```

```
  int    i=0;
  strcpy(s,"ali");
  do  putch (s[i]);    while ( s[i++]!='\0');          //1
  i=0 ; while (s[i++])   putch(s[i] );                 //2
  for(i=0; i< strlen(s); i++ )   putch(s[i]);          //3
}
```

حل: دستور while ، کاراکتر a رشته ali را چاپ نمی کند، چون بعد از چک شرط حلقه، مقدار i یک واحد اضافه شده و بعد از آن کارکتر i ام چاپ می شود.

■

فصل ۴ :
نوع شمارشی- استراکچر - یونیون

نوع شمارشی

به کمک کلمه کلیدی enum می توان یک نوع شمارشی تعریف کرد. به طور نمونه یک نوع به نام color با سه عنصر تعریف می کنیم:

```
enum   color{red , green , blue}c;
```

کامپایلر به هریک از عناصر نوع شمارشی، عددی را نسبت داده که معمولاً از ۰ شروع می شود و یکی یکی اضافه می شود. به طور نمونه شماره ۰ به red ، شماره ۱ به green و شماره ۲ به blue نسبت داده می شود.

می توان روند شماره گذاری را تغییر داد.

* در تعریف زیر مقدار منتسب شده به هر یک از عناصر کدام است؟

```
enum   x{a,b,c=5,d,e}t;
```

حل: مقدار هرکدام از عناصر داده شمارشی برابر است با:

a=0,b=1,c=5,d= 6,e=7

یعنی با انتساب ۵ به عنصر c، روند شماره گذاری تغییر می کند و شماره d برابر ۶ و شماره e برابر ۷ می شود.

متغیر از جنس نوع شمارشی ، ۲ بایتی است.

* تعریف زیر چند بایت اشغال می کند؟

```
enum   size{small ,medium,large,xlarge} s1,s2 ;
```

حل: دو متغیر s1وs2 ازجنس نوع شمارشی size تعریف شده است. بنابراین مقدار ۴ بایت فضا اشغال می شود.

می توان عملیات محاسباتی یا مقایسه ای را روی متغیرهای شمارشی انجام داد.

* خروجی برنامه زیر چیست؟

```
main( ){
  enum  e{x,y,z}a;
  a=z;
```

```
  a -=2;
  if(a==x)
    cout<<a;
}
```

حل: ابتدا مقدار a برابر شماره z یعنی ۲ شده و سپس دو واحد از آن کم می شود و چون شرط برقرار است، دستور چاپ اجرا می شود و مقدار صفر چاپ خواهد شد.

■

✱ خروجی برنامه زیر چیست؟

```
main( ) {
   enum   e{a,b,c,d} i;
   for (i=d ; i>=a; i-- )
      cout<<i;
}
```

حل: از متغیر شمارشی به عنوان شمارنده حلقه استفاده شده و مقادیر 3210 چاپ می شود.

■

✱ در دستورات زیر از متغیر شمارشی به عنوان اندیس آرایه استفاده شده است و کاراکتر A در خانه arr[2] قرار می گیرد:

```
enum    e {x,y,z,w}t;
char     arr[4] ;
t=w;
t --;
arr[t]='A' ;
```

■

عناصر نوع شمارشی نمی توانند رشته، کاراکتر و یا عدد باشند.

✱ خروجی چیست؟

```
enum  flower { roze, violet=3 , linda , mina};
cout<<mina;
flower  a=4;
cout<<(a==linda);
```

حل: خروجی دستور cout اول برابر 5 و خروجی دستور cout دوم برابر 1 می باشد.

تذکر: اگر در دستور آخر از پرانتز استفاده نشود، خطا تولید می شود.

■

استراکچر

ساختمان یا استراکچر، نامی‌برای مجموعه ای از متغیرهای همنوع یا غیر همنوع می‌باشد. بـرای تعریـف اسـتراکچر از دسـتور struct استفاده می‌شود که ساختار آن به صورت زیر است:

```
struct   نام ساختمان
{
    اجزای ساختمان
};
```

* تعریف یک استراکچر برای نگهداری مشخصات کارمند:

```
struct   personel{
         char   name[20];
         int     id;
         long  int salary;
};
```

که برای تعریف متغیری از نوع استراکچر کارمند، به صورت زیر عمل می‌کنیم:

```
struct  personel    x;
```

البته می‌توان بلافاصله بعد از تعریف استراکچر، متغیر را تعریف کرد:

```
struct personel{
         char   name[20]; int     id; int     salary;
}x;
```

* کل فضای اشغالی توسط تعریف زیر چند بایت است؟

```
struct   addr {
     char    city[15];
     char    street[30];
     long    int   zip;
   }x,y;
```

فضایی که هر فیلد استراکچر اشغال می‌کند را با یکدیگر جمع می‌کنیم:

city=15,street=30 ,zip=4

بنابراین متغیر از نوع استراکچر ۴۹ بایت فضا را اشغال می‌کند و چون دو متغیر x و y تعریف شده است، کل فضـای اشـغالی برابر ۹۸ بایت است.

دسترسی به اجزای ساختمان

برای دسترسی به هر جزء، از نام متغیر و عملگر نقطه و نام جزء استفاده می‌شـود. بطـور مثـال بـرای دسترسـی بـه جـزء id استراکچر کارمند، از دستور x. id استفاده می‌شود.

مقدار دهی اولیه به ساختمان

می‌توان به استراکچرها، مقدار اولیه داد. به طور نمونه برای مقدار دهی اولیه به ساختمان کارمند تعریف شده در بالا، می‌توان به صورت زیر عمل کرد:

```
struct  personel  x = {"ali"  , 1600 ,200000} ;
```

اگر از دستور زیر استفاده شود ، آنگاه مقدار salary برابر صفر در نظر گرفته می‌شود:

```
struct  personel  x = {"ali"  , 1600 };
```

انتساب ساختمان به یکدیگر

می‌توان متغیرهای ساختمان که از یک نوع هستند، را به یکدیگر انتساب داد.

* خروجی تکه برنامه زیر چیست؟

```
struct  a{
        char  name[30]; float  grade;
 }x,y;
x. grade=17.5;  y=x;
cout<< y. grade;
```

حل: مقدار 17.5 چاپ می شود. توسط دستور y=x; ، متغیر x که از نوع ساختمان داده a است را به متغیر y که از همین نوع است، منتسب می کنیم.

■

آرایه‌ای از ساختمان

می‌توان آرایه ای از ساختمان تعریف کرد که از پرکاربردترین موارد کاربردی ساختمان ها می‌باشد. بطور نمونه اگر بخواهیم مشخصات ۲۰ دانشجو را از ورودی خوانده و ذخیره کنیم، از تعریف زیر استفاده می‌شود:

```
struct  personel    x[20];
```

* تعریف زیر چند بایت از فضا را اشغال می‌کند؟

```
struct  time {
    int  hour;
    int  minute;
    int  second;
 } x[10];
```

حل : هر فیلد استراکچر ۲ بایت فضا را اشغال می‌کند، یعنی در مجموع ۶ بایت از فضا اشغال می‌شود، بنابراین آرایه x که دارای ۱۰ خانه ۶ بایتی است. مقدار ۶۰ بایت از فضا را اشغال می‌کند.

■

استراکچرهای متداخل

در زبان C می‌توان استراکچرهای تودرتو تعریف کرد. یعنی استراکچری که فیلدی از آن، خود یک استراکچر باشد. در مثال زیر، فیلد x که استراکچر تاریخ است، شامل ۳ فیلد معرف تاریخ استخدام یک کارمند یعنی سال و ماه و روز است:

```
struct date {
   int  year;
   int  month;
```

```
   int   day;
 };
struct   personel {
   char     name[30];
   struct   date  x ;
 }p;
```

در مثال بالا متغیر p ، ۳۶ بایت را اشغال می‌کند:

name=30, x=(2+2+2=6)

■

*مقدار دهی به فیلد day در مثال بالا به صورت زیر می‌باشد:

```
p.x.day=10;
```

■

*توسط دستور زیر مقدار دهی اولیه برای مثال بالا انجام می‌شود:

```
p={"Ali",1363,4,1};
```

■

*تعریف زیر چند بایت از فضا را اشغال می‌کند؟

```
struct  x{
   int    a;  char  b[3];
};
struct  y{
   float    c,d;
   struct  x  e;
}k[5];
```

حل: استراکچر x مقدار ۵ بایت از فضا را اشغال می‌کند و استراکچر y مقدار ۱۳ بایت را اشغال می‌کند. بنابراین آرایه k که دارای ۵ عضو ۱۳ بایتی است، مقدار ۶۵ بایت را اشغال می‌کند.

■

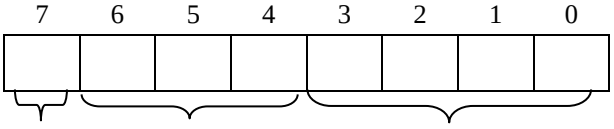
استراکچر بیتی

می‌توان استراکچری تعریف کرد که فیلدهای آن بیتی باشند.

*استراکچر بیتی s به صورت زیر تعریف شده است:

```
struct  s {
        unsigened  char  x:4;
        unsigened  char  y:3;
        unsigened  char  z:1;
}a;
```

که متغیر a مقدار ۸ بیت یعنی ۱ بایت را اشغال می‌کند.



Z y x

■

✱نتیجه اجرای دستورات زیر، با توجه به مثال قبل چیست؟

```
a.x = 10;
a.y = 3;
a.z = 0;
```

حل:

ابتدا مقدار x یعنی 10 به صورت دودویی(1010) در چهار بیت کم ارزش قرار گرفته و مقدار دودویی y یعنی 011 بعد از آن ذخیره شده و مقدار دودویی z ذخیره خواهد شد:

7	6	5	4	3	2	1	0
0	0	1	1	1	0	1	0

Z y x

■

بعد از علامت : حداکثر از عدد ۱۶ می‌توان استفاده کرد.

فیلد بیتی دارای علامت نیست و باید نوع unsigned را برای آن انتخاب کرد.

می‌توان بیتهای بلا استفاده در نظر گرفت.

✱تعریف زیر چند بایت فضا را اشغال می‌کند؟

```
struct  k{
   unsigened  a:5;   unsigened :2;  unsigened  b:1;
}s;
```

حل : متغیر s، مقدار ۸ بیت یعنی ۱ بایت را اشغال می‌کند.(بیتهای شماره ۵ و ۶ بلا استفاده است)

■

✱تعریف زیر چند بایت را اشغال می‌کند؟

```
struct  s{
   int  a;   unsigened b:10;   unsigened c:3;
}x;
```

حل : متغیر x مقدار ۴ بایت را اشغال می‌کند:

$$۲+\left\lceil \frac{۱۳}{۸} \right\rceil = ۲+۲ = ٤$$

✱خروجی دستورات زیر چیست؟

```
struct  s{
   unsigened   a: 4;    unsigened   b: 2;
 }x;
x.a=23;
cout<<x.a;
```

حل : مقدار ۲۳ در مبنای دو یعنی 00010111 ذخیره می شود. که ۴ بیت اول یعنی ۷ در متغیر بیتی a ذخیره می‌شود و خروجی چاپ عدد ۷ است.

البته می توان استراکچر را مانند کلاس تعریف کرد. در این حالت استراکچر دارای دو قسمت private و public می باشد. به مثال زیر توجه کنید:

```
struct  s{
        private : int  x;
        public:   s(int b) {x=b};
                  int  f( ) {cout <<x;}
};
void  main( ) {
  s  a(1);  a.f( );
}
```

البته اگر با کلاس آشنایی ندارید، بعد از مطالعه کلاس این نکته را مطالعه کنید.

تذکر: استفاده از کلمه public اختیاری است.(در کلاس استفاده از private اختیاری است)

یونیون ها

یونیون، محلی از حافظه است که توسط دو یا چند متغیر بطور اشتراکی مورد استفاده قرار می گیرد که این متغیرها نمی‌توانند بطورهمزمان از این فضا استفاده کنند. در واقع union ساختمانی است که آدرس شروع کلیه اجزای آن از یک نقطه است. شکل کلی تعریف یک اتحاد، به صورت زیر است:

```
union  <نام>{
    اجزاء
} نام متغیر ;
```

* با توجه به تعریف زیر، فضای اشغالی توسط متغیر z برابر ۴ بایت است، چون طول a برابر ۲ و طول b برابر ۴ است و طول b بزرگتر است.

```
union  u{
   int   a;
   float  b;
}z;
```

* بعد از اجرای دستورات زیر چه مقداری در b[1] قرار خواهد گرفت؟

```
union  u{
    int   a;
     char  b[2];
}x;
x.a = 265 ;
```

حل:

عدد ۲۶۵ در مبنای ۲ برابر است با : (۰۰۰۰۰۰۰۱۰۰۰۰۱۰۰۱) که ۸ بیت پرارزش آن معادل عدد یک در b[1] قرار می گیرد.

* خروجی دستورات زیر کدام است؟

```
union  a{int x; char  y[2];  }u;
u.y[0]=35;
u.y[1] =2;
cout<<u.x;
```

حل:

خروجی ۵۴۷ می باشد.

$$2\times256+35=547$$

✱ اندازه حافظه واحد t چند بایت است؟

```
union s{
    float     a; int       b;
    union   u{long  int d ;char e[3];}c;
 }t ;
```

حل:

متغیر a مقدار ۴ بایت، متغیر b مقدار ۲ بایت و متغیر c مقدار ۴ بایت را اشغال می کند. بنابراین متغیر t ، به اندازه بزرگترین آنها یعنی ۴ بایت فضا اشغال می کند.

■

✱ اندازه حافظه واحد t چند بایت است؟

```
union  s{
   float   a;
   int     b;
   struct  u{
             long  int d ;
             char  e[3];
             }c;
 }t ;
```

حل:

متغیر a مقدار ۴ بایت، متغیر b مقدار ۲ بایت و متغیر c مقدار ۷ بایت را اشغال می کند. بنابراین متغیر t، به اندازه بزرگتـرین آنها یعنی ۷ بایت فضا اشغال می کند.

■

✱ محتویات x[0] و x[1] چه خواهد شد؟

```
union  u{
        int    a;
       char   x[2];
}y;
y.a =100;
```

حل: مقدار صفر در x[1] و مقدار ۱۰۰ در x[0] قرار خواهد گرفت.

■

✱ تعریف زیر چند بایت از فضا را اشغال می‌کند؟

```
struct   s {
   int      i;
   float   f;
  union  u{char   ch;double  d;} a;
}b[2];
```

حل:

متغیر a از نوع union است و مقدار ۸ بایت از فضا را اشغال می‌کند. بنابراین استراکچر s با سه فیلد i و f وa ، مقدار (۸+۴+۲) یعنی ۱۴ بایت را اشغال می‌کند. در نهایت آرایه b با دو خانه ۱۴ بایتی مقدار ۲۸ بایت را اشغال می‌کند.

■

یونیون گمنام

یونیون بدون نام، نوع خاصی از یونیون می باشد که نام ندارد و نمی توان از آن اشیایی را تعریف کـرد. ایـن نـوع یونیـون بـه کامپایلر می گوید که اعضای آن می توانند از محل مشترکی استفاده کنند. برای دسترسی به عناصر ایـن یونیـون، نیـازی بـه استفاده از عملگر نقطه نمی باشد.

*خروجی دستورات زیر ۲ می باشد:

```
union  {
    int       x;
    double  y;
};
x=2;
cout<<x;
```

نکاتی در رابطه با یونیون ها در زبان c++

تذکر: اگر با مفاهیم کلاس آشنایی ندارید، این نکات را بعد از مطالعه فصل های آموزش کلاس مطالعه کنید.

۱- یونیون در c++ می تواند علاوه بر اعضای داده ای، حاوی توابع عضو نیز باشد.

۲- یونیون نمی تواند کلاس دیگری را به ارث ببرد.

۳- کلاس های دیگر نمی توانند از یونیون به ارث ببرند.

۴- توابع عضو یونیون نمی توانند مجازی باشند.

۵- اعضای یونیون نمی توانند استاتیک باشند.

۶- اعضای یونیون نمی توانند از نوع مرجع باشند.

۷- شیئی که دارای سازنده یا مخرب است، نمی تواند از یونیون استفاده کند.

فصل ۵: اشاره گر

اشاره گر متغیری است که حاوی آدرس متغیر دیگری است. در زبان C ، هر متغیر اشاره گر بدون توجه به اینکه به چه نوع داده ای اشاره می‌کند، مقدار ۲ بایت را اشغال می‌کند. فرم کلی تعریف اشاره گر بصورت زیر است:

نوع داده *نام اشاره گر ;

مثلا اشاره گر p از نوع int :

```
int  *p;
```

انواع اشاره گر

۱- اشاره گر به نوع

۲- اشاره گر به اشاره گر

۳- اشاره گر به تابع

۴- اشاره گر به void

* خروجی برنامه زیر چیست؟

```
main( ){
  int  x=5;
  int  *p;
 p=&x;
 cout <<*p;
}
```

حل: مقدار ۵ در متغیر x قرارداده می‌شود. سپس آدرس x در متغیر اشاره گر p گذاشته می‌شود. در دستور چاپ مقدار *p یعنی محتوای محلی که p به آن اشاره می‌کند، چاپ می‌شود.

■

دو عملگر*و& در مورد اشاره گرها مورد استفاده قرار می‌گیرد.

* اگر p اشاره گری از نوع int باشد و به محل 100 حافظه اشاره کند، با اجرای دستور p++ ، محتویات جدید اشاره گر p، برابر با 102 خواهد شد. (چون طول نوع int برابر 2 بایت است.)

■

بر روی اشاره گرها می‌توان فقط دو نوع عمل محاسباتی جمع و تفریق را انجام داد.

*عملکرد دستورات زیر چیست؟

```
float   x,*p;
p=&x;
cin>>p;
```

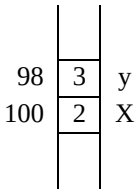
حل : عددی را از صفحه کلید گرفته و در متغیر x قرار می‌دهد.

■

✱خروجی برنامه زیر چیست؟

```
main( ){
   int   x=2,y=3,*p;
   p =&x;
   p--;
   cout<<*p;
}
```

حل : مقدار 3 چاپ می‌شود. ابتدا p به x اشاره می‌کند و بعد از دستور p-- به y اشاره خواهد کرد. (متغیر x چون زودتر از y تعریف شده است، در آدرس بیشتری قراردارد. مثلاً اگر x در آدرس 100 باشد، متغیر y در آدرس 98 قرار دارد). در این مثال متغیرها بصورت زیر در حافظه قرار می‌گیرند:

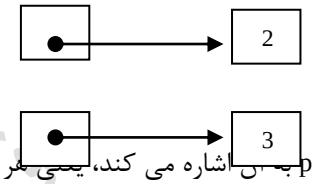


■

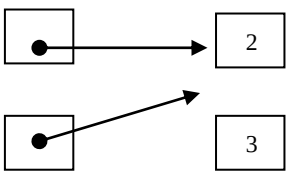
✱خروجی برنامه زیر چیست؟

```
main( ){
  int  x=2,y =3,*p,*q;
 p=&x;
 q=&y;
 q=p;
 cout<<*q;
}
```

حل: در ابتدا اشاره گر p به x با مقدار 2 و اشاره گر q به y با مقدار 3 اشاره می کند:



و بعد از اجرای دستور q=p; ، اشاره گر q به همان آدرسی اشاره خواهد کرد که p به آن اشاره می کند، یعنی هر دو به متغیر x اشاره خواهند کرد:



■

*خروجی برنامه زیر چیست؟

```
main( ){
 int   a=2,*p;
 void  *q;
 q=&a;
 p=q
 cout<<*p;
}
```

حل : اشاره گر تهی (q) برای نگهداری آدرس عدد صحیح a بکار رفته است. برای چاپ عدد، مجدداً آدرس آن را در اشاره گر همنوع عدد قرار داده و مقدار 2 چاپ می شود.

■

نوع هر متغیر با نوع اشاره گری که به آن اشاره می‌کند، باید یکسان باشد، در غیر اینصورت خطای منطقی رخ می‌دهد. برای رفع این مشکل از اشاره گر تهی (void) استفاده می‌شود که فرم تعریف آن به صورت void *p; می‌باشد.

*در برنامه زیر اگر آدرس x برابر A9 و آدرس y برابر BC باشد، آنگاه کدام درست است؟

```
int   x=1, y=2;
int   *px ,*py;
py=&y;
*py=y+1;
x =*py+1;
px=py;
*px=x+y;
```

۱) مقدار y برابر ۷ است.
۲) مقدار px برابر BC است.
۳) مقدار px* برابر ۷ است.
۴) همه موارد

حل:

گزینه ۴ درست است. ابتدا به کمک دستور py=&y; ، آدرس y در py قرار داده می شود و سپس یک واحد به متغیر y اضافه شده و برابر ۳ می شود. سپس توسط دستور x=*py+1; ، مقدار x برابر ۴ خواهد شد. و بعد از اجرای دستور px=py; ، اشاره گر px با مقدار py یعنی BC پر می شود، یعنی px به همان جایی که py اشاره می کند، اشاره خواهد کرد و در نهایت مقدار y برابر ۷ خواهد شد.

■

تغییر مقدار const به کمک اشاره گر

به کمک اشاره گر می‌توان مقدار ثابت را تغییر داد. به طور مثال برای تغییر مقدار ثابت x از ۵ به ۶ به صورت زیر عمل می کنیم:

```
const  int  x=5;
int   *p;
p= &x;
```

*p=6;

اشاره گر از نوع const

می توان اشاره گر از نوع const تعریف کرد. در این حالت می توان محتوای اشاره گـر را تغییـر داد، امـا نمـی تـوان محتـوای آدرسی که اشاره گر به آن اشاره می کند را تغییر داد. به طور مثال، اجرای دستورات زیر موجب چاپ مقدار ۲ می شود:

```
const int  *p;
int  x=2;
p=&x;
cout<<*p;
```

و اجرای دستوری مانند *p=6; موجب خطای کامپایلری می شود.

سئوال : تفاوت p با q در چیست؟

```
const int  *p;
int   *const  q;
```

حل: متغیر p اشاره گر به متغیر ثابت است و متغیر q یک اشاره گر ثابت است. بنـابراین محتـوای p و آدرس q را نمـی تـوان تغییر داد. بنابراین دستورهای زیر نادرست است:

```
*p=2;
q=&x;
```

رابطه اشاره‌گر با آرایه

ارتباطی بین اشاره گر و آرایه وجود دارد. اشاره گرها حاوی یک آدرس می‌باشند و اسم آرایه نیز یک آدرس است. بعبارتی نـام آرایه آدرس اولین عنصر آرایه را مشخص می‌کند. برای دسترسی به عناصر آرایه علاوه بر انـدیس، از اشـاره گـر نیـز می‌تـوان استفاده کرد.

✱ اگر آرایه ای به صورت int x[3]; و اشاره گری بنام p به صورت int *p; تعریف کـرده باشـیم، آنگـاه دسـتور p=&x[0]; ، آدرس شروع آرایه را در اشاره گر p قرار می‌دهد. (این دستور معادل دستور p=x; می‌باشد). در این حالـت *(p+1) ، معـادل x[1] می‌باشد و در حالت کلی *(p+i) به i +1 امین عنصر آرایه اشاره می‌کند.

✱ خروجی برنامه زیر چیست؟

```
main( ){
  int   i ;  int x[3] = {5,2,8};
  for (i=0; i<3 ;i ++)
     cout <<*(x+i);
}
```

حل: محتویات آرایه یعنی 528 چاپ خواهد شد.

اگر x یک آرایه دو بعدی باشد، آنگاه x[i][j] معادل با (x[i]+j)* می‌باشد.

اگر x یک آرایه دو بعدی باشد، آنگاه &x[i][j] معادل با x[i]+j می‌باشد.

*خروجی برنامه زیر چیست؟

```
main( ){
  int   x[5]={1,2,3,4,5};
  int  *p,*q;
  p=x;
  q=x+4;
  for( ; q>=p ; q --)
     cout<<*q;
}
```

حل: ابتدا اشاره گر p به خانه اول و اشاره گر q به خانه آخر اشاره خواهد کرد. سپس توسط حلقه محتویات آرایه از آخر به اول چاپ می شود. بنابراین خروجی 54321 می باشد.

■

*خروجی دستورات زیر کدام است؟

```
int    x[3][4]={{1,2,3,4}, {5,6,7,8} ,{9,10,11,12}};
cout<<*(x[1]+2);
```

حل: عدد 7 چاپ می شود. (منظور از (x[1]+2)* همان x[1][2] می باشد.)

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12

■

* کدام خط ها، نادرست هستند؟

```
void   main( ) {
    void   *q; int   *p; int   a[3]={5,8,2};
   q=p=a;
   a++ ;          //1
   p++;           //2
   q++;           //3
   cout<<*p;      //4
   cout<<*a;      //5
}
```

حل: خط شماره ۱ نادرست است، چون نمی توان از عملگر ++ برای اسم آرایه استفاده کرد. خط شماره ۳ نادرست است، چون اشاره گر از نوع void را نمی توان تغییر داد.

اشاره گرها و رشته ها

رشته های ارتباط بسیار نزدیکی با اشاره گرها دارند. چون رشته ها در زبان c به صورت آرایه تعریف می‌شوند.

✱ تبدیل کلیه حروف کوچک رشته s به حروف بزرگ:

```
char  *s;
gets(s);
while(*s) {
   if(*s>='a'  && *s<='z')  *s=*s -32;
     s++;
 }
```

(یادآوری : اختلاف کد اسکی حروف کوچک و بزرگ برابر ۳۲ است.)

✱ محاسبه تعداد دفعات تکرار کاراکتر A در رشته s :

```
int  c=0;
while(*s)
   if (*s++='A')  c++;
cout<<c;
```

درصورتی که رشته ها به صورت اشاره گر تعریف شوند، بخصوص در مواقعی که طول عناصر مختلف آن متفاوت باشند، موجب صرفه جویی در میزان حافظه می‌گردند.

✱ تعریف زیر موجب می‌شود که ۲۴ بایت از حافظه اشغال شود:

```
char    name[3][8]={"omid","ali","mohamad"};
```

	0	1	2	3	4	5	6	7
0	o	m	i	d	\0			
1	a	l	i	\0				
2	m	o	h	a	m	a	d	\0

در هنگام استفاده از آرایه دو بعدی، تعداد سطرها برابر با تعداد افراد و تعداد ستون ها برابر با طول طولانی ترین نام است.

```
char   *name[3]={"omid","ali","mohammad"};
```

0	o	m	I	d	\0			
1	A	l	I	\0				
2	m	o	H	a	m	a	d	\0

تعریف بالا ۶ بایت از حافظه را اشغال می‌کند. چون آرایه دارای ۳ خانه از نوع اشاره‌گر است.

* خواندن تعدادی نام از ورودی و قرار دادن در یک آرایه :

```
main( ){
  char    x[20][81];
  char   *p[20];
  int      c=0;
  while(c<20)     {
        gets(x[c]);
        if (strlen (x[c] ) == 0 )
              break;
       p[c]=x[c];
       c ++;
    }
}
```

✱ عملکرد دستورات زیر چیست؟ (با توجه به مثال قبل)

```
for (i =0 ; i <c-1; i++)
  for (j=i+1;j <c; j++)
    if (strcmp(p[i],p[j])>0)
        {  temp=p[j];  p[j]=p[i];  p[i]=temp;    }
```

حل : مرتب کردن اسامی‌دریافت شده از ورودی در مثال قبل.

■

✱ خروجی دستورات زیر چیست؟

```
char  * s[ ]={"ali","farshid","omid"};
cout<<s[1];
cout<<*s[1];
```

حل: خروجی دستور cout اول، چاپ رشته farshid است. خروجی دستور cout دوم، چاپ کاراکتر f است.

■

دستور ;p*++ ، یعنی افزایش محتوای آدرسی که p به آن اشاره می‌کند و دستور ;p++* ، یعنی محتوای آدرس بعدی.

✱ در برنامه‌ی روبه‌رو، خروجی کدام است؟

```
main( ){
   char    *p="ALIREZA";
   cout <<*p;
   cout <<*++p;
   cout <<++*p;
}
```

حل: کاراکترهای ALM چاپ می شوند. دستور اول کاراکتر A را چاپ کرده و توسط دستور دوم اشاره گر به کاراکتر بعدی، یعنی L اشاره کرده و آن را چاپ می کند. سپس توسط دستور سوم کاراکتر بعد از L در حروف الفبای انگلیسی، یعنی M چاپ می شود.

■

✱خروجی دستورات زیر کدام است؟

```
char   *x ="A";
char    y = 'B';
char    z[2]="C";
cout <<*x;
cout <<y;
cout <<z[0];
```

حل: کاراکترهای ABC چاپ می شوند. (توجه کنید که zوx رشته می باشند و هنگام مقدار دهی نیاز به دابل کوتیشن دارند.)

■

* عملکرد برنامه زیر کدام است؟

```
char   *p1,*p2, a[10] ,b[10];
char    m[ ] ="ABCD";
i=0;
while ((a[i]=m[i])!='\0')
      i++;
p1=m; p2=b;
while((*p2++ =*p1++)!= '\0');
```

حل: رشته ABCD در آرایه های a وb کپی می شود. توسط حلقه اول رشته m در آرایه a کپی شده و توسط حلقه دوم، رشته m به کمک اشاره گرها در آرایه b کپی می شود.

اشاره گر به استراکچر

می توان یک متغیر اشاره گر از نوع استراکچر تعریف کرد :

```
struct  s{  int  x;  char  y; }*p;
```

در این حالت برای دسترسی به فیلدهای استراکچر از عملگر -> باید استفاده کرد. به طور نمونه:

```
p->x =5;
```

اشاره گر به اشاره گرها

متغیر اشاره گر به اشاره گر، حاوی آدرس یک متغیر اشاره گر است و برای تعریف آن از دو علامت * استفاده می‌شود.

* خروجی برنامه زیر چیست؟

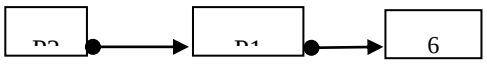
```
main( ){
  int   x ,*p ,**q;
  x=5;  p=&x;
  q=&p;
  cout<<**q;
}
```

حل: عدد 5 چاپ می شود. متغیر اشاره گر p به x اشاره می‌کند و متغیر اشاره گر q به p اشاره می‌کند.

* خروجی برنامه زیر چیست؟

```
main( ){
  int  x =6, *p1,**p2;
  p2=&p1;  *p2=&x;
 **p2=*p1-1;
  cout<<x;
}
```

حل : عدد ۵ چاپ می شود. بعد از اجرای دستور p2=&p1; ، اجرای دستور *p2 = &x;، معادل برقراری اتصال خط p1 به x خواهد شد:



اشاره گر آویزان

به برنامه زیر توجه کنید:

```
int  main( ){
  int   a=1;
  int   *p;
  *p=a;
}
```

در این برنامه به متغیر اشاره گر p ، آدرسی نسبت داده نشده است و یک اشاره گـر آویـزان محسـوب مـی شـود. در واقـع در آدرسی که به طور پیش فرض p به آن شاره می کند، مقدار ۱ قرار داده شده است.در واقع باید از دستور p=&a; استفاده می شد.

زباله

به برنامه زیر توجه کنید:

```
int  main( ){
  int   a=1,b=2;
  int   *p,*q;
  p=&a;
  q=&b;
  p=q;
}
```

ابتدا آدرس متغیر a در p و آدرس متغیر b در q ذخیره می شود ولی بعد از اجرای دستور p=q; ، اشاره گر p به همان محلی که اشاره گر q اشاره می کند، اشاره خواهد کرد. در نتیجه متغیر a دیگر از طریق اشاره گر p قابل دسترس نخواهد بـود و بـه یک زباله تبدیل می شود.

متغیرهای پویا

می‌دانیم که با اجرای یک برنامه و قرار گرفتن آن در حافظه، سگمنت های داده (DS) ، کـد (CS) و پشـته (SS) در حافظـه قرار می‌گیرند. هر سگمنت ۶۴ کیلو بایت را اشغال می‌کند و بقیه حافظه RAM را که در اختیار برنامه نیست را حافظـه پویـا (Heap) می‌گویند. متغیرهای سراسری در DS ، متغیرهای محلـی در SS و دسـتورالعملهای برنامـه در CS قـرار می‌گیرنـد. بنابراین برای رفع محدودیت اندازه ۶۴ کیلوبایتی از حافظه پویا استفاده می‌شود. برای گرفتن فضا از حافظه پویا از تـابع new و برای آزاد کردن فضای گرفته شده از تابع delete استفاده می‌شود.

تذکر: در زبان c از malloc به جای new و از free به جای delete استفاده می شود.

تذکر: کار با دستور new ساده تر از malloc می باشد. در دستور new نیازی به استفاده از sizeof برای محاسبه اندازه شی نمی باشد و نیازی به قالب بندی خروجی نیست. همچنین می توان مقدار دهی اولیه نیز انجام داد.

متغیر مرجع

در زبان c++ می توان متغیرهای مستقلی از نوع مرجع تعریف کرد. مرجع، نوع خاصی از اشاره گر است،که فقط می تواند به یک متغیر اشاره کند و این متغیر باید در هنگام تعریف متغیر مرجع مشخص شود. به عبارتی مرجع، نام دیگر برای متغیر است. متغیرهای مرجع می توانند به عنوان پارامتر تابع و یا مقدار برگشتی تابع نیز به کار روند. در هنگام فراخوانی نیاز به استفاده از & نمی باشد. در مثال زیر متغیر b، مرجع مستقل می باشد و هیچ تفاوتی با متغیر a ندارد:

```
main( ) {
  int  a;
  int  &b;
  a=1;
  b=a;
  ...
}
```

تذکر: توابع در زبان c++ می توانند رفرنس نیز برگردانند که در زبان c این کار ممکن نمی باشد.

✱ خروجی برنامه زیر 53 است:

```
main( ){
  int   a;
  int   &p=a;
  a=5;
  cout<<p;
  p=3;
  cout<<a;
}
```

■

✱خروجی برنامه زیر ۴ می باشد:

```
int  a=1;
int  &f( ) { return  a; }
main( ){
 f( ) =2;
 cout<< a + f( );
}
```

مقدار متغیر سراسری a بعد از خط اول تابع main برابر ۲ می شود و در خط cout مجموع خروجی تابع (یعنی ۲) و مقدار a (یعنی ۲) چاپ می شود.

■

آرایه ای از مرجع ها نمی توان تعریف کرد.

- اشاره گر به مرجع نمی توان تعریف کرد.
- برای یک مرجع نمی توان مرجع دیگری تعریف کرد.
- آدرس یک مرجع را نمی توان به دست آورد.

فصل ۶

تابع

برنامه های کوتاه فقط شامل یک تابع اصلی به نام main می‌باشند ولی در برنامه های طولانی و پیچیده که از چند بخش تقریباً مستقل تشکیل شده اند، برای هرقسمت، یک برنامه نوشته و سپس آنها را به هم متصل می‌کنیم. هریک از این قسمتها را یک زیر برنامه می‌گویند، که در زبان c به صورت شکل زیر است :

```
(نام پارامترها ) < تابع> < نوع خروجی تابع>
    تعریف پارامترها
{
   بدنه تابع
}
```

- اگر نوع خروجی تابع مشخص نشود، نوع int در نظر گرفته می‌شود.
- تعیین نوع پارامترها در حین تعریف تابع نیز امکان پذیر است.

✱ خروجی برنامه زیر چیست؟

```
int  f (int);
main( ){
   int   x=3;
   cout<< f (x);
}
int  f(int a)
{
  return(a+2);
}
```

حل: ابتدا در main( )، یک تعریف اولیه از تابع f اعلام شده و سپس تابع f با مقدار ۳ صدا زده می شود و تابع f مقدار ۵ را بر می گرداند.

- برای تعریف متغیرهای رسمی تابع f می توان به یکی از روشهای زیر عمل کرد:

```
int   f (x,y )
  int    x;   char  y;
{
...
}
```

و یا :

```
int   f (int  x , char  y)
{
```

```
  …
}
```

* برنامه ای که شعاع دایره ای را از ورودی خوانده و مساحت دایره را محاسبه کرده و در خروجی نمایش می دهد:

```
float   area(float);
main( ){
  float   x;
  cin>>x;
  cout<< area(x);
}
float   area(float  r)
{
 return (3.14*r*r);
}
```

*خروجی برنامه زیر چیست؟

```
int  f1(int, int);
main( ){
    int   x,y;
    x=2;
    y=4;
    cout<< f1(x, y);
}

f1(int a,b)
  {return (f3 (f2(a) , f2(b))); }
f2(int c)
    {return(c+c); }
 f3(int  m,n)
  {return(m*n); }
```

حل : مقادیر ۲ و۴ به تابع f1 فرستاده می‌شود و سپس تابع f2 یکبار با مقدار ۲ فراخوانی شده و مقـدار ۴ را برمی‌گردانـد و یکبار با مقدار ۴ فراخوانی و مقدار ۸ را برمی گرداند. در نهایت تابع f3 بـا مقـادیر ۴ و ۸ فراخـوانی می‌شـود و مقـدار ۳۲ را برمی‌گرداند و در تابع اصلی چاپ می‌شود.

بعضی از توابع مقداری را به تابع صدا زننده بر نمی‌گردانند. نوع خروجی این توابع void می باشد.

*خروجی برنامه زیر، چاپ مقدار ۶ است:

```
 void   f (int);
main  ( ){
```

```
 int   x=3;
 f(x);
}
void   f (int  a)
{
   cout<<a*2;
 }
```

متغیرهایی که در یک تابع تعریف می‌شوند، فقط در همان تابع قابل استفاده هستند و به محض برگشت کنترل اجرای برنامه از آن تابع ، این متغیرها از بین می‌روند. به این متغیرها، محلی (Local) می‌گویند.

✱خروجی برنامه زیر چیست؟

```
f( );
main( ){
  int  a=5;
  f( );
  cout<<a;
}
f( ){
     int a=6;
     cout<<a;
 }
```

حل : ابتدا مقدار ۶ در تابع چاپ می‌شود و سپس مقدار ۵ در تابع ( )main چاپ می‌شود.

تذکر: متغیر a داخل ( )main از متغیر a داخل ( ) f کاملاً جدا می‌باشد و در بخش مجزا در حافظه قرار دارند.

✱عملکرد تابع f چیست؟

```
void  f(char  *s1, const  char  *s2)
{
  while(*s1 != '\0')
     ++s1;
  for(  ;*s1=*s2 ; s1++ ; s2++);
}
```

حل: دو رشته s1 وs2 را به عنوان آرگومان پذیرفته و رشته s2 را به انتهای رشته s1 اضافه می کند.

انواع فراخوانی تابع در زبان c

۱- فراخوانی توسط ارزش (call by Value)

۲- فراخوانی توسط ارجاع (call by Reference)

در روش call by Value ، مقدار آرگومان تابع در پارامتر متناظر با آن کپی می‌شود. بنابراین هرگونه تغییری در پارامترها ، هیچگونه تاثیری در مقدار آرگومان نخواهد داشت. یعنی هیچ مقداری توسط پارامترها و آرگومان ها به تابع صدا زننده برگردانده نمی‌شوند. در روش call by Reference ، آدرس آرگومان به جای آرگومان در پارامتر کپی می‌شود. بنابراین در داخل تابع صدا زننده، آدرس آرگومان برای دسترسی به آرگومان مورد استفاده قرار می‌گیرد. بنابراین هر تغییری در پارامتر موجب تغییر در آرگومان خواهد شد.

*خروجی چیست؟

```
f (int  );
main( ){
  int  a=3;
  f(a);
 cout<< ("%d",a);
}
f (int   x) {
   x=5;
 }
```

حل: عدد ۳ چاپ می شود. (روش call by Value)

* خروجی چیست؟

```
f (int  *) ;
main( ){
 int   a=3;
 f(&a);
 cout<<a;
}
f (int  *x)
  {*x=5;}
```

حل: عدد ۵ چاپ می شود. (روش call by Refrence)

*خروجی برنامه زیر چیست؟

```
k(int  *,int) ;
main( ){
  int   x,y;   x=3;   y=5;
  k(&x,y);
  cout<<x ,y;
}
k (int  *a ,int  b) {
   *a=*a+3;
```

```
    b=b+1;
}
```

حل : عدد ۶۵ چاپ خواهد شد. مقدار جدید متغیر x یعنی ۶ برگردانده می‌شود، اما مقدار y تغییری نمی‌کند.(توجه کنید که متغیر x به روش با ارجاع و متغیر y به روش با مقدار فراخوانی شده است.) ■

متغیرهای محلی (Local) و سراسری (global)

متغیرهایی که در داخل یک تابع و پایین یک جفت آکولاد تعریف شوند، محلی نام دارند. این متغیرها فقط در محدوده همان بلوک شناخته شده اند. همچنین متغیرهایی که قبل از تابع main تعریف می‌شوند، سراسری نام دارند. همه توابعی که در پایین main تعریف می شوند، متغیرهای سراسری را می‌شناسند.

- متغیرهای محلی در پشته و متغیرهای سراسری در data segment ساخته می‌شوند.
- مقدار اولیه متغیرهای محلی، نامعلوم و مقدار اولیه متغیرهای سراسری، صفر است.

✱دریافت ۳ عدد از ورودی و محاسبه مجموع مربعات آنها :

```
int   x,y;
int   s=0;
f1( ){ cin>>x;  f2( ); s+=y;}
f2( ){ y=x*x ;}
main( ){
  int   i;
  for (i=0; i< 3 ; i++)
       f1( )
 cout<<s;
}
```

در این مثال، متغیرهای s, y,x سراسری هستند و از آنها در توابع f1( ) و f2( ) بدون تعریف این متغیرها، استفاده شده است. ■

- درصورتی که یک متغیر محلی همنام با یک متغیر سراسری باشد، با تغییر متغیر محلی در تابع، تغییری در متغیر سراسری همنام با آن ایجاد نمی‌شود.

✱خروجی برنامه زیر چیست؟

```
int    a;
f2( ){ int a; a=5;}
f1( ){ f2( ); cout<< a; }
main( ){
 a=10;
 f1( );
}
```

حل : متغیر a که خارج از main( ) تعریف شده است، سراسری و متغیر a که در داخل f2( ) تعریف شده است، محلی می‌باشد. بنابراین مراجعه به a داخل f2( )، مراجعه به متغیری است که در f2( ) تعریف شده است و تغییر آن تاثیری در مقدار متغیر سراسری a نخواهد داشت و خروجی ۱۰ خواهد بود.

■

در مواقعی که توابع موجود در برنامه به متغیرهای مشترکی نیاز داشته باشند، بهتر است از متغیرهای سراسری استفاده شود.

متغیرهایی که به ندرت در سراسر برنامه مورد استفاده قرار می‌گیرند، بهتر است به صورت متغیر سراسری تعریف نشوند، چون میزان حافظه زیادی اشغال خواهد شد. در شروع اجرای برنامه برای متغیرهای سراسری، حافظه اختصاص می‌یابد و بعد از پایان اجرای برنامه، این حافظه از متغیرها گرفته می‌شود.

✱ خروجی کدام است؟

```
void  main( ){
   int  x=2;
   cout<<x;
   {
    int  x=3;
    cout<<x;
   }
   cout<<x;
}
```

حل: خروجی 232 است.

تذکر: اگر int قبل از x ای که مقدار 3 دارد را بردارید، خروجی 233 می شود.

■

عملگر ::

می توان به کمک عملگر :: به محتویات یک متغیر سراسری در تابعی که دارای متغیری هم نام با آن می باشد، دسترسی پیدا کرد.

✱خروجی چیست؟

```
int   x=3;
void  main( ){
  int  x=1;
  x=::x-x;
  cout <<::x<<x;
 }
```

حل: در این برنامه برای تشخیص متغیر x سراسری از x محلی، قبل از x سراسری از عملگر :: استفاده شده است. خروجی این برنامه 32 می باشد:

■

ارسال آرایه به تابع

می‌توان آرایه را به عنوان آرگومان تابع ذکر کرد. در این حالت از اسم آرایه بدون ذکر اندیس استفاده می‌شود.

* برنامه زیر ۱۰ عدد را از ورودی خوانده و در آرایه ای قرار می‌دهد و سپس بزرگترین عنصر آرایه را توسط تابع max پیدا کرده و چاپ می‌کند.

```
max(int  [ ] );
main( ){
   int  a[10];
   int  i;
   for (i=0; i<10;i++)
       cin>>a[i];
  cout<< max(a);
}
max(int b[ ] ){
  int   k, m;
  m=b[0];
  for(k=0; k <10;k ++)
     if (b[k] > m)
         m=b[k];
 return(m);
}
```

تذکر: در تعریف max می‌توان به جای int b[ ] از int *b استفاده کرد.

*خروجی برنامه زیر چیست؟

```
f(int  [ ]);
main( ){
 int    x [5]={1,2,3,4,5};
 int    m;
 m=f(x[3]);
 cout<<m;
}
f(int  y[ ]){
    return(y[0]);
 }
```

حل : آرایه x از خانه با اندیس ۳ به بعد صدا زده می‌شود و مقدار خانه اول آرایه ارسال شده توسط تابع برگردانده می‌شود. بنابراین خروجی ۴ است.

*خروجی برنامه زیر چیست؟

```
void   k(int  a[ ][2]);
main( ){
 int    x[3][2]={1,2 ,3,4, 5,6};
 k(x);
```

```
}
void   k(int  a[ ][2]){
  cout<< a[2][1];
}
```

حل: در واقع در این مثال نحوه ارسال آرایه دو بعدی به تابع نشان داده شده است.در هنگام صدا زدن جلوی نام آرایه از براکت استفاده نمی‌شود و در هنگام معرفی در تابع k، براکت اول خالی و براکت دوم شامل تعداد ستونها می‌باشد. بنابراین خروجی ۶ است.

برای ارسال آرایه چند بعدی به تابع، تمام ابعاد به جز اولین بعد را باید مشخص کرد.

*خروجی برنامه زیر چیست؟

```
void    f (char  [ ][5] ,int );
void   main( ){
   char   x[ ][5]={"AB","mnp","AB","OR"};
   f(x,4);
   cout<< x[2] ;
}
void   f (char  y[ ][5] , int row){
  int  i;
  for (i=0; i <row ;i ++)
    if (!strcmp (y[i] ,"AB"))
      strcpy (y[i],"EF");
}
```

حل: تابع f همه رشته هایی که برابر AB هستند را به رشته EF تبدیل می‌کند. بنابراین خروجی x[2] برابر EF خواهد بود.

استفاده از مقادیر پیش فرض

اگر به پارامتری در هنگام صدا زدن تابع، مقداری داده نشود، از مقدار پیش فرض اش استفاده می کند.

* در برنامه ی رو به رو، خروجی کدام است؟

```
int  f( int  =1, int  =2);
void  main( ){
  cout  <<f( );
  cout << f(3);
  cout <<f(4,5);
}
int  f( int  a , int  b)
{
    return  a+b;
}
```

حل: خروجی دستور اول 3، دستور دوم 5 و دستور سوم 9 می باشد.

می توان تابع f را قبل از main قرار داد که دیگر لزومی به خط prototype ندارد:

```
int  f( int  a=1 , int  b=2)
{
  .........
}
void  main( ){
.....
}
```

اشاره گر به تابع

برای نگهداری آدرس تابع ها از اشاره گر تابعی استفاده می‌شود. توسط اشاره گر به تابع می‌توان تابعی را بعنوان آرگومـان بـه توابع دیگر فرستاد. فرم کلی تعریف اشاره گر تابع بصورت زیر است:

نوع داده (*نام اشاره گر) (لیست پارامترها);

*تعریف زیرمعرف این است که متغیرp می‌تواند آدرس همه تابع هایی را که پارامتر آنها از نوع int است و خروجـی آنهـا از نوع float باشد، را نگهداری کند.

```
float   (*p) (int a);
```

مثلاً به کمک اشاره گر می‌توان تابع float f(int); را فراخوانی کرد:

```
p =f;
n =(*p) (m);
```

که معادل n= f(m) است.(m از نوع int و n از نوع float است.)

■

* در برنامه زیر p اشاره گر به تابع f است و خروجی چاپ پیغام ok است:

```
void  f( );
main( ) {
   void   (*p)( );
   p=f;
   p( );
}
void   f( ) {
   cout<< "ok";
 }
```

■

خروجی یک تابع می‌تواند از نوع اشاره گر باشد. تابع strchr که از توابع کتابخانه ای رشته ای است از این نوع محسوب می‌شود. خروجی این تابع آدرس اولین محل وقوع یک کاراکتر در یک رشته می‌باشد.

```
char   *strchr  (char  *s,  char  ch)
{
  int i;
  for ( i=0 ; s[i]! = '\0' ; i ++)
       if (ch == s[i])
            return (&s[i]) ;
  return   NULL;
}
```

در روش انتقال یک اشاره گر به تابع، بالاترین سطح دسترسی وقتی فراهم می شود، که اشاره گر غیر ثابت بـه داده ی غیر ثابت داشته باشیم. چون وقتی که اشاره گر ثابت باشد، امکان تغییر آن درون بدنه تابع نمی باشد.

تابع بازگشتی

تابعی که درون بدنه خود، خود را فراخوانی کند، بازگشتی (Recursive)می‌باشد. ایـن توابـع دارای یـک شـرط جهـت انجـام فراخوانی ها دارند. از مثالهای معروف می‌توان فاکتوریل را نام برد.

*خروجی تابع زیر به ازای n=4 چقدر است؟

```
int   fact (int  n){
  if (n== 1)
        return  1;
 else
        return  n* fact (n-1);
}
```

حل :

fact (4) = 4* fact (3) = 4 * 6 = 24
fact (3) = 3* fact (2) = 3 * 2 = 6
fact (2) = 2* fact (1) = 2 * 1 = 2

■

*خروجی تابع زیر به ازای n=4 , m=3 چیست؟

```
int   mul  (int   m, int  n){
  int   ans;
  if (n==1)   ans=m;
  else  ans=m+mul (m,n-1);
  return(ans);
}
```

حل :

mul(3,4) = 3+ mul (3,3)
mul(3,3) = 3+ mul (3,2)
mul(3,2) = 3+ mul (3,1)

و چون mul (3,1) =3 ، داریم:

mul(3,2) = 3+3 = 6
mul(3,3) = 3+6 = 9
mul(3,4) = 3+9 =12

یعنی خروجی ضرب ۳ و۴ یعنی ۱۲ می‌باشد.■

* تابع زیر چه عملی انجام می دهد؟

```
int   f( int  a, int b) {
    if(b==1)    return  a;
   else    return(f (a,b-1)*a);
}
```

حل: تابع داده شده a را به توان b می رساند. مثلا :

```
f(2,3)=f(2,2)*2=4*2=8
f(2,2)=f(2,1)*2=2*2=4
f(2,1)=2
```

* عملکرد تابع زیر کدام است؟

```
int   k(char  ch,const  char  *str){
   int   x;
   if(str[0] == '\0')
         x=0;
  else  if(ch == str[0] )
              x=1+k(ch,&str[1]);
        else
              x=k(ch ,&str[1]);
 return(x);
}
```

حل: تابع k به طور بازگشتی، تعداد کاراکترهای ch موجود در رشته str را می شمارد.

* عملکر تابع زیر چیست؟(آرایه x دارای ۳ خانه شامل اعداد ۵و۶و۷ می باشد)(n=3)

```
int  f(int  x[ ],int  n)
{
 if(n==1)
    return  x[0];
 else
    x[n-1]+f(x,n-1);
}
```

حل: تابع f، مجموع خانه های آرایه را محاسبه می کند:

```
f(x,3)=x[2]+f(x,2)=7+11=18
f(x,2)=x[1]+f(x,1)=6+5=11
f(x,1)=x[0]=5
```

*عملکرد تابع زیر کدام است؟ (n : تعداد عناصر آرایه)

```
void   f(int  x[ ]  , int  n)
{
  if(n>0)
     {
      f(&x[1],n-1);
      cout<<x[0];
     }
}
```

حل: چاپ محتویات آرایه **x** از انتها به ابتدا.

کلاس های حافظه

توسط کلاس حافظه طول عمر متغیر و حوزه متغیر مشخص می‌شود. انواع کلاس‌های حافظه عبارتند از:

۱- اتوماتیک (auto) ۲- ثبات (register) ۳- استاتیک (static) ۴- خارجی (extern)

کلاس حافظه اتوماتیک

متغیرها از این نوع کلاس با فراخوانی تابع ایجاد و با خاتمه اجرای تابع از بین می‌روند. کلیه متغیرهای محلی دارای این نوع کلاس هستند، بنابراین نیازی به ذکر کلمه auto برای آنها نمی‌باشد.

کلاس حافظه ثبات

متغیر از نوع کلاس register ، به جای حافظه RAM در ثبات ها ذخیره می‌شود. در این حالت سرعت انجام محاسبات سریعتر می‌شود. البته چون تعداد ثابت های CPU محدود است، بهتر است فقط متغیرهای مهم برنامه را از نوع این کلاس در نظر بگیریم.

متغیرهای سراسری را نمی‌توان از نوع کلاس register در نظر گرفت.

برای متغیر از این نوع کلاس، آدرس مفهومی‌ندارد. یعنی از عملگر & برای آنها نباید استفاده شود.

*در زیر مثالی برای کلاس register آورده شده است:

```
register  int  i ;
for (i=0 ; i <=60000 ; i ++);
cout<<i;
```

کلاس حافظه استاتیک

متغیرها از این نوع کلاس، فقط یکبار مقداردهی اولیه می‌شوند و هنگام خروج از تابع، آخرین مقدار خود را حفظ می‌کنند.

*خروجی برنامه زیر چیست؟

```
f( );
main( ){
  int  i;
  for (i=0;i <3;i ++)
       f( );
}
f( ){
   static  int  x=5;
   cout<<x;
   x--;
```

```
}
```

حل : متغیر x از نوع استاتیک است و فقط یکبار در بار اول مقداردهی اولیه می‌شود. بنابراین خروجی ۵۴۳ است. ■

کلاس حافظه خارجی

برنامه های طولانی را بهتر است به چند قسمت منطقی کوچکتر (unit) تقسیم کرد و هر قسمت را در یک فایل قرار داده و بعد از کامپایل آنها به طور مجزا، همه قسمت ها را با یکدیگر اجرا کرد. در صورتی که بخواهیم از متغیرهای تعریف شده در واحد اصلی، در واحدهای فرعی نیز استفاده کنیم، بدون اینکه در واحدهای فرعی حافظه ای به آنها اختصاص داده شود، باید در واحدهای فرعی توسط کلمه کلیدی extern به کامپایلر گفته شود که این متغیرها در واحد اصلی تعریف شده اند.

✱ مثال زیر، کاربرد extern را نشان می‌دهد:

File1

```
int  a,b ;


main( )
{
.
.

}
f1 ( )
{
a=6;
}
```

File2

```
extern  int  a,b ;


f2 ( )
{
a = b+1 ;
.
.
}
f3 ( )
{
b=3
}
```

در این مثال کلمه کلیدی extern به کامپایلر می‌گوید که متغیرهای b,a قبلاً در جای دیگری تعریف شده اند، لذا در File2 ، حافظه جدیدی به آنها اختصاص نمی‌یابد.

آرگومانهای تابع اصلی ( )main

تابع ( )main دارای دو پارامتر به نام های argc و argv می‌باشد. تعداد آرگومان های خط فرمان در argc قرار دارد و argv به آرایه ای رشته ای اشاره می‌کند که عناصر آن به آرگومانهای خط فرمان اشاره می‌کنند. به عنوان مثال فرض کنید برنامه ای به نام add.c نوشته، توسط کامپایلر C ترجمه شده و برنامه ای به نام add.exe از آن می‌سازیم. برای اجرای این برنامه در سیستم عامل به صورت زیر عمل می‌کنیم.

```
>add  2  5
```

عملکرد این برنامه، جمع دو عدد دریافتی از طریق آرگومانهای ورودی است که با یک فاصله از هم جدا شده اند. در این حالت argc شامل عدد ۳ می‌باشد و argv [0] حاوی نام برنامه یعنی add.exe و argv [1] حاوی آرگومان اول یعنی ۲ و argv[2] حاوی آرگومان دوم یعنی ۵ است.

* برنامه زیر یک عدد را بعنوان آرگومان پذیرفته و عمل شمارش معکوس از آن عدد تا صفر را انجام می‌دهد:

```
main (int  argc,char *argv[ ] ){
  int  i;
  for(i=atoi(argv[1]); i ;i --)
       printf ("%d",i);
}
```

در مثال بالا از تابع atoi برای تبدیل مقدار عددی رشته ای به مقدار عددی صحیح استفاده شده است. چون پارامترهای ورودی به عنوان رشته محسوب می‌شوند.

- حداکثر تعداد آرگومان ها برابر ۳۲۷۶۷ می‌باشد.
- جداکننده در آرگومانهای خط فرمان، blank وTab است.

* با توجه به اجرای خط فرمان زیر، argc و argv را مشخص کنید؟

A :\ > test 21, 32 43 ↵

حل : argc برابر ۳ خواهد بود و argv به صورت زیر می باشد:

```
argv[0] = test
argv[1] = 21, 32
argv[2] = 43
```

در واقع کاراکتر کاما، جداکننده محسوب نمی‌شود.

دسته‌بندی موضوعی آموزش‌های فرادرس، در ادامه آمده است:

			
برنامه نویسی – کلیک (+)	آموزش های دانشگاهی و تخصصی – کلیک (+)	هوش مصنوعی و یادگیری ماشین – کلیک (+)	مهندسی برق الکترونیک و روباتیک – کلیک (+)
			
دروس دانشگاهی – کلیک (+)	مباحث مشترک – کلیک (+)	مهارت های دانشگاهی – کلیک (+)	نرم افزارهای تخصصی – کلیک (+)
			
مهندسی نرم افزار – کلیک (+)	نرم افزارهای عمومی – کلیک (+)	طراحی و توسعه وب – کلیک (+)	آموزش های عمومی – کلیک (+)

منبع مطالعاتی تکمیلی مرتبط با این کتاب

آموزش ویدئویی برنامه نویسی C++

عمومیت زبان C++ در میان زبان‌های برنامه‌نویسی بسیار بالا است و می تواند به عنوان اولین زبان نیز یاد گرفته شود و به پیش نیاز دیگر احتیاج نباشد.

مجموعه فیلم‌های آموزشی برنامه‌نویسی C++، با این فرض تهیه شده است که مخاطب هیچ دانش و تجربه قبلی در زمینه برنامه‌نویسی ندارد و در این مجموعه آموزشی، همه مباحث با بیان و تشریح مبانی نظری و سپس با پیاده‌سازی گام به گام مثال‌های عملی آموزش داده می‌شوند و از این نظر، در ایجاد یک دانش عمیق در زمینه برنامه‌نویسی، بسیار کارآمد است.

مدرس: مهندس فرشید شیر افکن

مدت زمان: ۲۰ ساعت

جهت مشاهده آموزش ویدئویی این آموزش – کلیک کنید **faradars.org /fvcp9504**