

شیء گرایى در C++

مؤلف: فرشىد شىرافكن

دانشجوى دكترى بىوانفورماتىك دانشگاه تهران



ناشر: سازمان علمى 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 سوالات باشد؛ ارائه نشده است. همین موضوع، کنکور را به ویژه کنکور کارشناسی ارشد به عنوان یک آزمون متمرکز و سراسری، از اهمیت دو چندانی برخوردار می‌کند.

یکی از آسیب‌های همراه با این آزمون سراسری این است که فضای رقابتی آن با ایجاد موسسات 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 گام‌های دیگر خود را در راستای آرمان فرادرس برداشتیم. کتاب حاضر که حاصل نزدیک به دو 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 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، می‌توانند 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.



فرادرس

فراتر از یک کلاس درس
www.faradars.org

سازمان علمی آموزش فرادرس

بزرگترین پلتفرم آموزش آنلاین ایران

وب: www.faradars.org

منبع مطالعاتی تکمیلی مرتبط با این کتاب

آموزش ویدئویی شیء‌گرایی در C++

در آموزش مقدماتی C++ (سی پلاس پلاس)، مفاهیم اولیه برنامه نویسی در ++C آموزش داده شد. (انواع داده ها، دستورهای شرطی، دستورهای تکرار، آرایه، اشاره گر، استراکچر (Structure) آموزش داده شده است.)

در این آموزش، مفاهیم شیء‌گرایی در ++C آموزش داده شده است. مفاهیمی مانند: کلاس، تابع دوست، کلاس دوست، ارث بری، پلی مورفیزم (Polymorphism)، سر بارگذاری، قالب. در این آموزش، ابتدا مفاهیم به صورت پاورپوینت آموزش داده می شود و سپس این مفاهیم در محیط عملی با مثال های متنوعی بررسی می شوند. از نقاط قوت این آموزش، تدریس مطالب شیء‌گرایی با ساده ترین روش و با ذکر مثال های بسیار است.

مدرس: مهندس فرشید شیرافکن
مدت زمان: ۸ ساعت

جهت مشاهده آموزش ویدئویی این آموزش [کلیک](#) کنید.

faradars.org/fvcp95042

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 تألیفی توسط مؤلف کتاب [کلیک](#) کنید.

Email: fshirafkan51@gmail.com

شماره همراه مدرس: ۰۹۱۲۱۹۷۲۰۲۸

کتاب رایگان دیگر از این مجموعه آموزشی

۱. آموزش نظریه زبان ها و ماشین - [کلیک کنید \(+\)](#)
۲. آموزش برنامه‌نویسی سی پلاس پلاس - [کلیک کنید \(+\)](#)
۳. آموزش پایگاه داده ها - [کلیک کنید \(+\)](#)
۴. آموزش ساختمان داده ها - [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>



دسته بندی موضوعی آموزش های فرادرس، در ادامه آمده است:

 <p>مهندسی برق الکترونیک و رباتیک</p>	 <p>هوش مصنوعی</p>	 <p>آموزش‌های دانشگاهی و تخصصی</p>	 <p>برنامه‌نویسی</p>
<p>مهندسی برق الکترونیک و رباتیک - کلیک (+)</p>	<p>هوش مصنوعی - کلیک (+)</p>	<p>آموزش‌های دانشگاهی و تخصصی - کلیک (+)</p>	<p>برنامه‌نویسی - کلیک (+)</p>
 <p>مهارت‌های دانشگاهی</p>	 <p>مباحث مشترک</p>	 <p>دروس دانشگاهی</p>	 <p>مهندسی نرم‌افزار</p>
<p>مهارت‌های دانشگاهی - کلیک (+)</p>	<p>مباحث مشترک - کلیک (+)</p>	<p>دروس دانشگاهی - کلیک (+)</p>	<p>مهندسی نرم‌افزار - کلیک (+)</p>
 <p>آموزش‌های عمومی</p>	 <p>طراحی و توسعه وب</p>	 <p>نرم‌افزارهای عمومی</p>	 <p>نرم‌افزارهای تخصصی</p>
<p>آموزش‌های عمومی - کلیک (+)</p>	<p>طراحی و توسعه وب - کلیک (+)</p>	<p>نرم‌افزارهای عمومی - کلیک (+)</p>	<p>نرم‌افزارهای تخصصی - کلیک (+)</p>

فهرست مطالب

فصل ۱: کلاس‌ها و اشیا

فصل ۲: اعضای کلاس با ویژگی استاتیک

فصل ۳: وراثت

فصل ۴: پلی مورفیسم

فصل ۵: سربارگذاری عملگرها

فصل ۶: قالب

فصل ۷: فایل

فصل ۸: آزمون با حل

فصل ۱: کلاس‌ها و اشیا

کلاس مجموعه‌ای از متغیرها و توابع کار با آنها می‌باشد. اعضاء یک کلاس می‌توانند در یکی از طبقه بندی‌های زیر باشند:

قابل دسترس از داخل کلاس	Private
قابل دسترس از داخل و خارج کلاس	public
قابل دسترس از داخل کلاس و یا کلاسهای ارث برده شده از این کلاس و توابع دوست آنها	protected

تذکر: تا حد امکان به جای `protected` از `private` استفاده کنید. چون `protected` مفهوم بسته بندی را از بین می‌برد.

* در این مثال، یک کلاس به نام `c` با یک عضو خصوصی به نام `a` تعریف شده است:

```
class c{
    private:
        int a;
};
void main(){
    c x;
    cout<<x.a;
}
```

در ابتدای تابع `main`، یک متغیر به نام `x` از کلاس `c` تعریف شده است. به این متغیر، شیء می‌گویند. باید توجه داشت که توسط این متغیر نمی‌توان از خارج کلاس به متغیر خصوصی `a` دسترسی داشت. بنابراین دستور `x.a`، تولید خطا می‌کند. ■

تذکر: به طور پیش فرض همه عناصر کلاس، خصوصی هستند. بنابراین کلمه `private` در مثال بال را می‌توان قرار نداد.


*خروجی برنامه زیر کدام است؟

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

حل: خروجی ۱ می باشد. توجه کنید که به متغیر خصوصی کلاس یعنی x می توان از طریق تابع همان کلاس یعنی f دسترسی داشت.

تذکر: تابع f را می توان خارج از کلاس نیز به صورت زیر تعریف کرد:

```
class a{
private:
    int x;
public:
    int f();
};
int a::f() {
    x=1;
    cout<<x;
}
void main(){
    a ob;
    ob.f();
}
■
```

در هنگام تعریف کلاس نباید به متغیرها مقدار دهی اولیه کرد. بنابراین عبارت زیر نادرست است: 

```
class a{
private:
    int x=1;
public:
    int f();
};
```

* تعریف یک کلاس برای صف حاوی کاراکترها:

صف ساختمان داده ای است که عمل اضافه کردن به انتها و عمل حذف از ابتدای آن انجام می شود. صف به کمک یک آرایه و دو متغیر front و rear پیاده سازی می شود. front به عنصر ما قبل اول و rear به عنصر آخر صف اشاره دارد. در ابتدا که صف خالی است، هر دو متغیر front و rear صفر می باشند. با هر بار اضافه کردن، یک واحد به rear اضافه شده و با هر بار حذف، یک واحد به front اضافه می شود.

```
#include <iostream.h>
class q{
private:
    char  a[10];
    int   front , rear;
public:
    void  set(void);
    void  insert(char ch);
    char  remove(void);
};
void q ::set (void) {
    front=0;
    rear=0;
}
void q::insert(char ch)
{
    if (rear==9) { cout<<"full"; return ; }
    rear++;
    a[rear]=ch;
}
char q::remove(void)
{
    if(front==rear){ cout<<"empty"; return 0;}
    front++;
    return a[front];
}
main(){
    q  x ;
    x.set ( );
    x.insert('A');
    x.insert('B');
    cout<<x.remove( );
}

```

در برنامه بالا ابتدا دو کاراکتر A و B به صف اضافه شده و سپس یک کاراکتر حذف شده و در خروجی نمایش داده می شود. بنابراین خروجی A است، چون حذف از اول صف انجام می گیرد.

■

سازنده و مخرب

در بیشتر برنامه‌ها، یکسری عملیات باید در هنگام ساخته شدن یک شیء و یا هنگام از بین رفتن شیء انجام شود. این عملیات توسط توابعی به نام‌های سازنده (constructor) و مخرب (destructor) انجام می‌گیرد. سازنده، تابعی هم نام با کلاس است که در هنگام ساخته شدن یک شیء از کلاس به طور خودکار صدا زده می‌شود و مخرب تابعی هم نام با کلاس (البته با یک علامت ~ قبل از اسم آن) می‌باشد که هنگام از بین رفتن شیء به طور خودکار فراخوانی می‌شود.

توابع سازنده و مخرب در قسمت public تعریف می‌شوند.

*خروجی برنامه زیر چیست؟

```
class c{
    private:
        int x;
    public:
        c() {cout<<'a'; }
};
void main(){
    c t;
}
```

حل: خروجی برنامه a است. به محض ایجاد شیء t در داخل تابع main، تابع سازنده (تابع هم نام با اسم کلاس)، به طور خودکار صدا زده می‌شود. ■

*خروجی برنامه زیر چیست؟

```
class c{
    private:
        int x;
    public :
        c(int=3);
        int fun();
};
c::c(int n) { x=n; }
int c::fun() { x++; return x; }
void main(){
    c ob;
    cout<<ob.fun();
}
```

حل: خروجی برنامه 4 است. ابتدا شیء ای به نام Ob از کلاس c ساخته شده که باعث صدا زدن اتوماتیک تابع سازنده می‌شود. که این تابع مقدار 4 را در متغیر خصوصی x ذخیره می‌کند. ■

* خروجی کدام است؟

```

class c{
private: int x;
public:
    c(int=0);
    void print();
};
c::c(int a) { x=a; }
void c::print() { cout<<x; }
void main(){
    c ob1(1);
    c ob2(2);
    ob1.print();
    ob2.print();
}

```

حل: خروجی 12 می باشد. ابتدا شیء ای به نام ob1 از کلاس c ایجاد شده و سازنده صدا زده می شود و مقدار x برابر 1 خواهد شد. سپس شیء دیگری به نام ob2 از کلاس c ساخته شده و سازنده صدا زده می شود و مقدار x برای این شیء برابر 2 خواهد شد. در خط آخر برنامه، تابع print توسط هر دو شیء صدا زده شده و ابتدا مقدار 1 و سپس مقدار 2 چاپ خواهد شد. توجه کنید که هر شیء دارای اعضای مخصوص به خود می باشد. ■


* خروجی برنامه زیر چه می باشد؟

```

class c{
public:
    c() { cout<<'a'; }
    ~c() {cout<<'b'; }
};
void main(){
    c t;
    cout<<'c';
}

```

حل: خروجی برنامه acb می باشد. هنگامی که شیء t در تابع main ساخته می شود، تابع سازنده به طور خودکار صدا زده شده و کاراکتر a را نمایش می دهد. سپس کاراکتر c نمایش داده شده و در نهایت با اتمام برنامه، تابع مخرب به طور خودکار صدا زده شده و کاراکتر b را چاپ می کند. ■

یک کلاس می تواند بیش از یک تابع سازنده داشته باشد. 

* خروجی برنامه زیر چه می باشد؟

```

class test{
public:
    test() {cout<<'a';}
    test(int m) {cout<<m;}
};
main(){
    test x(1);
    test y;
}

```

حل: خروجی برنامه 1a می باشد. زمانی که شیء X ساخته می شود، تابع سازنده که دارای پارامتر ورودی است صدا زده می شود و زمانی که شیء Y ساخته می شود، تابع سازنده بدون پارامتر ورودی صدا زده می شود.

تابع سازنده یک شیء سراسری، فقط یکبار آنهم در هنگام شروع اجرای برنامه، فراخوانی می شود.

تابع سازنده یک شیء محلی، با هر اجرای دستور معرفی آنها، فراخوانی می شود.

یک شیء سراسری زمانی که برنامه تمام شود از بین می رود.

یک شیء محلی زمانی از بین می رود که اجرای برنامه از میدان دید آنها خارج شود.

*خروجی برنامه زیر چه می باشد؟


```
class c{
public:
    c() { cout<<'a'; }
    ~c() { cout<<'b'; }
};
void main(){
    c a,b;
    { c d; }
    cout<<'c';
}
}
```


حل: خروجی برنامه aaabcbbb می باشد. ابتدا شی های a,b ساخته شده و دوبار سازنده صدا زده شد و aa چاپ می شود. سپس شیء دیگری به نام d ساخته شده و سازنده صدا زده شده و a چاپ می شود. بعد از پایان حوزه دید d، مخرب صدا زده شده و یک b نمایش داده شده و بعد از آن یک c و در نهایت بعد از پایان برنامه به علت اینکه در ابتدای برنامه دو شیء تعریف شده بود، دو بار مخرب صدا زده شده و دو b نمایش داده می شود.

*خروجی برنامه زیر چه می باشد؟

```
class c{
public:
    c() { cout<<'a';}
    ~c() { cout<<'b';}
};
void main(){
    c ob1,ob2;
    {c ob3; }
    c ob4;
}
}
```

حل: خروجی aaababbb می باشد.

نمی‌توان آدرس توابع سازنده و مخرب را به دست آورد. 

تابع مخرب دارای پارامتر ورودی و یا خروجی نمی‌باشد. 



انتساب اشیاء به یکدیگر

می‌توان اشیاء با نوع‌های یکسان را به یکدیگر نسبت داد. با این عمل، داده‌های یک شیء در شیء دیگر کپی می‌شود. در مثال زیر ob1 و ob2 هر دو از نوع کلاس c می‌باشند که توسط دستور ob2=ob1، اعضای داده‌ای ob1 در ob2 کپی می‌شوند. خروجی برنامه 2 می‌باشد:

```
class c{
public: int x;
      void f() { cout<< ++x;}
};
void main(){
c ob1,ob2;
ob1.x=1;
ob2=ob1;
ob2.f();
}
```

آرایه‌ای از کلاس

می‌توان آرایه‌ای تعریف کرد که اعضای آن کلاس باشند. در مثال زیر آرایه x با ۳ خانه، آرایه‌ای از کلاس c می‌باشد:

```
class c{
int a;
public: c(int i) {a=i;}
      int f() {return a;}
};
int main(){
int i;
c x[3]={c(5),c(4),c(3) };
cout<<x[0].f() + x[2].f();
}
```

خروجی این برنامه 8 می‌باشد. در تابع main، ابتدا آرایه‌ای سه خانه‌ای به نام x از نوع کلاس c ساخته شده که محتویات متغیر a برای این خانه‌ها برابر ۵ و ۴ و ۳ می‌باشد. این برنامه محتویات خانه اول و آخر آرایه را با هم جمع می‌کند.



ارسال شیء به تابع

اشیاء را می‌توان به عنوان آرگومان به توابع ارسال کرد.

* خروجی برنامه زیر ۲ می‌باشد. در تابع main ابتدا شیء ای به نام x ساخته شده و در این زمان سازنده اجرا شده و عدد ۲ را در i ذخیره می‌کند. سپس در خط بعدی، شیء x به تابع k ارسال شده که منجر به صدا زدن تابع f می‌شود.

```
class c{
    int i; int j;
public: c(int n) { i=n;}
       int f() { j=i; return j; }
};
void k(c ob) { cout<<ob.f(); }
void main(){
    c x(2);
    k(x);
}
```

تابع سازنده هنگام ارسال شیء به تابع اجرا نمی‌شود. 

تابع مخرب هنگام از بین رفتن شیء در زمان پایان اجرای تابع، اجرا می‌شود. 

* خروجی چیست؟

```
class c{
    int i; int j;
public: c(int n) { i=n;}
       ~c()      { cout<<'b';}
       int f() { j=i; return j; }
};
void k(c ob) { cout<<ob.f(); }
void main(){
    c x(2);
    k(x);
    cout<<'a';
}
```

حل: خروجی 2bab می‌باشد. توجه کنید که تابع مخرب دو بار صدا زده شده است. (یک بار در پایان تابع k به علت از بین

رفتن شیء ob و یکبار در پایان اجرای برنامه اصلی به علت از بین رفتن شیء x)

تابع با خروجی از نوع شیء

خروجی یک تابع می‌تواند از نوع شیء باشد. در مثال زیر خروجی تابع k از نوع کلاس c می‌باشد:

```
class c{ public: int i;};
c k( ){
    c ob;
    ob.i=2;
    return ob;
}
void main(){
    c x;
    x=k();
    cout<<x.i;}
```

خروجی برنامه بالا، ۲ می‌باشد.

■

*خروجی برنامه زیر چه می‌باشد؟

```
class c{
    public: int i ;
           ~c() {cout<<'a';}
};
c k( ){
    c ob;
    ob.i=2;
    return ob;
}
void main(){
    c x;
    x=k();
    cout<<x.i;
}
```

حل: خروجی aa2a می‌باشد. توجه کنید که تابع مخرب ۳ مرتبه صدا زده شده است.

■

توابع دوست (friend function)

تابعی که عضو کلاس نباشد، اما بتواند به اعضای خصوصی کلاس دسترسی داشته باشد را تابع دوست می‌نامیم. در مثال زیر تابع f که عضو کلاس c نمی‌باشد، به عضو خصوصی کلاس یعنی a می‌تواند دسترسی داشته باشد:

```
class c{
    int a;
public:
    c(int x) { a=x;}
    friend int f(c);
};
int f(c z)
{ cout<<z.a; }
main(){
    c ob(3);
    f(ob);
}
```

تذکر: توجه کنید که باید پیش‌الگوی تابع f را در کلاس c قرار دهیم.

تذکر: دستوری مانند $ob.f()$ غلط است، چون تابع f عضوی از کلاس c نمی‌باشد.


تذکر: دستور $cout<<a$ در تابع f نادرست است، چون تابع f فقط از طریق یک شیء می‌تواند به عضو خصوصی کلاس دسترسی داشته باشد.

*خروجی برنامه زیر چیست؟

```
class c{
    int x;
public:
    c(int i) {x=i;}
    friend f(c z) { cout<<z.x; }
    k() { cout<<x;}
};
void main(){
    c ob(2);
    f(ob);
    ob.k();
}
```

حل: خروجی برنامه 22 است. توجه کنید که برای صدا زدن تابع دوست f در تابع $main$ ، نباید از دستور $ob.f()$ استفاده شود، چون تابع f عضوی از کلاس نمی‌باشد ولی تابع k را به صورت $ob.k()$ صدا می‌زنیم چون تابع k عضو کلاس است.



تابع دوست می تواند دوست دو کلاس باشد. 

*خروجی چه می باشد؟


```
class d;
class c{
    int i;
public:
    c(int x){i=x;}
    friend int f(c ob1,d ob2);};
class d{
    int i;
public:
    d(int y) {i=y;}
    friend int f(c ob1,d ob2);
};
int f(c ob1,d ob2)
{
    return ob1.i+ob2.i;
}
main()
{
    c m(3);
    d n(4);
    cout<<f(m,n);
}
```

حل: خروجی برنامه ۷ می باشد. توسط تابع دوست می توان به عضو مشترک بین دو کلاس دسترسی داشت. در مثال بالا متغیر i در هر دو کلاس وجود دارد و توسط تابع دوست f به این متغیرهای مشترک دسترسی شده است.

تذکر: در تابع f موجود در کلاس c، نامی از d برده شده در حالی که کلاس d بعدا معرفی شده است. بنابراین قبل از تعریف کلاس c باید خط زیر را نوشت. به این عمل تعریف پیشاپیش (forward declaration) می گویند:

```
class d;
```

■

تابع دوست یک کلاس می تواند عضو کلاس دیگری باشد. 

کلاس دوست

کلاسی که دوست کلاس دیگری می باشد، می تواند به اعضای خصوصی آن دسترسی داشته باشد. در مثال زیر کلاس d دوست کلاس c است، بنابراین می تواند به متغیر i دسترسی داشته باشد:

```
class c{
    int i;
public:
    c(int x) { i=x;}
    friend class d;
};
class d{
public:
    int f(c z) {cout<<z.i;}
};
main()
{
    c ob1(3);
    d ob2;
    ob2.f(ob1);
}
```

خروجی برنامه بالا، ۳ می باشد.

■

توابع inline

توابعی که به صورت inline تعریف می شوند، فراخوانی نمی شوند و کد آنها در مکانی که صدا زده شده، کپی می شود. این عمل باعث افزایش سرعت اجرای برنامه می شود (به علت عدم نیاز به انتقال اطلاعات توسط پشته)، ولی حجم برنامه زیاد می شود.

*خروجی برنامه زیر ۲ می باشد و گذاشتن کلمه inline تاثیری در خروجی ندارد:

```
inline int f(int x) {x=x*2; return x; }
void main(){
    cout<<f(1);
}
```

■

وقتی تابعی در داخل اعلان کلاس ظاهر شود، به طور خودکار به یک تابع inline تبدیل می شود. در این صورت نیازی به کلمه کلیدی inline نیست.


فصل دوم


اعضای کلاس با ویژگی استاتیک


هر یک از اعضای کلاس (اعضای داده ای و یا توابع عضو)، را می توان به صورت static تعریف کرد.

اعضای داده ای استاتیک

وقتی یک متغیر کلاس را static معرفی می کنید، به کامپایلر می گوئید که فقط یک کپی از آن وجود دارد و تمام اشیای آن کلاس، آن متغیر را به اشتراک می گذارند.

مقدار اولیه متغیر عضو استاتیک برابر صفر می باشد. 

مقدار دهی اولیه متغیر عضو استاتیک نمی تواند در داخل کلاس انجام شود. 

وقتی یک عضو داده ای static را در داخل کلاسی اعلان کنید، حافظه به آن اختصاص نمی یابد و باید در خارج از کلاس به صورت عمومی تعریف شود. این کار به کمک عملگر تعیین حوزه (::)، برای شناسایی کلاسی که متغیر به آن تعلق دارد، انجام می شود. 

*خروجی چیست؟

```
class c{
    int a; static int b;
public:
    f(int x, int y) { a=x;b=y;}
    void g() { cout<<a<<b;}
};
int c::b;
main(){
    c ob1,ob2;
    ob1.f(1,2);
    ob2.f(3,4);
    ob1.g();
    ob2.g();
}
```

حل: خروجی 1434 می باشد. ابتدا دو شیء از کلاس c تعریف شده و تابع f توسط این شیء صدا زده شده و مقدار a برابر 1 و مقدار b برابر 2 می شود. سپس تابع f توسط شیء ob2 صدا زده شده و مقدار a برابر 3 و مقدار b برابر 4 خواهد شد. سپس با فراخوانی تابع g، مقدار a,b مربوط به شیء ob1 و سپس ob2 چاپ می شود. دقت کنید که چون متغیر b از نوع استاتیک است، مقدارش برابر آخرین مقدار ذخیره شده در آن یعنی 4 می باشد ولی مقدار a برای هر شیء متفاوت است. به عبارتی برای ob1 و ob2 که دو شیء از کلاس c می باشند، یک b مشترک وجود دارد. (تذکر: اگر static قبل از b را برداریم، خروجی 1234 خواهد بود.)

■

*خروجی چیست؟

```
class c{
public: static int x;
       c() {x++;}
};
int c::x;
main() {
    c m,n,p; cout<<c::x;
    c q; cout<<c::x;
}
```

حل: خروجی ۳۴ می باشد. ابتدا ۳ شیء m,n,p ساخته شده و سه مرتبه سازنده صدا زده می شود و مقدار x برابر ۳ شده و در خط بعد چاپ می شود. سپس شیء دیگری به نام q ساخته شده و سازنده صدا زده شده و به مقدار x قبلی یک واحد اضافه شده و برابر ۴ می شود.

تذکر: از آنجا که قبل از ایجاد شیء‌ای از کلاس، متغیر عضو static وجود دارد، در داخل برنامه می توان مستقل از هر نوع شیئی به آن دسترسی داشت. دستور cout<<c::x، در مثال قبل معرف همین موضوع است.

تذکر: از آنجا که متغیرهای سراسری، با اصل کپسوله سازی در تضاد می باشند، متغیرهای عضو استاتیک معرفی شدند.

توابع عضو استاتیک

توابع عضو را نیز می توان به صورت استاتیک معرفی کرد. این توابع فقط مستقیماً می توانند به اعضای استاتیک کلاس مراجعه کنند. از این توابع، می توان برای مقداردهی اولیه داده های استاتیک خصوصی، قبل از ایجاد شیء استفاده کرد. (تذکر: تابع عضو استاتیک می تواند به اعضای عمومی کلاس مراجعه کند.)

*خروجی چیست؟

```
class c{
    static int s;
public: static int f();
};
int c::s=3;
int c::f(){ cout<<s; }
int main(){
    c x; x.f();
    c::f();
}
```

حل: خروجی 33 می باشد. این برنامه، دو نحوه فراخوانی تابع static را نشان می دهد:

۱- به کمک یک شیء (دستور: x.f())

۲- با استفاده از نام کلاس و عملگر تعیین حوزه (دستور: c::f())

تابع عضو استاتیک می تواند به داده استاتیک خصوصی، قبل از این که شیء ایجاد شود، مقدار اولیه دهد.

* خروجی برنامه، 3 می باشد:

```
class c{
    static int s;
public: static int f(int x) {s = x; }
        k() {cout<<s;}
};
int c::s;
int main(){
    c::f(3);
    c ob;
    ob.k();
}
```

تذکر: شاگرد هر یک از عبارات static در برنامه بالا را حذف کنیم، کامپایلر خطا می گیرد.

■

* خروجی چه می باشد؟

```
class c{
    static int a;
    int b;
public: static f(){ a=1; b=2; }
};
int c::a=3;
main(){
    c ob;
    ob.f();
}
```

حل: برنامه خطا دارد. نمی توان به عضو b در داخل تابع f بدون استفاده از شیء دسترسی داشت. (تابع عضو استاتیک فقط می تواند مستقیماً به متغیر استاتیک دسترسی داشته باشد).
تذکر: اگر b از اعضای عمومی بود، خطا گرفته می شد.

■

اعضای کلاس با ویژگی ثابت

هر یک از اعضای داده ای و یا توابع عضو یک کلاس را می توان به صورت `const` تعریف کرد.

*خروجی چیست؟

```
class c{
    int a;
public: c(int b=1): a(b) { }
        int f() const {return ++a;}
};
main(){
    c ob; cout<<ob.f();
}
```

حل: خطا دارد. تابع `f` نمی تواند مقدار `a` را تغییر دهد. اگر `const` بعد از تابع `f` را برداریم، خروجی 2 خواهد بود.

■ تابع عضو کلاس را می توان `const` تعریف کرد. در این صورت، تابع نمی تواند شیء ای را که موجب فراخوانی آن شده است را تغییر دهد.

شیء ثابت نمی تواند تابع عضو غیر ثابت را فراخوانی کند.

*خروجی چیست؟

```
class c{
    int x;
    const int y;
public: c( int a, int b) : x(a),y(b) { };
        f1() { x=y+1;}
        f2() const {cout << x<<y;}
};
int main(){
    c ob(1,2);
    ob.f1();
    ob.f2();}
```

حل: خروجی 32 است.

تذکر: استفاده از دستوری مانند $y=x+1$ در تابع `f1` و یا `f2` نادرست است، چون مقدار ثابت `y` را نمی توان تغییر داد.

*خروجی چیست؟

```
class c{
public: int x;
        c() { cout<<'a'; }
};
main(){
    const c ob;
    ob.x=1;
}
```

حل: برنامه خطا دارد. اگر `const` نباشد، خروجی `a` می باشد.

تابع عضو استاتیک نمی تواند حاوی اشاره گر `this` باشد.

تابع عضو استاتیک نمی تواند مجازی باشد.

تابع عضو استاتیک نمی تواند به صورت `const` و `volatile` تعریف شود.

اشاره گر به اشیاء

می توان یک اشاره گر به یک شیء تعریف کرد. برای دسترسی به اعضای شیء توسط اشاره گر، از علامت `->` (منها و علامت بزرگتر) ، باید استفاده کرد. در مثال زیر متغیر `p` اشاره گری از این نوع است:

```
class c{
    int x;
public: c(int y) { x=y;}
       int f() { return x;}
};
int main(){
    c a(2);
    c *p;
    p=&a;
    cout << p->f();
}
```

در ابتدا شیء `a` از کلاس `c` تعریف شده و مقدار اولیه `x` توسط تابع سازنده برابر ۲ می شود. سپس اشاره گر `p` از نوع کلاس `c` تعریف شده و آدرس شیء `a` در آن قرار می گیرد. در نهایت تابع `f` که عضوی از `p` است، اجرا شده و مقدار ۲ در خروجی چاپ می گردد.

* خروجی برنامه زیر چیست؟

```
class c{
    int x;
public: c(int y) { x=y;}
       int f() { return x;}
};
int main(){
    c a[2]={5,3};
    c *p;
    p=a;
    p++;
    cout << p->f();
}
```

حل: خروجی این برنامه ۳ است. ابتدا یک آرایه از کلاس `c` ساخته شده و سپس یک اشاره گر به کلاس `c` به نام `p` تعریف می شود که با دستور `p=a`، اشاره گر به خانه اول آرایه اشاره می کند و بعد از اجرای دستور `p++`، به خانه دوم آرایه اشاره می کند و در نهایت مقدار ۳ چاپ خواهد شد.

*خروجی برنامه زیر چیست؟

```

class c{
public:
    int x;
    void f() { cout<<'a';}
};
void main(){
    c ob,*p;
    p=&ob;
    ob.f();
    p->f();
    (*p).f();
}

```

حل: خروجی aaa می باشد. هر سه دستور آخر معادل هم می باشند. (برای دسترسی به تابع f، از سه روش استفاده شده است.)

■

*خروجی برنامه زیر 1 می باشد. متغیر p در این مثال، اشاره گر به x است:

```

class c{
public: int x;
    c(int a) {x=a;}
};
int main(){
    c ob(1);
    int c::*p;
    p=&c::x;
    cout<<ob.*p;
}

```

■

*خروجی برنامه زیر ۲ می باشد. متغیر p در این مثال، اشاره گر به تابع f می باشد:

```

class c{
public: int x;
    c(int a) {x=a;}
    int f() { return ++x;}
};
int main(){
    c ob(1); int (c::*p)();
    p = &c::f;
    cout<<(ob.*p)();
}

```

■

دستیابی به عضو عمومی شیء از طریق اشاره گر

می توان آدرس عضو عمومی شیء را به اشاره گری نسبت داد و از طریق آن اشاره گر به آن دست یافت. به مثال زیر که خروجی آن 3 می باشد، توجه کنید:

```
class c{ public: int a; };  
int main(){  
    c ob; int *p;  
    p=&ob.a;  
    *p=3;  
    cout<<*p;  
}
```



ارسال مرجع به اشیاء

وقتی شیئی به عنوان آرگومان به تابعی ارسال می‌شود، یک کپی از آن ایجاد می‌شود و در زمان پایان اجرای تابع، مخرب شیء کپی فراخوانی می‌شود. اگر نخواهید مخرب اجرا شود، می‌توانید شیء را از طریق مرجع ارسال کنید تا دیگر کپی از آن ایجاد نشود.

* خروجی برنامه زیر 1a می‌باشد:

```
class c{
    int a;
public: int b;
    ~c() {cout<<'a';}
    void f(c &x) { --x.b; }
};
int main(){
    c ob; ob.b=2; ob.f(ob); cout<<ob.b;
}
```

تذکر: اگر & قبل از x در تابع f را برداریم، خروجی a2a می‌باشد.

* خروجی برنامه زیر چیست؟

```
class c{
    private: int x;
public: c(): x(1) { }
    f() {cout<<x;}
    friend k(c &,int);
};
k(c &a,int val) { a.x = val;}
int main(){
    c ob;
    k(ob,2);
    ob.f();
}
```

حل: خروجی ۲ است. ■

اشاره گر this

وقتی تابع عضوی فراخوانی می‌شود، یک آرگومان ضمنی به طور خودکار به آن ارسال می‌شود که اشاره‌گری به شیء فراخوانی‌کننده است. این اشاره گر، `this` نام دارد. در واقع هر شیء به کمک اشاره گر `this` به آدرس خود دسترسی پیدا می‌کند.

تذکر: اشاره گر `this`، جزئی از خود شیء نمی‌باشد.

توابع عضو استاتیک، اشاره گر `this` ندارند، چون مستقل از هر شیئی از کلاس وجود دارند.

* خروجی چیست؟

```
class c {
private:
    int x;
public:
    c( int a ) {x=a; }
    void f( ) { cout<<x << this->x; }
};
int main(){
    c ob(2);
    ob.f();
}
```

حل: خروجی 22 می‌باشد. وقتی تابع `f` در دستور `ob.f()` فراخوانی می‌شود، به طور خودکار اشاره گر `this` به تابع `f` ارسال می‌شود که به شیء `ob` اشاره خواهد کرد.

تذکر: دستور `cout<<x` با دستور `cout<<this->x` معادل است.

تذکر: می‌توان به جای دستور `this->x` از دستور `x (*this)` نیز استفاده کرد.

■

دستورات new, delete

توسط new و delete در C++ می‌توان عمل تخصیص فضا از حافظه Heap و آزاد سازی را انجام داد. (به جای malloc و free در زبان C). کار با دستور new ساده‌تر از malloc در زبان C می‌باشد. در این دستور نیازی به استفاده از sizeof برای محاسبه اندازه شی نمی‌باشد و نیازی به قالب بندی خروجی نمی‌باشد و می‌توان مقدار دهی اولیه نیز انجام داد. * ایجاد یک متغیر با مقدار اولیه ۲ در heap و در نهایت از بین بردن آن:

```
int *a;
a= new int(2);
delete a;
```

■

* برای تعریف یک آرایه با ۳ خانه در heap، از دستور زیر استفاده می‌کنیم:

```
int *x;
x=new int[3];
```

■

و برای حذف آرایه، از دستور delete [] x; یا delete [3] x استفاده می‌کنیم.

* پیاده سازی یک پشته به کمک دستور new :

```
class stack{
    private:
        int size,top,*a;
    public:
        stack(int size=10):top(-1) {a=new int[size]; }
        ~stack(){delete [ ] a;}
        void push(const int& item) {a[++top]=item;}
        void pop(int& item) {item=a[top--]; }
};
void main()
{
    stack s;
    int x=4,y,z;
    s.push(x);
    s.push(x+1);
    s.pop(y);
    s.push(x+5);
    s.push(x-2);
    s.pop(z);
    s.pop(z);
    cout<<y+z;
}
```

در ابتدا با ایجاد شیء s، سازنده فراخوانی شده و آرایه ای ۱۰ خانه ای از heap گرفته می‌شود و top برابر ۱- می‌شود. سپس مقادیر ۵ و ۴ در پشته push شده و سپس عمل pop انجام شده و عدد ۵ در y قرار می‌گیرد. بعد از آن دو مقدار ۹ و ۲ در پشته push شده و در نهایت دو بار عمل pop انجام شده و عدد ۹ در z قرار خواهد گرفت. بنابراین خروجی یعنی مجموع مقادیر y و z برابر ۱۴ چاپ خواهد شد. در پایان برنامه، مخرب صدا زده شده و آرایه ایجاد شده از بین می‌رود.

فصل ۳: وراثت

یکی از خصوصیات زبانهای شیء‌گرا، ارث بری می باشد. فرض کنید کلاسی به نام shape داریم که شامل ویژگی های مشترک همه اشکال می باشد. حال می توان از این کلاس، کلاسهایی مانند دایره، مثلث و یا مستطیل را به ارث برد. یا فرض کنید که کلاسی برای اتومبیل تهیه کرده اید که شامل خصوصیات مشترک اتومبیل ها مانند تعداد چرخ و تعداد سرنشین ها باشد. حال می توان کلاسی برای کامیون تعریف کرد به طوری که از کلاس اتومبیل ارث بری کند و شامل خصوصیات دیگری مانند مقدار بار قابل حمل نیز باشد. یک کلاس می تواند به یکی از ۳ حالت زیر از یک کلاس پایه، مشتق شود:

تمام اعضای عمومی کلاس پایه، اعضای عمومی کلاس مشتق خواهند بود و تمام اعضای محافظت شده کلاس پایه نیز به عنوان اعضای محافظت شده کلاس مشتق منظور می شوند.	public
تمام اعضای عمومی و محافظت شده کلاس پایه، به عنوان اعضای اختصاصی کلاس مشتق منظور می شوند.	private
تمام اعضای عمومی و محافظت شده کلاس پایه به عنوان اعضای محافظت شده کلاس مشتق منظور می شود.	protected

* خروجی برنامه زیر ۱۲ است: (کلاس b از کلاس a به صورت public ارث برده است)

```
class a{
    public:
        int f(int n) { return n;}
};
class b : public a{
    public:
        int g(int n) { return n;}
};
void main() {
    b x ;
    cout<< x.f(1);
    cout<< x.g(2);
}
```

در این مثال چون ارث بری public می باشد، عضو عمومی کلاس a یعنی تابع f، عضو عمومی کلاس b خواهد بود. بنابراین شیء x که از نوع کلاس b می باشد، می تواند به تابع f نیز دسترسی داشته باشد. تذکر: اگر ارث بری از نوع protected و یا private بود، در خط (x.f(1) خطا ایجاد می شد.



* خروجی برنامه زیر چیست؟

```

class a{
    public:
        f( ) {cout<<'a';}
};
class b: public a{
    public:
        g( ) {cout<<'b'; }
};
class c : public b{
    public:
        h( ) {cout<<'c'; }
};
main() {
    c x; x.f(); x.g(); x.h();
    b y; y.f(); y.g();
}

```

حل: خروجی abcab می باشد. کلاس b از کلاس a ارث برده و کلاس c از کلاس b. بنابراین شیء x که از نوع کلاس c تعریف شده به تمامی توابع f,g,h دسترسی دارد و شیء y که از کلاس b به ارث برده به توابع f,g دسترسی دارد.

تذکر: دستوری مانند () y.h نادرست است. چون از طریق شیء y فقط می توان به اعضای عمومی کلاسهای a و b دسترسی داشت و نمی توان به اعضای عمومی کلاس c دسترسی داشت.

■

* خروجی چیست؟

```

class a{
    public:
        a() { cout<<'1'; }
        ~a() { cout<<'2'; }
};
class b : public a{
    public:
        b() { cout<<'3'; }
        ~b() { cout<<'4'; }
};
main(){
    b x;
}

```

حل: خروجی 1342 می باشد. وقتی شیء x از کلاس b ساخته می شود، ابتدا سازنده کلاس a و سپس سازنده کلاس b صدا زده می شوند و ۱۳ به نمایش در می آید. هنگامی که برنامه تمام می شود، ابتدا مخرب کلاس b و سپس مخرب کلاس a صدا زده می شوند.

■

*خروجی چیست؟

```

class X{
public:
    X() { cout<<"1";}
    ~X() { cout<<"6"; }
};
class Y : public X{
public :
    Y() { cout<<"2";}
    ~Y() { cout<<"5";}
};
class Z : public Y{
public:
    Z() { cout<<"3";}
    ~Z() { cout<<"4";}
};
void main(){
    Z ob;
}

```

حل: خروجی 123456 می باشد.

تذکر: اگر شیء ob از کلاس Y باشد، خروجی 1256 می باشد.

■

*خروجی چیست؟

```

class c1{
    protected: int i;
    public: c1(int x) {i=x; cout<<i+1;}
};
class c2: public c1{
    public: c2(int x) : c1(x) {cout<<i -1; }
};
main(){
    c2 a(3);
}

```

حل: خروجی ۴۲ است. وقتی شیء a ایجاد می شود، ابتدا سازنده کلاس c1 صدا زده شده و ۴ چاپ می شود. سپس سازنده کلاس c2 صدا زده شده و ۲ چاپ می شود.

تذکر: به تعریف تابع سازنده c2 توجه کنید.

تذکر: اگر شیء a از کلاس c1 تعریف شود، خروجی ۴ می باشد.

*خروجی چیست؟

```

class c1{
    public: c1(int x) {cout<<x;}
};
class c2: public c1{
    public: c2(int x) : c1(++x) {cout<<x; }
};
main(){
    c1 ob1(3);
    c2 ob2(3);
}

```

حل: خروجی 344 می باشد. وقتی شیء ob1 ساخته می شود، سازنده کلاس c1 فراخوانی شده و عدد ۳ چاپ می شود. سپس با ایجاد شیء ob2، سازنده کلاس c1 و سپس سازنده کلاس c2 فراخوانی شده و 44 چاپ می شود.

■

* خروجی چیست؟

```

class X{
    public : X(){ cout<<'a';}
            ~X() {cout<<'b';}
};
class Y: public X{
    public: Y () {cout<<'c';}
            ~Y() {cout<<'d';}
};
void main() {
    X x;
    Y y;
}

```

حل: خروجی aacdbb می باشد. ابتدا شیء ای از کلاس X ساخته شده و سازنده آن صدا زده شده و a چاپ می شود. سپس شیء ای از کلاس Y ساخته شده و سازنده کلاس X و سپس سازنده کلاس Y صدا زده شده و ac چاپ می شود. بعد از بین رفتن شیء y، تابع مخرب کلاس Y و سپس مخرب کلاس X صدا زده شده و db چاپ می شود. در نهایت هم با اتمام برنامه، مخرب کلاس X صدا زده شده و b چاپ می شود.

■

* خروجی چیست؟

```

class X{ public : int i; };
class Y : public X{ public: int i; };
void main(){
    X x1,x2;Y y1;
    x1.i=1;
    y1.X::i=2;
    y1.i=3;
    x2=y1;
    cout<<x2.i;
}

```

حل: خروجی ۲ می باشد.

خط ۱: $x1.i=1$ ، باعث ذخیره شدن مقدار ۱ در عضو i مربوط به شیء $x1$ از کلاس X می شود.
 خط ۲: $y1.X::i=2$ ، باعث ذخیره شدن مقدار ۲ در عضو i ارث برده از کلاس X مربوط به شیء $y1$ می شود.
 خط ۳: $y1.i=3$ ، باعث ذخیره شدن مقدار ۳ در عضو i مربوط به شیء $y1$ از کلاس Y می شود.
 خط ۴: $x2=y1$ ، باعث ذخیره شدن i مربوط به شیء $y1$ با مقدار ۲ در i مربوط به شیء $x2$ می شود.
 تذکر: در این مثال چند متغیر i داشتیم:


۱- یکی مربوط به شیء $x1$ با مقدار ۱
 ۲- دو تا مربوط به شیء $y1$ با مقادیر ۲ و ۳


کلاس‌های دارای چند مبنا (وراثت چندگانه)

یک کلاس می‌تواند مستقیماً چند کلاس مبنا را به ارث ببرد. در مثال زیر، کلاس c3 از دو کلاس c1, c2 ارث برده است. خروجی این برنامه abc12 است:

```
class c1{
    int a;
public :
    c1(int x) { a=x;cout<<'a';}
    int f() {cout<<a;}
};
class c2{
    int b;
public :
    c2(int x) { b=x;cout<<'b'; }
    int g() { cout<<b; }
};
class c3 : public c1, public c2{
public :
    c3 ( int m , int n) : c1(m),c2(n) { cout<<'c';}
    void fun () { f () ; g () ; }
};
main() {
    c3 t(1,2);
    t.fun();}
```

تذکر: اگر در کلاس c3 به جای public از private و یا protected استفاده می‌شد، خروجی فرقی نمی‌کرد.

کلاسهای istream, ostream با وراثت یگانه از کلاس پایه ios مشتق شده‌اند. 

کلاس ostream با وراثت چندگانه از هر دو کلاس ostream و istream مشتق شده‌اند. 

فصل ۴:

پلی مورفیسم (چند شکلی)

استفاده از یک نام در چندین مورد مربوط به هم و یا به عبارتی استفاده از یک رابط برای یک دسته کلی از اعمال را پلی مورفیسم می‌نامند. در مثال زیر دو تابع به نام‌های یکسان fun وجود دارد که کامپایلر با توجه به ورودی آنها، می‌تواند تشخیص دهد که کدام تابع را باید اجرا کند. خروجی این برنامه 6B می‌باشد:

```
int fun(int a) { return ++a; }
char fun(char b) { return ++b; }
void main(){
    cout<<fun(5);
    cout<<fun('A');
}
```

* خروجی چیست؟


```
int f(int a) { return ++a; }
char f(int b) { return ++b; }
void main(){
    cout<<f(1);
    cout<<f(2);
}
```

حل: برنامه خطا دارد. کامپایلر نمی‌تواند تابع‌ها را از هم تشخیص دهد. اگر نوع متغیر b برابر char یا float بود، خروجی 266 بود.

■

تابع مجازی

تابعی که در یک کلاس پایه اعلان شده و مجدداً در کلاس مشتق تعریف می‌گردد را تابع مجازی می‌گویند. مثلاً همه اشیاء دایره، مثلث و مستطیل که از کلاس شکل مشتق می‌شوند، دارای تابعی مثلاً به نام `draw` می‌باشند که شکل را رسم می‌کند و برای هر یک از اشیاء متفاوت است. حال برای اینکه برنامه در زمان اجرا خودش تشخیص دهد که کدام تابع مد نظر است، تابع را در کلاس پایه به صورت **virtual** تعریف می‌کنیم.

تابع مجازی موجب پشتیبانی چندریختی در زمان اجرا می‌شود. 

*خروجی برنامه زیر چیست؟

```
class c1{
public: virtual void f(){ cout<<'1'; }
};
class c2 : public c1{
public: void f() {cout<<'2'; }
};
class c3 : public c1{
public: void f() {cout<<'3'; }
};
main(){
c1 *p, ob1; c2 ob2; c3 ob3;
p=&ob1;
p->f();
p=&ob2;
p->f();
p=&ob3;
p->f();}
```

حل: خروجی 123 است. اگر کلمه `virtual` را برداریم، خروجی 111 خواهد بود.

تذکر: اگر `p` به کلاس `c1` یا `c2` اشاره کند، برنامه بالا دارای خطا خواهد بود. به عبارتی فقط با اشاره گر به کلاس پایه، می‌توان به کلاسهای مشتق نیز دسترسی داشت.

■

* خروجی برنامه زیر چیست؟

```

class c1{
public:
    virtual void f(){ cout<<'1'; }
};
class c2 : public c1{
public:
    void f() {cout<<'2'; }
};
main(){
    c1 ob1,*p1;
    c2 ob2,*p2;
    p1=&ob1;
    p1->f();
    p2=&ob2;
    p2->f();
}

```

حل: خروجی 12 است. اگر کلمه virtual را برداریم، خروجی باز هم 12 خواهد بود. چون هر تابع f() را به کمک اشاره گر مربوط به آن کلاس صدا زده ایم.

■


* خروجی برنامه زیر چیست؟

```

class c1{ public: virtual void f(){ cout<<'1'; } };
class c2 : public c1{ public: void f() {cout<<'2'; } };
class c3 : public c1{ public: void f() {cout<<'3'; } };
main(){
    c1 ob1;
    c2 ob2;
    c3 ob3;
    ob1.f();
    ob2.f();
    ob3.f();}

```

حل: خروجی 123 است. اگر کلمه virtual را برداریم، باز هم خروجی 123 خواهد بود. چون به هر تابع با متغیر از نوع کلاس مربوطه دسترسی داشته ایم.

صفت مجازی موروثی است. یعنی اگر کلاس c3 از کلاس c2 در مثال بالا مشتق شود، باز هم خروجی فرقی نمی کند. 

فراخوانی تابع مجازی از طریق مرجع کلاس پایه

می‌توان یک تابع مجازی را از طریق مرجع کلاس پایه صدا زد. در مثال زیر تابع مجازی $f()$ ، توسط مرجع p از کلاس پایه صدا زده شده است. خروجی برنامه 12 است:

```
class c1{ public: virtual void f(){ cout<<'1'; } };
class c2 : public c1{ public: void f( ) {cout<<'2'; } };
void k(c1 &p) {
    p.f();
}
main(){
    c1 ob1;
    c2 ob2;
    k(ob1);
    k(ob2);
}
```

تذکر: اگر متغیر مرجع p در تعریف تابع k از کلاس $c2$ باشد، از خط $k(ob1)$ در برنامه، خطا گرفته می‌شود.

تابع مجازی محض

تابع مجازی محض، نوعی تابع مجازی است که در کلاس پایه تعریف نشده باشد. در این حالت هر کلاس مشتق باید تعریف خاص خودش را ارائه کند.

*تابع f در برنامه زیر، یک تابع مجازی محض است. این تابع در کلاس پایه تعریف نشده و کلاسهای $cdec$ و $chex$ که از این کلاس مشتق شده‌اند، تعریف خاص خودشان از این تابع را دارند.

```
class number{
    protected: int x;
    public: void set(int i){ x=i;}
    virtual void f()=0;
};
class cdec : public number{ public: void f() {cout<<x; } };
class chex : public number{ public: void f() {cout<<hex<<x; } };
main(){
    cdec d; chex h;
    d.set(5);
    d.f();
    h.set(10);
    h.f();
}
```

خروجی برنامه 5A می‌باشد. (متغیر x از نوع $protected$ اعلان شده تا بتوان از آن در کلاسهای مشتق استفاده کرد.)

* خروجی چیست؟

```

class X{      public : void f() {cout<<"a";}};
class Y : public X{public : void f(){ cout<<"b";}};
void main(){
    X x; Y y;
    x.f();
    y.X::f();
    y.f();
    y.Y::f();}

```

حل: خروجی aabb می باشد. (تذکر: دستوری مثل x.Y::f() نادرست است).

■

کلاس انتزاعی

کلاسی که حداقل دارای یک تابع مجازی محض باشد را کلاس انتزاعی می گویند. هیچ شیئی از این نوع کلاس نمی توان ایجاد کرد. البته می توان اشاره گر و مرجع به این کلاس ایجاد کرد.

* خروجی برنامه زیر چیست؟

```

class c1{ public: virtual void f()=0; };
class c2 : public c1{ public: void f() {cout<<'a'; } };
main(){
    c1 ob1; c2 ob2;
    ob2.f();
}

```

حل: کلاس c1 انتزاعی می باشد، بنابراین خط c1 ob1، که از کلاس مجازی، شیء ای تعریف شده، خطا دارد.

■

منتخبی از عناوین آموزشی منتشر شده بر روی فرادرس

برنامه نویسی	
مدت زمان تقریبی	عنوان آموزش
۲ ساعت	آموزش اصول و مبانی برنامه نویسی - کلیک کنید (+)
۲۳ ساعت	آموزش برنامه نویسی جاوا - کلیک کنید (+)
۲۹ ساعت	آموزش برنامه نویسی PHP - کلیک کنید (+)
۲۰ ساعت	آموزش برنامه نویسی ++C - کلیک کنید (+)
۸ ساعت	آموزش پیشرفته ++C (شیء گرایی در سی پلاس پلاس) - کلیک کنید (+)
۸ ساعت	آموزش مقدماتی برنامه نویسی سی شارپ (#C) - کلیک کنید (+)
۱۴ ساعت	مجموعه آموزش های کاربردی برنامه نویسی #C (سی شارپ) - کلیک کنید (+)
۱۴ ساعت	آموزش شیء گرایی در سی شارپ (#C) - کلیک کنید (+)
۱۸ ساعت	آموزش برنامه نویسی پایتون - مقدماتی - کلیک کنید (+)
۵ ساعت	آموزش تکمیلی برنامه نویسی پایتون - کلیک کنید (+)
۱۳ ساعت	آموزش برنامه نویسی C - کلیک کنید (+)
۱۱ ساعت	آموزش SQL Server - مقدماتی - کلیک کنید (+)
۶ ساعت	آموزش SQL Server - تکمیلی - کلیک کنید (+)
۳ ساعت	آموزش کار با دستورات پایگاه داده در SQL Server - کلیک کنید (+)
۹ ساعت	مجموعه آموزش های برنامه نویسی متلب (MATLAB) - کلیک کنید (+)
۷ ساعت	مجموعه آموزش های برنامه نویسی متلب پیشرفته - کلیک کنید (+)
۱۶ ساعت	آموزش برنامه نویسی اندروید (Android) - مقدماتی - کلیک کنید (+)
۲۰ ساعت	آموزش برنامه نویسی اندروید (Android) - تکمیلی - کلیک کنید (+)
۲۶ ساعت	آموزش برنامه نویسی اندروید (Android) - پیشرفته - کلیک کنید (+)
۳ ساعت	آموزش پروژه محور برنامه نویسی دلفی (Delphi) - کلیک کنید (+)
۴ ساعت	آموزش زبان برنامه نویسی فرترن (Fortran) - مقدماتی - کلیک کنید (+)
۲ ساعت	آموزش مروری زبان برنامه نویسی FORTRAN با مثال های کاربردی - کلیک کنید (+)
۱۰ ساعت	آموزش پروژه محور اینترنت اشیا (IoT) - کنترل لوازم منزل با شبکه و پیامک و ماژول های SIM و ESP8266 - کلیک کنید (+)
۵ ساعت	آموزش برنامه نویسی Swift (سوئیفت) برای برنامه نویسی iOS - کلیک کنید (+)
۶ ساعت	آموزش برنامه نویسی پاسکال (Pascal) - کلیک کنید (+)
۱۶ ساعت	آموزش برنامه نویسی برای دانش آموزان با اسمال بیسیک (Small Basic) - کلیک کنید (+)
۶ ساعت	آموزش برنامه نویسی تصویری به کودکان با زبان اسکرچ (Scratch) - کلیک کنید (+)
۱ ساعت	آموزش محافظت از کدهای نرم افزاری با SmartAssembly (اسمارت اسمبلی) - کلیک کنید (+)

برنامه نویسی (ادامه از صفحه قبل)	
مدت زمان تقریبی	عنوان آموزش
۱۰ ساعت	آموزش مبانی برنامه نویسی (الگوریتم و فلوچارت) با رویکرد حل مسأله - کلیک کنید (+)
۱۰ ساعت	آموزش پروژه محور Python (پایتون) - ساخت نرم افزار برای Windows و Linux - کلیک کنید (+)
۵ ساعت	آموزش برنامه نویسی آردوینو (Arduino) با پایتون (Python) - کلیک کنید (+)
۲ ساعت	آموزش ساخت اپلیکیشن اندروید (Android) با PHP و MySQL - کلیک کنید (+)
۳ ساعت	آموزش مقدماتی زبان برنامه نویسی کاتلین (Kotlin) برای توسعه اندروید (Android) - کلیک کنید (+)
۱ ساعت	آموزش داکر (Docker) برای توسعه دهندگان - کلیک کنید (+)
۱ ساعت	آموزش توسعه وب با زبان برنامه نویسی Go - کلیک کنید (+)
۱۳ ساعت	آموزش ویژوال بیسیک دات نت (Visual Basic.NET) - مقدماتی - کلیک کنید (+)
۱۶ ساعت	آموزش ویژوال بیسیک دات نت (Visual Basic.NET) - تکمیلی - کلیک کنید (+)
۱ ساعت	آموزش تاریخچه گیت (Git) و مقدمات آن - کلیک کنید (+)
۱ ساعت	آموزش گیت (Git) برای مدیریت نسخه توزیع شده - کلیک کنید (+)
۵ ساعت	آموزش زبان برنامه نویسی AWK در لینوکس - پردازش و آنالیز فایل های متنی - کلیک کنید (+)
۷ ساعت	آموزش برنامه نویسی R و نرم افزار RStudio - کلیک کنید (+)
۵ ساعت	آموزش تکمیلی برنامه نویسی R و نرم افزار RStudio - کلیک کنید (+)
۱ ساعت	آموزش مقدماتی ساخت ربات تلگرام با PHP - کلیک کنید (+)
۹ ساعت	آموزش کتابخانه قالب استاندارد (STL) در ++C برای ساده نویسی برنامه ها - کلیک کنید (+)
۴ ساعت	آموزش برنامه نویسی با روش سه لایه به زبان VB.Net - کلیک کنید (+)
۴ ساعت	آموزش بازی سازی در پایتون با کتابخانه Pygame - کلیک کنید (+)
۵ ساعت	آموزش پایتون گرافیکی (رابط های گرافیکی پایتون) - کلیک کنید (+)
۱ ساعت	آموزش ساخت ربات تلگرام با پایتون (Python) - کلیک کنید (+)
۷ ساعت	مجموعه آموزش های طراحی رابط های گرافیکی (GUI) در متلب
۱۴ ساعت	آموزش تکمیلی طراحی رابط گرافیکی کاربر (GUI) با متلب (MATLAB) - کلیک کنید (+)
۱ ساعت	آموزش ساخت ربات تلگرام با متلب (MATLAB) - کلیک کنید (+)
۱ ساعت	آموزش ساخت ربات تلگرام با جاوا (Java) - کلیک کنید (+)
۲ ساعت	آموزش پایگاه داده ها در جاوا - کلیک کنید (+)
۳ ساعت	آموزش کار با فایل ها در جاوا با پکیج Java/IO (به همراه پروژه دفترچه یادداشت) - کلیک کنید (+)
۱ ساعت	آموزش پروژه محور جاوا - تولید QR code و ایجاد و اسکن بارکد (Barcode) - کلیک کنید (+)

برنامه نویسی (ادامه از صفحه قبل)	
مدت زمان تقریبی	عنوان آموزش
۶ ساعت	آموزش پروژه محور جاوا (Java) - طراحی و ساخت شبکه اجتماعی - کلیک کنید (+)
۴ ساعت	آموزش طراحی رابط کاربری مدرن در .NET با کامپوننت های جانوس (Janus) - کلیک کنید (+)
۲ ساعت	آموزش ساخت بازی ساده در ویژوال بیسیک - کلیک کنید (+)
۱۰ ساعت	آموزش پیاده سازی سیستم انبارداری و فروش با ویژوال بیسیک دات نت (VB.NET) - مقدماتی - کلیک کنید (+)
۱۲ ساعت	آموزش پیاده سازی سیستم انبارداری و فروش با ویژوال بیسیک دات نت (VB.NET) - تکمیلی - کلیک کنید (+)
۴ ساعت	آموزش مقدماتی پیاده سازی مسائل بهینه سازی در پایتون (Python) - کلیک کنید (+)
۱۲ ساعت	آموزش الگوهای طراحی (Design Patterns) در پایتون (Python) - کلیک کنید (+)
۲ ساعت	آموزش رایگان نکات و ترفندهای متلب - کلیک کنید (+)
۱۴ ساعت	مجموعه آموزش های برنامه نویسی متلب برای علوم و مهندسی - کلیک کنید (+)
۶ ساعت	آموزش فریم ورک Spring در جاوا - کلیک کنید (+)
۱۸ ساعت	آموزش ORM هایبرنیت (Hibernate) جاوا (Java) - کلیک کنید (+)
۲ ساعت	آموزش آشنایی با LINQ to SQL در C# - کلیک کنید (+)
۱ ساعت	آموزش برنامه نویسی تحت شبکه با سی شارپ در قالب پروژه - کلیک کنید (+)
۴ ساعت	آموزش برنامه نویسی با روش سه لایه به زبان سی شارپ - کلیک کنید (+)
۳ ساعت	آموزش Cryptography در دات نت - کلیک کنید (+)
۴ ساعت	آموزش قفل نرم افزاری در C# از طریق رجیستری - کلیک کنید (+)
۱۹ ساعت	آموزش پروژه محور سی شارپ (C#) - سیستم مدیریت آموزشگاه - کلیک کنید (+)
۱۴ ساعت	آموزش پروژه محور سی شارپ (C#) - سیستم مدیریت حسابداری و انبارداری - کلیک کنید (+)
۷ ساعت	آموزش برنامه نویسی سورس کدهای الگوریتمی در سی شارپ - کلیک کنید (+)
۱۰ ساعت	آموزش پروژه محور سی شارپ (C#) - پیاده سازی سیستم هتلداری - کلیک کنید (+)
۴ ساعت	آموزش ساخت سیستم آزمون گیر در سی شارپ - کلیک کنید (+)
۳ ساعت	آموزش ارسال و دریافت پیامک با مودم در سی شارپ (C#) - کلیک کنید (+)
۱۱ ساعت	آموزش پروژه محور سی شارپ (C#) - پیاده سازی سیستم مدیریت آتلیه - کلیک کنید (+)
۷ ساعت	آموزش پروژه محور سی شارپ (C#) - پیاده سازی سیستم نوبت دهی - کلیک کنید (+)
۹ ساعت	آموزش پروژه محور سی شارپ (C#) - پیاده سازی سیستم صدور بلیط - کلیک کنید (+)
۳ ساعت	آموزش رسم نمودار در سی شارپ (C#) - کلیک کنید (+)
۷ ساعت	آموزش پروژه محور سی شارپ (C#) - سیستم مدیریت بانک - کلیک کنید (+)
۵ ساعت	آموزش ساخت بازی با سی شارپ (C#) - کلیک کنید (+)
۲ ساعت	آموزش Metaprogramming (برنامه نویسی متا) در پایتون - کلیک کنید (+)

برنامه نویسی (ادامه از صفحه قبل)	
مدت زمان تقریبی	عنوان آموزش
۱۲ ساعت	آموزش کاربردی Entity Framework در سی شارپ (C#) - کلیک کنید (+)
۶ ساعت	آموزش پروژه محور سی شارپ (C#) - پیاده سازی سیستم مدیریت کارمندان - کلیک کنید (+)
۶ ساعت	آموزش پروژه محور سی شارپ (C#) - پیاده سازی سیستم مدیریت چک های بانکی - کلیک کنید (+)
۴ ساعت	آموزش پروژه محور سی شارپ (C#) - شبیه سازی دستگاه عابر بانک - کلیک کنید (+)
۲ ساعت	آموزش گزارش گیری با کریستال ریپورت و استیمول سافت (Stimulsoft) در سی شارپ (C#) - کلیک کنید (+)
۲ ساعت	آموزش پایگاه داده اس کیو لایت (SQLite) در سی شارپ (C#) - کلیک کنید (+)
۶ ساعت	آموزش پروژه محور سی شارپ (C#) - پیاده سازی نرم افزار حسابداری شخصی - کلیک کنید (+)
۱۵ ساعت	آموزش پروژه محور سی شارپ (C#) - پیاده سازی سیستم رستوران آنلاین - کلیک کنید (+)
۳ ساعت	آموزش کار با بانک اطلاعاتی اکسس (Access) در سی شارپ (C#) - کلیک کنید (+)
۲ ساعت	آموزش کار با فایل ها و دایرکتوری در C# (سی شارپ) - کلیک کنید (+)
۳ ساعت	آموزش پروژه محور سی شارپ C# - پیاده سازی نرم افزار FeedReader (فیدریدر) - کلیک کنید (+)
۴ ساعت	آموزش کار با دستورات گرافیکی در سی شارپ (C#) - کلیک کنید (+)
۳ ساعت	آموزش مقدماتی WPF در سی شارپ (C#) برای ساخت و اجرای اینترفیس - کلیک کنید (+)
۴ ساعت	آموزش WPF در سی شارپ برای ساخت و اجرای اینترفیس - پیشرفته - کلیک کنید (+)
۲ ساعت	آموزش WPF در سی شارپ (C#) برای ساخت و اجرای اینترفیس - تکمیلی - کلیک کنید (+)
۱ ساعت	آموزش ساخت ربات تلگرام با سی شارپ (C#) - کلیک کنید (+)
۱۲ ساعت	آموزش پروژه محور پایگاه داده LocalDB در سی شارپ (C#) - سیستم مدیریت کارمندان - کلیک کنید (+)
۵ ساعت	آموزش کتابخانه Spire.Doc در سی شارپ (C#) برای ایجاد و ویرایش فایل Word - کلیک کنید (+)
۱ ساعت	آموزش پروژه محور سی شارپ (C#) - ساخت برنامه دانلودگر - کلیک کنید (+)

علوم کامپیوتر	
مدت زمان تقریبی	عنوان آموزش
۱۰ ساعت	آموزش ساختمان داده ها - کلیک کنید (+)
۸ ساعت	آموزش پیشرفته ساختمان داده (همراه با حل نمونه سوالات کنکور ارشد و دکتری) - کلیک کنید (+)
۲۰ ساعت	آموزش ساختمان داده ها (مرور - تست کنکور ارشد) - کلیک کنید (+)
۱۱ ساعت	آموزش سیستم های عامل - کلیک کنید (+)
۱۲ ساعت	آموزش سیستم عامل (مرور - تست کنکور ارشد) - کلیک کنید (+)
۸ ساعت	آموزش پایگاه داده ها - کلیک کنید (+)
۸ ساعت	آموزش پایگاه داده ها (مرور - تست کنکور ارشد) - کلیک کنید (+)
۱۰ ساعت	آموزش طراحی و پیاده سازی زبان های برنامه سازی - کلیک کنید (+)
۹ ساعت	آموزش طراحی و پیاده سازی زبان های برنامه سازی (مرور - تست کنکور ارشد) - کلیک کنید (+)
۱۵ ساعت	آموزش طراحی الگوریتم - کلیک کنید (+)
۸ ساعت	آموزش طراحی الگوریتم به همراه حل مثال های عملی - کلیک کنید (+)
۴ ساعت	آموزش روش های حل روابط بازگشتی - کلیک کنید (+)
۲ ساعت	آموزش رابطه های بازگشتی در طراحی الگوریتم و ساختمان گسسته (مرور - تست کنکور ارشد) - کلیک کنید (+)
۱۵ ساعت	آموزش طراحی الگوریتم (مرور - تست کنکور ارشد) - کلیک کنید (+)
۲ ساعت	آموزش روش تقسیم و حل در طراحی الگوریتم (مرور - تست کنکور کارشناسی ارشد) - کلیک کنید (+)
۱۸ ساعت	آموزش مبانی منطق و نظریه مجموعه ها - کلیک کنید (+)
۱۵ ساعت	آموزش مدار منطقی (مرور - تست کنکور ارشد) - کلیک کنید (+)
۱۲ ساعت	آموزش مبانی الکترونیک دیجیتال - بخش اول - کلیک کنید (+)
۱۱ ساعت	آموزش مبانی الکترونیک دیجیتال - بخش دوم - کلیک کنید (+)
۲۰ ساعت	آموزش معماری کامپیوتر با رویکرد حل مساله - مقدماتی - کلیک کنید (+)
۱۴ ساعت	آموزش معماری کامپیوتر با رویکرد حل مساله - پیشرفته - کلیک کنید (+)
۹ ساعت	آموزش نظریه زبان ها و ماشین ها - کلیک کنید (+)
۸ ساعت	آموزش نظریه زبان ها و ماشین (مرور - تست کنکور ارشد) - کلیک کنید (+)
۳ ساعت	آموزش شبیه سازی و تست نظریه زبان ها و ماشین در JFLAP - کلیک کنید (+)
۴ ساعت	آموزش یادگیری عمیق در پزشکی - کلیک کنید (+)
۱۶ ساعت	آموزش ساختمان گسسته با رویکرد حل مساله - کلیک کنید (+)
۱۲ ساعت	آموزش ساختمان گسسته (مرور و حل تست های کنکور کارشناسی ارشد) - کلیک کنید (+)
۱۳ ساعت	آموزش آمار و احتمال مهندسی - کلیک کنید (+)
۱۴ ساعت	آموزش آمار و احتمال مهندسی (حل تمرین و تست کنکور ارشد) - کلیک کنید (+)

علوم کامپیوتر (ادامه از صفحه قبل)	
مدت زمان تقریبی	عنوان آموزش
۱۹ ساعت	آموزش شبکه های کامپیوتری ۱ - کلیک کنید (+)
۲۳ ساعت	آموزش شبکه های کامپیوتری ۲ - همراه با حل مسأله - کلیک کنید (+)
۱ ساعت	آموزش شبیه سازی شبکه های مبتنی بر سیسکو (Cisco) در نرم افزار Boson NetSim - کلیک کنید (+)
۱۵ ساعت	آموزش طراحی کامپایلر - کلیک کنید (+)
۱۵ ساعت	آموزش طراحی کامپایلر (مرور و حل تست های کنکور کارشناسی ارشد) - کلیک کنید (+)
۸ ساعت	آموزش ذخیره و بازیابی اطلاعات - کلیک کنید (+)
۲۳ ساعت	آموزش ساختمان داده ها همراه با پیاده سازی در ++C - کلیک کنید (+)
۹ ساعت	آموزش انتقال داده ها در شبکه های کامپیوتری و اینترنت به همراه حل مسائل - کلیک کنید (+)
۵ ساعت	آموزش مقدماتی Hadoop (هدوپ) برای تجزیه و تحلیل کلان داده - کلیک کنید (+)
۲۲ ساعت	آموزش نظریه صف (Queueing theory) - کلیک کنید (+)
۴ ساعت	آموزش گرافیک کامپیوتری - کلیک کنید (+)
۱۶ ساعت	آموزش گرافیک کامپیوتری با OpenGL - کلیک کنید (+)
۱۲ ساعت	آموزش مهندسی نرم افزار ۱ - کلیک کنید (+)
۱۰ ساعت	آموزش زبان تخصصی مهندسی کامپیوتر - کلیک کنید (+)
۱ ساعت	آموزش مروری بر پیچیدگی محاسبات (Computational Complexity) - کلیک کنید (+)
۱ ساعت	آموزش تبدیل فلوچارت به کد با Flowgorithm - کلیک کنید (+)
۱ ساعت	آموزش نرم افزار RAPTOR برای ترسیم فلوچارت - کلیک کنید (+)
۲ ساعت	آموزش حل سوالات کنکور کارشناسی ارشد کامپیوتر - ۹۴ - کلیک کنید (+)
۱۷ ساعت	آموزش مرور و حل تست های کنکور کارشناسی ارشد معماری کامپیوتر - کلیک کنید (+)
۱۰ ساعت	آموزش برنامه نویسی اسمبلی - کلیک کنید (+)
۱۵ ساعت	آموزش نظریه گراف و کاربردها - کلیک کنید (+)
۱۵ ساعت	آموزش ترکیبیات و کاربردها - کلیک کنید (+)
۸ ساعت	آموزش نظریه بازی ها و کاربرد آن در اقتصاد - کلیک کنید (+)
۴ ساعت	آموزش مدل سازی UML با نرم افزار Rational Rose - کلیک کنید (+)
۱۳ ساعت	آموزش الگوریتم موازی و پردازش موازی - کلیک کنید (+)
۲۰ ساعت	آموزش ترکیب اطلاعات (Information Fusion) - کلیک کنید (+)
۳ ساعت	آموزش نهان نگاری دیجیتال (Digital watermarking) - کلیک کنید (+)
۳ ساعت	آموزش نهان نگاری و پیاده سازی آن در سی شارپ (C#) - کلیک کنید (+)
۱ ساعت	آموزش واترمارکینگ در نهان نگاری با سی شارپ (C#) - کلیک کنید (+)
۱ ساعت	آموزش شبکه های پیچیده پویا (Complex Dynamical Networks) - کلیک کنید (+)

علوم کامپیوتر (ادامه از صفحه قبل)	
مدت زمان تقریبی	عنوان آموزش
۳ ساعت	آموزش تخمین تلاش لازم برای توسعه نرم افزارها با متلب - کلیک کنید (+)
۴ ساعت	آموزش توسعه نرم افزار با متد ICONIX و زبان مدل سازی UML- کلیک کنید (+)
۱ ساعت	آموزش نگارش پروپوزال، پایان نامه و ارائه - ویژه پردازش تصویر و بینایی کامپیوتر - کلیک کنید (+)

هوش مصنوعی	
مدت زمان تقریبی	عنوان آموزش
۴ ساعت	آموزش هوش مصنوعی - کلیک کنید (+)
۱۹ ساعت	آموزش هوش مصنوعی - تکمیلی - کلیک کنید (+)
۲۴ ساعت	آموزش هوش مصنوعی توزیع شده - کلیک کنید (+)
۱۵ ساعت	آموزش هوش مصنوعی (مرور و حل تست کنکور ارشد) - کلیک کنید (+)
۲۸ ساعت	مجموعه آموزش های شبکه های عصبی مصنوعی در متلب - کلیک کنید (+)
۱۰ ساعت	مجموعه آموزش های کاربردی شبکه های عصبی مصنوعی - کلیک کنید (+)
۲ ساعت	آموزش طراحی سیستم های فازی عصبی یا ANFIS با استفاده از الگوریتم های فراابتکاری و تکاملی - کلیک کنید (+)
۵ ساعت	آموزش کاربردی میکروکنترلر - محاسبات پایه طراحی درب های خودکار - کلیک کنید (+)
۴ ساعت	آموزش شبکه عصبی GMDH به همراه پیاده سازی عملی در متلب - کلیک کنید (+)
۳ ساعت	آموزش شبکه های عصبی گازی به همراه پیاده سازی عملی در متلب - کلیک کنید (+)
۳ ساعت	آموزش طبقه بندی و بازشناسی الگو با شبکه های عصبی LVQ در متلب - کلیک کنید (+)
۳ ساعت	آموزش یادگیری عمیق (Deep learning) - کلیک کنید (+)
۳ ساعت	آموزش برنامه نویسی یادگیری عمیق با پایتون (TensorFlow و Keras) - کلیک کنید (+)
۲۱ ساعت	مجموعه آموزش های سیستم های فازی در متلب - کلیک کنید (+)
۱۴ ساعت	مجموعه آموزش های تئوری و عملی الگوریتم ژنتیک - کلیک کنید (+)
۱ ساعت	آموزش پیاده سازی ترکیب الگوریتم ژنتیک و PSO در متلب - کلیک کنید (+)
۱۹ ساعت	مجموعه آموزش های بهینه سازی چند هدفه در متلب - کلیک کنید (+)
۲ ساعت	آموزش حل مساله فروشنده دوره گرد با استفاده از الگوریتم ژنتیک - کلیک کنید (+)
۲ ساعت	آموزش انتخاب ویژگی یا Feature Selection - کلیک کنید (+)
۴ ساعت	آموزش انتخاب ویژگی با استفاده از الگوریتم های فراابتکاری و تکاملی - کلیک کنید (+)
۲ ساعت	آموزش تخمین خطای طبقه بندی یا Classifier Error Estimation - کلیک کنید (+)
۱ ساعت	آموزش کاهش تعداد رنگ تصاویر با استفاده از روش های خوشه بندی هوشمند - کلیک کنید (+)

مهندسی برق	
مدت زمان تقریبی	عنوان آموزش
۱۱ ساعت	آموزش الکترونیک ۱ - کلیک کنید (+)
۲۵ ساعت	آموزش حل تمرین الکترونیک ۱ - کلیک کنید (+)
۱۱ ساعت	آموزش الکترونیک ۱ (مرور و حل تست) - کلیک کنید (+)
۸ ساعت	آموزش الکترونیک ۲ - کلیک کنید (+)
۲۵ ساعت	آموزش حل تمرین الکترونیک ۲ - کلیک کنید (+)
۱۲ ساعت	آموزش الکترونیک ۲ (مرور و حل تست) - کلیک کنید (+)
۱۵ ساعت	آموزش الکترونیک ۳ - کلیک کنید (+)
۱۲ ساعت	آموزش الکترونیک ۳ (مرور و حل مساله) - کلیک کنید (+)
۱۴ ساعت	آموزش مبانی مهندسی برق ۱ - کلیک کنید (+)
۱۴ ساعت	آموزش مبانی مهندسی برق ۱ (مرور و حل مساله) - کلیک کنید (+)
۱۶ ساعت	آموزش مبانی مهندسی برق ۲ - کلیک کنید (+)
۲۴ ساعت	آموزش مدارهای الکتریکی ۱ - کلیک کنید (+)
۱۷ ساعت	آموزش حل تمرین مدارهای الکتریکی ۱ - کلیک کنید (+)
۱۴ ساعت	آموزش مدارهای الکتریکی ۱ (مرور و حل تست) - کلیک کنید (+)
۱۱ ساعت	آموزش مدارهای الکتریکی ۲ - کلیک کنید (+)
۱۸ ساعت	آموزش مدارهای الکتریکی ۲ (حل تمرین) - کلیک کنید (+)
۱۳ ساعت	آموزش مدارهای الکتریکی ۲ (مرور و حل تست) - کلیک کنید (+)
۹ ساعت	آموزش هندسه فینسلر (Finsler Geometry) - کلیک کنید (+)
۱۶ ساعت	آموزش زبان تخصصی مهندسی برق (همراه با حل نمونه سوالات کنکور ارشد) - کلیک کنید (+)
۲ ساعت	آموزش مروری ماشین های الکتریکی ۱ - کلیک کنید (+)
۱۴ ساعت	آموزش ماشین های الکتریکی ۱ - کلیک کنید (+)
۱۶ ساعت	آموزش ماشین های الکتریکی ۱ (مرور و حل تست) - کلیک کنید (+)
۱۷ ساعت	آموزش ماشین های الکتریکی ۲ - کلیک کنید (+)
۱۳ ساعت	آموزش ماشین های الکتریکی ۲ (مرور و حل تست کنکور ارشد) - کلیک کنید (+)
۱۱ ساعت	آموزش ماشین های الکتریکی ۳ - کلیک کنید (+)
۹ ساعت	آموزش مدارهای مخابراتی - کلیک کنید (+)
۹ ساعت	آموزش ماشین های الکتریکی مخصوص - کلیک کنید (+)
۱۴ ساعت	آموزش کنترل موتورهای الکتریکی صنعتی ۱ - کلیک کنید (+)
۱۷ ساعت	آموزش کنترل موتورهای الکتریکی صنعتی ۲ - کلیک کنید (+)
۱۵ ساعت	آموزش کنترل صنعتی - حل مساله - کلیک کنید (+)

مهندسی برق (ادامه از صفحه قبل)	
مدت زمان تقریبی	عنوان آموزش
۹ ساعت	مجموعه آموزش های مدارهای منطقی (طراحی دیجیتال) - کلیک کنید (+)
۱۲ ساعت	آموزش مبانی الکترونیک دیجیتال - بخش اول - کلیک کنید (+)
۱۱ ساعت	آموزش مبانی الکترونیک دیجیتال - بخش دوم - کلیک کنید (+)
۲۰ ساعت	آموزش الکترومغناطیس مهندسی - کلیک کنید (+)
۱۸ ساعت	آموزش الکترومغناطیس مهندسی (مرور و حل تست کنکور ارشد) - کلیک کنید (+)
۹ ساعت	آموزش الکترومغناطیس ۱ - کلیک کنید (+)
۳ ساعت	آموزش مقدماتی شبکه های هوشمند (Smart grid) انرژی الکتریکی - کلیک کنید (+)
۵ ساعت	آموزش کاربردی میکاترونیک - محاسبات پایه طراحی درب های خودکار - کلیک کنید (+)
۲ ساعت	آموزش کامسول مولتی فیزیکس - COMSOL Multiphysics (مباحث منتخب) - کلیک کنید (+)
۴ ساعت	آموزش نرم افزار COMSOL Multiphysics برای پدیده های الکترومغناطیس - کلیک کنید (+)
۸ ساعت	آموزش شبیه سازی مدولاسیون دیجیتال در متلب (MATLAB) - کلیک کنید (+)
۴ ساعت	آموزش نرم افزار سیلواکو (SILVACO) برای شبیه سازی افزاره های نیمه رسانا - کلیک کنید (+)

ابزار مدیریت وبسایت	
مدت زمان تقریبی	عنوان آموزش
۱ ساعت	آموزش کار با cPanel - کلیک کنید (+)
۱ ساعت	آموزش کار با کنترل پنل plesk برای ارائه خدمات وب - کلیک کنید (+)
۱ ساعت	آموزش مدیریت هاست با دایرکت ادمین (DirectAdmin) - کلیک کنید (+)
۳ ساعت	آموزش جوملا (Joomla) برای راه اندازی و مدیریت وب سایت - کلیک کنید (+)
۱ ساعت	آموزش نرم افزار Mobirise برای ساخت وب سایت بدون کدنویسی - کلیک کنید (+)
۸ ساعت	آموزش مقدماتی Oracle APEX برای تولید و توسعه نرم افزارهای تحت وب - کلیک کنید (+)
۱۱ ساعت	آموزش وردپرس (WordPress) - مقدماتی - کلیک کنید (+)
۶ ساعت	مجموعه آموزش های راه اندازی سایت و کار با وردپرس - تکمیلی - کلیک کنید (+)
۳ ساعت	آموزش گوگل آنالیتیکس (Google Analytics) برای تحلیل آمار وب سایت - کلیک کنید (+)
۲ ساعت	آموزش آنالیز و بهینه سازی سایت با GTmetrix - کلیک کنید (+)
۱۰ ساعت	آموزش تست نفوذ در وب و راه های مقابله با آن - کلیک کنید (+)

طراحی و گرافیک	
مدت زمان تقریبی	عنوان آموزش
۱۲ ساعت	آموزش فتوشاپ (Photoshop) - مقدماتی - کلیک کنید (+)
۱۴ ساعت	مجموعه آموزش های کاربردی طراحی و گرافیک با نرم افزار کورل (CorelDRAW) - کلیک کنید (+)
۱۰ ساعت	آموزش کورل پیشرفته - کلیک کنید (+)
۱۰ ساعت	آموزش نرم افزار طراحی گرافیکی Adobe Illustrator - کلیک کنید (+)
۵ ساعت	آموزش نرم افزار طراحی گرافیکی Adobe Illustrator - تکمیلی - کلیک کنید (+)
۱ ساعت	آموزش مقایسه نرم افزارهای گرافیکی Photoshop , CorelDRAW , Illustrator - کلیک کنید (+)
۲ ساعت	آموزش مقدماتی طراحی کاراکتر در Adobe Illustrator (کاراکتر پلیس) - کلیک کنید (+)
۸ ساعت	آموزش نرم افزار Blender برای طراحی و مدل سازی سه بعدی - کلیک کنید (+)
۱۴ ساعت	آموزش مدل سازی کاراکتر در بلندر - ایده پردازی تا ریگ - کلیک کنید (+)
۱ ساعت	آموزش طراحی فلت (Flat design) با کورل (CorelDRAW) - کلیک کنید (+)
۳ ساعت	آموزش فتوشاپ برای استفاده در صنعت چاپ و تبلیغات - کلیک کنید (+)
۳ ساعت	آموزش تایپوگرافی (Typography) - سایه زدن حروف در فتوشاپ (Photoshop) - کلیک کنید (+)
۱ ساعت	آموزش پروژه محور فتوشاپ (Photoshop) - ساخت ماسک (Mask) فانتزی - کلیک کنید (+)
۲ ساعت	آموزش آشنایی با پنل Adjustments در فتوشاپ - کلیک کنید (+)
۳ ساعت	آموزش ساخت جلوه های ویژه در فتوشاپ - کلیک کنید (+)
۶ ساعت	آموزش Adobe Lightroom (ادوبی لایت روم) برای ویرایش و روتوش تصاویر - کلیک کنید (+)
۲ ساعت	آموزش Adobe Dimension برای ایجاد تصاویر سه بعدی محصولات و تبلیغات - کلیک کنید (+)
۲ ساعت	آموزش Adobe Bridge (ادوبی بریج) برای مدیریت فایل های چندرسانه ای - کلیک کنید (+)
۴ ساعت	آموزش ایندیزاین (InDesign) برای چاپ و صفحه آرایی کتاب و مجلات - کلیک کنید (+)
۲۲ ساعت	آموزش نرم افزار SketchBook Pro (اسکچ بوک) برای طراحی آناتومی حیوانات - کلیک کنید (+)
۳ ساعت	آموزش طراحی مفهومی (Conceptual) خودرو با SketchBook - کلیک کنید (+)
۳ ساعت	آموزش نقاشی سه بعدی با نرم افزار Paint 3D ویندوز - کلیک کنید (+)
۳ ساعت	آموزش مبانی هنرهای تجسمی ۱ - کلیک کنید (+)
۱ ساعت	آموزش اصول طراحی نشانه (Sign Design) - کلیک کنید (+)
۹ ساعت	آموزش نقاشی دیجیتال با Photoshop و Mischieff - کلیک کنید (+)
۲ ساعت	آموزش طراحی لوگو با Adobe Illustrator - کلیک کنید (+)
۳ ساعت	آموزش طراحی محیط بازی های دو بعدی با نرم افزار Adobe Illustrator و Photoshop - کلیک کنید (+)


طراحی و گرافیک (ادامه از صفحه قبل)	
مدت زمان تقریبی	عنوان آموزش
۱ ساعت	آموزش طراحی لوگو - آشنایی با مفاهیم و ایده پردازی (همراه با مثال عملی) - کلیک کنید (+)
۲ ساعت	آموزش طراحی برجسب و کاور CD و DVD - کلیک کنید (+)
۵ ساعت	آموزش ابزار سه بعدی در فتوشاپ (Photoshop) - کلیک کنید (+)
۲ ساعت	آموزش فتوشاپ سه بعدی - طراحی و متحرک سازی - کلیک کنید (+)
۲ ساعت	آموزش موتور رندر آرنولد (Arnold Renderer) برای ایجاد جلوه های ویژه در DS Max ^۳ - کلیک کنید (+)
۵ ساعت	آموزش کاربرد فتوشاپ در معماری - کلیک کنید (+)

روباتیک	
مدت زمان تقریبی	عنوان آموزش
۲ ساعت	آموزش رباتیک و ربات های سری - کلیک کنید (+)
۹ ساعت	آموزش برنامه نویسی آردوینو (Arduino) با محوریت پروژه های رباتیک - مقدماتی - کلیک کنید (+)
۷ ساعت	آموزش برنامه نویسی آردوینو (Arduino) - تکمیلی - کلیک کنید (+)
۳ ساعت	آموزش سینماتیک مستقیم و معکوس روبات ها - کلیک کنید (+)
۵ ساعت	آموزش مبانی ربات های برنامه پذیر - کلیک کنید (+)
۱۶ ساعت	آموزش شبیه سازی و کنترل ربات با اندروید (Android) - کلیک کنید (+)
۱۳ ساعت	آموزش مکترونیک کاربردی ۱ - کلیک کنید (+)
۷ ساعت	آموزش مکترونیک کاربردی ۲ - کلیک کنید (+)
۵ ساعت	آموزش کاربردی مکترونیک - محاسبات پایه طراحی درب های خودکار - کلیک کنید (+)
۱۹ ساعت	آموزش برد آردوینو (Arduino) با انجام پروژه های عملی - کلیک کنید (+)
۵ ساعت	آموزش پروژه محور آردوینو با سی شارپ و اندروید - کنترل موتور پله ای - کلیک کنید (+)
۵ ساعت	آموزش پیاده سازی سیستم های کنترلی با آردوینو - کلیک کنید (+)

فصل ۵:

سربارگذاری عملگرها (تعریف مجدد عملگرها)

سربارگذاری یک عملگر، یعنی استفاده دیگری از عملگر به جز کار اصلی اش، مثلاً استفاده از عملگر `==` برای تعیین برابری دو نقطه در مختصات دو بعدی یا استفاده از عملگر `+` برای جمع دو ماتریس و

برای سربارگذاری از تابع `operator` استفاده می‌کنیم. 

* سربار گذاری عملگر `==` برای کلاس مختصات

می‌خواهیم تعریف جدیدی برای عملگر `==` ارائه دهیم تا بتواند مثلاً دو عضو `(2,2)` و `(2,3)` را با هم مقایسه کند. اگر درایه اول هر دو با هم برابر بود و همچنین درایه دوم آنها نیز با هم برابر بود، آن دو با هم برابر می‌باشند. به عبارتی عمل مساوی بودن در مختصات دو بعدی را بررسی می‌کنیم.

```
class c{
    int x,y;
public:
    c(int i , int j) { x=i,y=j;}
    int operator == (c k);
};
int c::operator == (c k){
    if( x==k.x && y==k.y ) return 1; else return 0;
}
main (){
    c ob1(2,2);
    c ob2(2,3);
    if (ob1==ob2)
        cout<<"equal";
    else
        cout<< " not equal";
}
```

خروجی این برنامه `not equal` است. در واقع عملگر `==` وقتی دو شیء را مساوی فرض می‌کند که هر دو عنصر آنها با یکدیگر برابر باشند.

■

* تعریف مجدد عملگر جمع (می‌خواهیم عملگر + را طوری تعریف کنیم که بتواند دو عنصر را به صورت زیر جمع کند):

$(1,2) + (3,4) = (1+3, 2+4) = (4,6)$

```
class c{
    int a; int b;
public:
    c ( ) { }
    c( int x , int y) { a=x; b=y; }
    void f()          { cout<<a<<b; }
    c operator + (c k);
};
c c::operator + (c k){
    c t;
    t.a = k.a+a;
    t.b = k.b+b;
    return t;
}
main (){
    c ob1(1,2);
    c ob2(3,4);
    ob1=ob1+ob2;
    ob1.f();
}
```

خروجی این برنامه 46 است. در ابتدا دو شیء ob1, ob2 ساخته شده و توسط دستور $ob1=ob1+ob2$ ، عمل جمع با تعریف جدید انجام می‌شود. (تذکر: تابع $operator+$ ، دارای یک پارامتر است، چون پارامتر دیگر به صورت ضمنی از طریق اشاره گر $this$ به تابع ارسال می‌شود).

* تعریف مجدد عملگر ++ پیشوندی:

می‌خواهیم عملگر ++ را طوری تعریف کنیم که بتواند روی یک جفت عنصر کار کند:

$++(2,5) = (3,6)$

```
class c{
    int a; int b;
public:
    c() { }
    c( int x , int y) {a=x; b=y; }
    void f() {cout<<a<<b;}
    c operator++( );
};
c c::operator ++ (){
    a++; b++;
    return *this;
}
main (){
    c ob1(1,2);
    ++ob1;
    ob1.f();
}
```


خروجی ۲۳ است. برای تعریف عملگر ++ پسوندی، تابع را به صورت زیر تعریف می‌کنیم:

```
c c::operator ++ (int k)
```



تعریف مجدد عملگرها به کمک تابع دوست

عملگرها را می‌توان به کمک توابعی که عضو کلاس نمی‌باشند، مجدداً تعریف کرد. البته این توابع باید دوست این کلاس باشند. توجه کنید که چون توابع دوست فاقد اشاره گر `this` می‌باشند، عملوندها صریحاً ارسال می‌شوند. بنابراین تابعی که عملگر با یک عملوند را تعریف می‌کند، یک پارامتر دارد و تابعی که عملگری با دو عملوند را تعریف می‌کند، دارای دو پارامتر است.

با استفاده از این توابع دوست، امکان تعریف عملگرهای `->`، `[]`، `()`، `=` نمی‌باشد.

برای تعریف مجدد عملگرهای `--`، `++`، باید از پارامترهای مرجع استفاده کرد.

* تعریف مجدد عملگر + توسط تابع دوست :

```
class c{
    int a; int b;
public: c() { }
    c(int x ,int y) { a=x; b=y; }
    void f()      { cout<<a<<b; }
    friend c operator + (c m, int k);
};
c operator + (c m, int k){
    c t;
    t.a = m.a+k;
    t.b = m.b+k;
    return t;
}
main(){
    c ob1(1,2); c ob2;
    ob2=ob1+2;
    ob2.f();
}
```

توسط عملگر جمع، به هر یک از مقادیر `ob1`، دو واحد اضافه می‌شود :

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

بنابراین خروجی 34 می‌باشد.

تعریف مجدد عملگرهای << و >>

در زبان C++ می‌توان با استفاده از عملگرهای `<<` و `>>` اعمال ورودی و خروجی را بر روی انواع اولیه انجام داد. این عملگرها را می‌توان مجدداً تعریف کرد تا اشیایی از کلاس‌ها را پردازش کنند.

* سربرگذاری عملگر درج‌کننده `<<` : (تذکر: درج‌کننده باید به صورت تابع `friend` تعریف شود).

```

class c{
    int x;
public: c(int y) { x=y;}
    friend ostream &operator <<(ostream &s , c ob);
};
ostream &operator <<(ostream &s , c k) { s <<k.x + 4; return s;}
main(){
    c ob(2);
    cout <<ob;
}

```

خروجی برنامه بالا ۶ می باشد.



عملگرهای مقابل نمی توانند مجددا تعریف شوند:



sizeof ? :: .*



فصل ۶:

قالب‌ها

توسط قالب می‌توان مجموعه کاملی از توابع مرتبط به هم (توابع همنام) را که توابع قالب نام دارند، یا مجموعه‌ای از کلاسهای مرتبط به هم را که کلاسهای قالب نام دارند، مشخص کرد.

تابع قالب

تابع کلی یا تابع قالب با کلمه کلیدی `template` ایجاد می‌شود. این نوع تابع مجموعه‌ای از اعمال را تعریف می‌کند که بر روی انواع مختلفی از داده‌ها انجام می‌شوند. نوع داده‌ای که تابع باید بر روی آن عمل کند، به عنوان آرگومان به آن ارسال می‌شود. به عنوان مثال الگوریتم مرتب‌سازی سریع برای آرایه صحیح و یا آرایه اعشاری، با یک روش عمل می‌کند و با ایجاد یک تابع کلی، الگوریتم را مستقل از نوع داده تعریف می‌کنیم. در این حالت کامپایلر کد مناسب را در زمان اجرا تولید می‌کند. شکل کلی تعریف تابع قالب به صورت زیر می‌باشد:

template <class نوع > نوع برگشتی نام تابع (لیست پارامترها)

{ بدنه تابع }

تابع کلی `f`، محتویات دو متغیر را با هم تعویض می‌کند. (چون شیوه تعویض محتویات دو متغیر به نوع آنها بستگی ندارد، از تابع استفاده می‌کنیم)

```
template <class X> void f(X &a,X &b)
{ X t; temp=a; a=b; b=temp; }
```

در اینجا `X` یک نوع کلی است. اگر در تابع `main()` تابع `f()` یکبار با نوع صحیح و یکبار با نوع اعشاری فراخوانی شود، کامپایلر دو نسخه از این تابع را ایجاد می‌کند. تابع `f()` را می‌توان به صورت زیر نیز نوشت:

```
template <class X>
void f(X &a,X &b)
{ X t; temp=a; a=b; b=temp; }
```

تذکره: در این حالت، نباید دستوری بین دو خط اول نوشت.



* روش پیدا کردن کوچکترین عنصر بین سه عنصر، به نوع آنها بستگی دارد. بنابراین تابع `main()` را به صورت تابع قالب می نویسیم:

```
template <class T>
T min(T x, T y, T z){
    T m = x;
    if (y < m) m = y;
    if (z < m) m = z;
    return m;
}
int main(){
    int a,b, c;
    cin >> a >> b >> c;
    cout << min(a,b,c);
    float m,n,p;
    cin >>m >>n >>p;
    cout << min(m,n,p);
}
```

حل: این برنامه ۳ عدد را از ورودی خوانده و کوچکترین آنها را چاپ می کند.(یکبار اعداد صحیح خوانده و یکبار اعداد اعشاری)

■

* عملکرد برنامه زیر چیست؟

```
template < class T>
void p ( const T *x , const int size ){
    int i;
    for ( i = 0; i < size ; i++ )
        cout <<x[i];
}
int main(){
    const int n=4,m=2;
    int a[n] = {3,2,5,8};
    float b[m] = {1.5,2.5};
    p(a,n);
    p(b,m);
}
```

حل: برنامه داده شده محتویات آرایه با نوع `int` و یا نوع `float` را چاپ می کند.

■

تابعی با دو نوع کلی

می‌توان در دستور `template` بیش از یک نوع داده را تعریف کرد. در مثال زیر، `A` و `B` دو نوع کلی می‌باشند که در فراخوانی‌های مختلف، مقادیر متفاوتی می‌گیرند:

```
template <class A ,class B>
void f(A x,B y){
    cout<<x<<y;}
main(){
    f("ok",2.5);
    f(4L,15);}
```

در فراخوانی اول تابع `f()`، انواع `* char` و `float` به جای نوع‌های `A` و `B` قرار گرفتند و در فراخوانی دوم تابع `f()`، انواع `long` و `int` به جای نوع‌های `A` و `B` قرار گرفتند.

کلاس‌های کلی

علاوه بر توابع کلی، می‌توان کلاس‌ها را نیز به صورت کلی تعریف کرد. در این نوع کلاس، همه الگوریتم‌هایی که توسط کلاس استفاده می‌شود، تعریف می‌شود اما نوع واقعی که باید مورد استفاده قرار بگیرد، هنگامی که اشیا از آن کلاس ایجاد می‌شوند، به صورت پارامتر مشخص می‌گردد. وقتی کلاس کلی ایجاد کنید، عمل مورد نظر را می‌توان بر روی هر نوع داده‌ای انجام داد. بر اساس نوع تعیین شده در زمان ایجاد شیء، کامپایلر کد مربوطه را تولید می‌کند.

شیوه اعلان کلاس کلی:





```
template <class نوع> class نام کلاس {
....}
```

که نوع در هنگام ایجاد شیء مشخص می‌شود.

* خروجی برنامه زیر `A3` می‌باشد:

```
template <class T>
class c{
    private:
        T x;
    public:
        c(T n);
        void f();
};
template <class T>
c<T>::c(T n) { x=n;}
template <class T>
void c<T>::f(){ cout<<x;}
int main(){
    c<char> x('A');
    x.f();
    c<int> y(3);
    y.f();
}
```

■

- قالب کلاس می تواند از قالب کلاس دیگر مشتق شود. 
- قالب کلاس می تواند از کلاس غیر قالب مشتق شود. 
- کلاس قالب می تواند از قالب کلاس مشتق شود. 
- کلاس غیر قالب می تواند از قالب کلاس مشتق شود. 



فصل ۷:

فایل

در زبان C++، فایل به صورت مجموعه‌ای از بایت‌ها در نظر گرفته شده که انتهای هر فایل با کاراکتر انتهای فایل مشخص می‌شود. با باز کردن یک فایل، شیئی ایجاد شده و یک جریان (stream) به آن مربوط می‌شود.

کلاس‌های فایل

برای ورودی-خروجی فایل در C++، باید فایل `<fstream.h>` را به برنامه ضمیمه کرد. این فایل کلاسهای زیر را تعریف می‌کند:

۱- `ifstream` (برای خواندن از فایل)

۲- `ofstream` (برای نوشتن در فایل)

۳- `fstream` (برای خواندن و نوشتن در فایل)

تذکر: کلاس `ifstream` از کلاس `istream`، کلاس `ofstream` از کلاس `ostream` و کلاس `fstream` از کلاس `iostream` مشتق می‌شود.

تذکر: کلاسهای `istream`، `ostream` و `iostream`، از `ios` مشتق می‌شوند.

باز کردن فایل

برای باز کردن فایل، حالت‌های زیر وجود دارد:

باز کردن فایل متنی در حالت ورودی	ios:in
باز کردن فایل متنی در حالت خروجی	ios:out
باز کردن فایل متنی در حالت افزودن به انتها	ios:app
باز کردن فایل در حالت خروجی (در هر جای فایل می‌توان نوشت).	ios:ate
باز کردن فایل باینری	ios:binary
اگر فایل موجود باشد، محتویات قبلی آن از بین می‌رود.	ios:trunk
اگر فایل موجود نباشد، عمل باز کردن با شکست مواجه می‌شود.	ios:nocreate
اگر فایل موجود باشد، عمل باز کردن با شکست مواجه می‌شود.	ios:noreplace

*باز کردن فایل `a.dat` در حالت خروجی:

```
ofstream f; f.open("a.dat", ios::out);
```

که می‌توان از دستورهایی زیر نیز استفاده کرد:

```
ofstream f; f.open("a");
```

البته کلاس `ofstream` دارای سازنده‌ای است که به طور خودکار فایل را باز می‌کند و نیازی به استفاده از دستور `open` نمی‌باشد، بنابراین دستور را می‌توان ساده‌تر نیز نوشت:

```
ofstream f("a");
```

■

دستورات کار با فایل

خواندن کاراکتر	get
نوشتن کاراکتر	put
خواندن رکورد	read
نوشتن رکورد	write
نادیده گرفتن کاراکترهای خوانده شده از یک جریان ورودی	ignore
انتقال اشاره گر خواندن به رکورد دلخواه	seekg
انتقال اشاره گر نوشتن به رکورد دلخواه	seekp
تعیین مکان اشاره گر خواندن	tellg
تعیین مکان اشاره گر نوشتن	tellp
بدست آوردن اطلاعات وضعیت I/O	rdstate

تذکر: هر فایل دارای دو اشاره گر `get` و `put` می باشد. اشاره گر `get` مشخص می کند که عملیات خواندن بعدی باید از آنجا شروع شود و اشاره گر `put` نشان می دهد که عملیات نوشتن بعدی از آنجا باید انجام شود.

تابع `seekg` اشاره گر `get` و تابع `seekp` اشاره گر `put` را حرکت می دهد. این توابع دارای دو ورودی هستند که اولین ورودی تعداد بایتهای حرکت را نشان می دهد و دومین ورودی محل شروع را نشان می دهد که مقادیر `beg` و `cur` و `end` می باشند. (ابتدا، جاری، انتها)

تابع `tellg` محل اشاره گر `get` و تابع `tellp` محل اشاره گر `put` را تعیین می کند.

* نحوه کار با توابع `seekg` و `seekp` :

```
fstream f("test",ios::in | ios::out);
f.seekg(0);
f.seekp(20,ios::cur);
```

دستور اول فایل `test` را به صورت ورودی و خروجی باز می کند و به جریان `f` نسبت می دهد. دستور دوم اشاره گر خواندن فایل را به ابتدای فایل انتقال می دهد و دستور سوم، اشاره گر فایل را با شروع از موقعیت فعلی، ۲۰ بایت به طرف انتهای فایل حرکت می دهد.

* خواندن خط به خط محتویات یک فایل متنی و نمایش آن در صفحه نمایش:

(یک فایل متنی به عنوان آرگومان برنامه به آن داده می شود)

```
main( int argc , char *argv[ ] ){
    char s[255];
    ifstream f(argv[1] );
    f.getline(s,255);
    while(!f.eof( ) ) {
        cout<<s<<endl;
        f.getline(s,255);
    }
    f.close( )
}
```

■

* توسط دستورات زیر یک فایل باینری باز شده و عدد ۳ در آن نوشته می شود:

```
int a=3;
ofstream f("test", ios::out | ios::binary);
f.write((char *) &a, sizeof(int));
```

■

* دستورات زیر یک فایل باینری را باز کرده و عمل خواندن را انجام می دهد:

```
int a;
ifstream f("test", ios::in | ios::binary);
f.read( (*char) &a, sizeof(int) );
```

■

* گرفتن اطلاعات دانشجویان و ذخیره آن در فایل:

```
void enter() {
    struct student {char name[11];char family[21]; int stno;};
    ofstream fp("st.dat", ios::out | ios::binary);
    while(1) {
        cin.getline(s.name,10);
        if (strlen(s.name)==0) break;
        cin.getline(s.family,20);
        cin >> s.stno;
        cin.get( );
        fp.seekp( sizeof(struct student) *s.stno , ios::beg );
        fp.write( (char *) &s , sizeof(struct student) );
    }
    fp.close( );
}
```

■

فصل ۸ :

آزمون

۱- خروجی برنامه زیر کدام است؟

```

class c{
private:
    int a;
public:
    int f() { a=5; cout<<a;}
};
void main(){
    c x;
    x.f();
}

```

پاسخ : جواب 5 است.

۲- خروجی برنامه زیر کدام است؟

```

class a{
private:
    int x;
public :
    a(int=2);
    b(int=5);
    void p() const;
};
a::a(int n) {x=n;}
a::b(int m) {x=m;cout <<"a"<<x; }
void a::p() const{cout<<"b"<<x; }
void main(){
    a f;
    f.p();
    cout<<"c";
    f.b();
}

```

پاسخ : جواب b2ca5 است.

۳- خروجی چیست؟

```
class c{
    int a;
public:
    c()    { a=1;}
    c(int x) { a=x;}
    void f() { cout<<a; }
};
main(){
    c o1;
    o1.f();
    c o2(3);
    o2.f();
}
```

پاسخ : جواب ۱۳ است.

۴- خروجی چیست؟

```
class c{
    int a; int b;
public:
    c(int x=1,int y=2) { a=x;b=y;}
    void f() { cout<<a<<b; }
};
main(){
    c o1;
    o1.f();
    c o2(3);
    o2.f();
}
```

پاسخ : جواب ۱۲۳۲ است.

۵- خروجی برنامه زیر کدام است؟

```
class a{
public:
    static int c;
    a() { ++c; }
    ~a() { cout<<"a"; }
};
int a::c=0;
void main(){
    a w,x;
    cout<<w.c;
    { a w,x,y,z; cout<<w.c; }
    cout<<w.c;
    a y;
    cout<<w.c;
}
```

پاسخ : جواب 26aaaa67aaa است.

۶- خروجی برنامه زیر کدام است؟

```
class a{
public:
    static int c;
    a() { ++c; }
    ~a() { --c; }
};
int a::c=0;
void main( ){
    a w,x;
    cout<<w.c;
    { a w,x,y,z; cout<<w.c; }
    cout<<w.c;
    a y;
    cout<<w.c;
}
```

پاسخ : جواب 2623 است.

۷- خروجی برنامه زیر کدام است؟

```
class t{
    float a,b;
public :
    t(float x=2 , float y=3) : a(x) , b(y) { cout<<"a"; }
    void f() { --a; ++b; }
    double k() { return a+b; }
    void p() { cout<<a<<" "<<b; }
};
void main(){
    t d; cout<<d.k();
    d.f();
    d.p();
}
```

پاسخ : جواب 4 51 a است.

۸- خروجی برنامه زیر کدام است؟

```
class a{
    int x;
public:
    int f(int n) { return --n;}
};
class b : public a{
    int y;
public:
    int g(int n) { return ++n;}
};
void main() {
    b t;
    cout<< t.f(2);
    cout<< t.g(4);
}
```

پاسخ : جواب ۱۵ است.

۹- خروجی برنامه زیر کدام است؟

```
class a{
    int x;
public:
    int f(int n) { return --n; }
};
class b : private a{
    int y;
public:
    int g(int n) { return ++n; }
};
void main(){
    b t;
    cout<< t.f(2);
    cout<< t.g(4);
}
```

پاسخ : خطا دارد.

۱۰- خروجی برنامه زیر کدام است؟

```
class a{
    protected: static int x;
};
int a::x=3;
class b : public a{
    public : void f(int n) {cout<<x+n;}
};
void main(){
    b t;
    t.f(2);
}
```

پاسخ : جواب ۵ است.

۱۱- کدام خط(ها) خطا دارد؟

```

class a{
    protected: int x;
    public:
        void f(int i) {x=i;}
};
class b : private a{
    int y;
    public:
        void g() { y=i+1;} //1
};
class c : public b{
    int z;
    public:
        void h() {z=i+2;} //2
};
main(){
    b ob1;
    c ob2;
    ob1.f(1); //3
    ob2.f(5); //4
}

```

پاسخ : هر چهار خط.

۱۲- خروجی چیست؟

```

class a{
    public:
        a() { cout<<"A"; }
        ~a() { cout<<"B"; }
};
class b : public a{
    public:
        b() { cout<<"C"; }
        ~b() { cout<<"D"; }
};
main(){
    b ob;
}

```

پاسخ : جواب ACDB است.

۱۳- خروجی چیست؟

```

class a{
    protected: int i;
    public:    a(int x) {i=x; cout<<"A"; }
};
class b : public a{
    int j;
    public:
        b(int x, int y) : a(y) { j=x ; cout<<"C"; }
        ~b()      { cout<<"D"; }
        void f() { cout<<i<<j;}
};
main(){
    b ob(1,2);
    ob.f();
}

```

پاسخ : جواب AC21D است.

۱۴- خروجی برنامه زیر کدام است؟

```

class a{
    int i;
    public: a(int n);
    ~a();
    void f1(int n) { i=n; }
    int  f2()      { return i; }
};
a::a(int n)      { i=n; cout<<'b';}
a::~~a()         { cout<<'a'; }
void f3(a ob)   { cout<<ob.f2(); }
void main(){
    a x(2);
    f3(x);
    cout<<x.f2();
}

```

پاسخ : جواب b2a2a است.

۱۵- خروجی کدام است؟

```
class c1{
    int a;
    public : c1(int x) { a=x;}
            int f()  { return a; }
};

class c2{
    int b;
    public : c2(int x) { b=x; }
            int g()  { return b; }
};

class c3 : public c1, public c2{
    int c;
    public : c3 ( int x , int y , int z) : c1(z) ,c2(y) { c=x;}
            void p () { cout <<f () << g() << c; }
};

main() {
    c3 ob(5,6,7);
    ob.p();
}
```

پاسخ : جواب 765 است.

۱۶- خروجی کدام است؟

```

class a{
protected :
    int i;
public :
    a(int x)  { i=x; cout<<"A";}
    ~a()      {cout<<"B";}
};

class b{
protected: int j;
public :   b(int x)  { j=x; cout<<"C";}
           ~b() { cout<<"D";}
};

class c : public a, public b{
    int k;
public :
    c ( int x , int y , int z ) : a(y) , b(z) { k=x; cout<<"E";}
    ~c ( )          { cout<<"F";}
    void p ( )     { cout <<i <<j <<k; }
};

main() {
    c ob(1,2,3);
    ob.p();
}

```

پاسخ : جواب ACE231FDB است.

۱۷- خروجی برنامه زیر کدام است؟

```
class a{
    int x;
    int y;
public:
    a(int i, int j) {x=i; y=j;}
    int f() { return x;}
    int g() { return y;}
};

int main(){
    int i;
    a ob[3]={a(1,2), a(3,4), a(5,6)};
    int s=0;
    for(i=0 ; i<3 ;i++)
        s+=ob[i].f() + ob[i].g();
    cout<<s;
}
```

پاسخ : جواب ۲۱ است.

۱۸- خروجی چیست؟

```
class c{
    int a;
public:
    c(int x) { a=x;}
    friend int f(c);
};

int f(c z)
{ cout<<z.a; }

main(){
    c k(1);
    f(k);
}
```

پاسخ : جواب ۱ است.

۱۹- خروجی برنامه زیر کدام است؟

```
class a{
    int x;
    public:
        f() { cout<<"a"; }
        friend void g() { cout<<"b";}
};

void main(){
    a t;
    t.f();
    g();
}
```

پاسخ : جواب ab است.

۲۰- خروجی برنامه زیر کدام است؟

```
class a {
    private: int x;
    public:
        a() : x(5) {cout<<"a";}
        void print() const { cout << x ; }
        friend void f(a &, int);
};

void f(a &c, int val)
{ c.x = val; }

int main(){
    a ob;
    ob.print();
    f ( ob, 6);
    ob.print();
}
```

پاسخ : جواب a56 است.

۲۱- خروجی چیست؟

```

class c{
    int a;
public:
    c(int x) { a=x;}
    friend class d;
};
class d{
public :
    int f(c z) { cout<<z.a;}
};
main(){
    c m(1);
    d n;
    n.f(m);
}

```

پاسخ : جواب ۱ است.

۲۲- خروجی چیست؟

```

class c{
    static int a;
    int b;
public:
    f(int x,int y) { a=x;b=y;}
    void g() { cout<<a<<b;}
};
int c::a;
main(){
    c o1,o2;
    o1.f(1,2);
    o1.g();
    o2.f(3,4);
    o2.g();
    o1.g();
}

```

پاسخ : جواب 123432 است.

۲۳- خروجی چیست؟

```

class c{
    public:
        static int a;
};
int c::a;
main(){
    c ::a=5;
    cout<<c::a;
    c x;
    cout<<x.a;
}

```

پاسخ : جواب ۵۵ است.

۲۴- خروجی چیست؟

```

class a{
    public: static int c;
        a() {c++;}
        ~a() {c--;}
};
int a::c;
void f()
{
    a b;
    cout<<a::c;
}
main(){
    a ob1,ob2;
    cout<<a::c;
    a ob3;
    cout<<a::c;
    f();
    cout<<a::c;
}

```

پاسخ : جواب 2343 است.

۲۵- کدام خط خطا دارد؟

```
class c{
    static int a; int b;
public:
    int d;
    static f() //1
    {
        a=2; //2
        b=3; //3
        d=4; //4
    }
};
int c::a=5;
main(){
    c o; o.f(); //5
}
```

پاسخ : خط ۳ و ۴

۲۶- خروجی چیست؟

```
class a{
    static int r;
public:
    static int f();
};
int a::r=1;
int a::f()
{ cout<<r; }
int main() {
    a ob1;
    a::f();
    ob1.f();
}
```

پاسخ : جواب 11 است.

۲۷- خروجی چیست؟

```

class a{
public:
    static int r;
    static int f(int x) { r = x; }
};
int a::r;
int main(){
    a::f(5);
    cout <<a::r;
    a ob1;
    cout <<a::r;
}

```

پاسخ: جواب 55 است.

۲۸- خروجی چیست؟

```

class c {
    int x;
    const int c;
public:
    c( int c=0, int i=1 );
    void f() { x = c; }
    void p() const;
};
c::c( int c, int i ) : x(c), c(i)
{ x=c; c=i; }
void c::p() const
{ cout << x<< c ; }
int main() {
    c a(2,5);
    for ( int j = 0; j <2; j++ )
    {
        a.f();
        cout << j ;
        a.p();
    }
}

```

پاسخ: جواب 055155 است.

۲۹- کدام خط(ها)، خطا دارد؟

```
class c{
public:
    int a;
    c() {cout<<"ok";} //1
};
main() {
    const c o; //2
    o.a=5; //3
}
```

پاسخ: جواب 3 است.

۳۰- چند خطا وجود دارد؟

```
class a{
    int data;
    static int count;
public:
    a(int y=10) : data(y) { cout<<"a"; }
    int f() const { return ++data; }
    static int g() { cout<<data; return count; }
};
```

پاسخ: جواب 2 است.

۳۱- کدام درست است؟

```
class a{
private: int x;
        static int c;
public:
    a(int y=5) { x=++y;}
    int f() { return ++x; }
    static int g() const {cout<<x; }
};
main() {
    a t;
    cout <<t.f();
    t.g();
}
```

پاسخ: اگر static قبل از تابع g را برداریم، خروجی دارد.

۳۲- خروجی چیست؟

```

class a{
    int x;
public:
    a()    { x=0;}
    a(int y) { x=y;}
    int f() { return x;}
};
int main() {
    a z[4]={1,2,3,4};
    a *p;
    int i;
    p= z;
    for(i=0 ; i<4 ; i++)
    {
        cout<<p->f();
        p++;
    }
}

```

پاسخ : جواب 1234 است.

۳۳- خروجی برنامه زیر کدام است؟

```

class count{
public:
    int x;
    void p() { cout<<x++;}
};

void main() {
    count c,*k=&c;
    c.x=1;
    c.p();
    k->x=3;
    k->p();
    (*k).x=4;
    (*k).p();
}

```

پاسخ : جواب 134 است.

۳۴- خروجی کدام است؟

```
class a{
public:
    int x;
    a(int y) { x=y;}
};

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

پاسخ : جواب 5 است.

۳۵- خروجی برنامه زیر کدام است؟

```
class a{
public :
    int b;
    void f() { cout<<b++;}
};

void main(){
    a c; c.b=5; c.f(); c.f();
    a t; t=c; t.f();
    a *p;
    p=&c;
    p->b=3;
    p->f();
    (*p).b=8;
    (*p).f();
}
```

پاسخ : جواب 56738 است.

۳۶- خروجی برنامه زیر کدام است؟

```
class a{
    int x;
public:
    a(int y) {x=y;}
    int f() {return x;}
};
void main(){
    a t(5);
    a *p;
    p=&t;
    cout<<t.f();
    cout<<p->f();
}
```

پاسخ : جواب 55 است.

۳۷- خروجی چیست؟

```
class a{
public:
    a(int x) {y=x;};
    int f() { return y+2;}
    int y;
};
int main(){
    a ob(2);
    int a::*p1;
    p1=&a::y;
    cout<<ob.*p1;
    int (a::*p2)();
    p2 = & a::f;
    cout<<(ob.*p2)();
}
```

پاسخ : جواب 24 است.

۳۸- با توجه به برنامه زیر، در محل خالی کدام باید قرار بگیرد؟

```
void f(int &a)
{ a++; }
main() {
  int x=5;
  .....
  cout<<x;
}
```

پاسخ : جواب f(x) است.

۳۹- خروجی چیست؟

```
main() {
  int x;
  int &p=x;
  x=2;
  cout<<x+p;
  p=3;
  cout<<x+p;
  ++p;
  cout<<x+p;
}
```

پاسخ : جواب 468 است.

۴۰- خروجی چیست؟

```
int main(){
  int x=3;
  int &y=x;
  cout <<x<<y ;
  y=7;
  cout <<x<<y;
}
```

پاسخ : جواب 3377 است.

۴۱- خروجی چیست؟

```

class a{
public:
    a(int x) { cout<<'a'; k=x;}
    ~a()     { cout<<'b';}
    int f() { return k+2;}
    int k;
};
int main(){
    int (a::*p)();
    a ob(2);
    p=&a::f;
    cout << (ob.*p)();
}

```

پاسخ : جواب a4b است.

۴۲- خروجی برنامه زیر کدام است؟

```

class count {
public:
    int x;
    void print() { cout <<x; }
};
int main(){
    count a;
    count *p=&a;
    count &c=a;
    a.x=1; a.print();
    c.x = 2; c.print();
    p->x = 3; p->print();
}

```

پاسخ : جواب 123 است.

۴۳- خروجی چیست؟

```

class test {
    private: int x;
    public: test(int = 0); void print() const;
};
test::test(int value) : x(value)
    { cout <<x; }
void test::print() const
    { cout <<x; cout<<this->x; cout<<(*this).x; }
int main() {
    test a(5);
    a.print();
}

```

پاسخ : جواب 5555 است.

۴۴- خروجی برنامه زیر کدام است؟

```

int f(int x) { return ++x; }
int f(int y,int z) { return y+z; }
char f(char ch) { return --ch; }
void main() {
    cout<<f(3);
    cout<<f(3,5);
    cout<<f('B');
}

```

پاسخ : جواب 48A است.

۴۵- خروجی برنامه زیر کدام است؟

```

void f(int x) { cout<< ++x; }
void f(int y, int z) { cout<<y+z; }
void main(){
    void (*p1)(int);
    void (*p2)(int,int);
    p1=f; p2=f;
    p1(3); p2(6,2);
}

```

پاسخ : جواب 48 است.

۴۶- خروجی برنامه زیر کدام است؟

```
int f(int x) { return ++x;}
int f(int y) { return ++y;}
void main(){
    cout<<f(3); cout<<f(5);
}
```

(۴) خطا دارد.

(۳) 46

(۲) 45

(۱) 43

پاسخ : خطا دارد.

۴۷- خروجی برنامه زیر کدام است؟

```
int f(int a=1, int b=1, int c=1);
int main(){
    cout << f();
    cout << f(2);
    cout << f(3,4);
    cout << f(5,6,7);
}
int f(int a, int b, int c)
{
    return a+b+c;
}
```

پاسخ : جواب 34818 است.

۴۸- خروجی برنامه زیر کدام است؟

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

پاسخ : جواب 359 است.

۴۹- خروجی چیست؟

```

class a{
    public : virtual void f(){ cout<<"A"; }
};
class b : public a{
    public : void f() {cout<<"B"; }
};
class c : public a{
    public: void f() {cout<<"C"; }
};
main(){
    a *p,x; b ob1; c ob2;
    p=&x;    p->f();
    p=&ob1;  p->f();
    p=&ob2;  p->f();}

```

پاسخ : جواب ABC است.

۵۰- اگر کلمه **virtual** در تست قبل را برداریم، خروجی چیست؟

پاسخ : جواب AAA است.

۵۱- خروجی چیست؟

```

class a{
    public:
        virtual void f(){ cout<<"A"; }
};
class b : public a{
    public:
        void f( ) {cout<<"B"; }
};
class c: public a{
    public:
        void f() {cout<<"C"; }
};
void m(a &r) { r.f(); }
main(){
    a x;
    b ob1;
    c ob2;
    m(x);
    m(ob1);
    m(ob2);}

```

پاسخ : جواب ABC است.

۵۲- خروجی چیست؟

```
class number{
    protected:
        int xl;
    public:
        void set(int i){ x=i;}
        virtual void f()=0;
};
class cdec : public number{
    public:
        void f() {cout<<x; }
};
class chex : public number{
    public:
        void f() {cout<<hex<<x; }
};
main() {
    cdec d;
    chex h;
    d.set(2);
    d.f();
    h.set(15);
    h.f();
}
```

پاسخ : خطا دارد.

۵۳- خروجی چیست؟

```

class c{
    int a;
    int b;
public:
    c() {}
    c(int x ,int y) { a=x; b=y; }
    void f()          { cout<<a<<b; }
    c operator + (c op2);
};
c c::operator + (c op2)
{
    c t;
    t.a=op2.a+a;
    t.b=op2.b+b;
    return t;
}
main () {
    c ob1(1,2);
    c ob2(3,4);
    ob1=ob1+ob2;
    ob1.f();
}

```

پاسخ : جواب 46 است.

۵۴- خروجی چیست؟

```

class c{
    int a;
    int b;
public:
    c() { }
    c(int x ,int y) { a=x; b=y; }
    void f() { cout<<a<<b; }
    friend c operator + (c op1, int op2);
    friend c operator + (int op1, c op2);
};

c operator + (c op1,int op2)
{ c t; t.a = op1.a+op2; t.b = op1.b+op2; return t; }

c operator + (int op1,c op2)
{ c t; t.a = op1+op2.a; t.b = op1+op2.b; return t; }

main () {
    c ob1(1,2);
    c ob2(3,4);
    ob2=ob1+5;
    ob2.f();
    ob1=2+ob2;
    ob1.f();
}

```

پاسخ : جواب 6789 است.

۵۵- خروجی چیست؟

```

class c{
    int a; int b;
public:
    c() {}
    c(int x ,int y) { a=x; b=y; }
    void f()          { cout<<a<<b; }
    c operator ++ ( );
};
c c::operator ++ ( )
{a++; b++; return *this; }
main () {
    c ob1(1,2);
    ++ob1;
    ob1.f();
}

```

پاسخ: جواب 23 است.

۵۶- در صورت ورود عدد ۴، خروجی کدام است؟

```

class a{
    int x;
public: friend istream &operator>>(istream &stream , a &ob); // extractor
        friend ostream &operator<<(ostream &stream , a ob); //insertion
};
istream &operator >> (istream &stream, a &ob)
{ stream>>ob.x; return stream; }
ostream &operator <<(ostream &stream , a ob)
{ stream<<ob.x+2; return stream; }
main() {
    a m;
    cin >>m;
    cout <<m;
}

```

پاسخ: جواب 6 است.

۵۷- خروجی کدام است؟

```

class a{
    int x,y;
public: a( int i ,int j) { x=i,y=j;}
    int operator == ( a ob2);
};

int a::operator == ( a ob2){
    if( x==ob2.x && y==ob2.y ) return 1; else return 0;
}

main (){
    a m(1,2), n(1,3);
    if (m==n) cout<<"equal"; else cout<<"not equal";}

```

پاسخ : جواب not equal است.

۵۸- خروجی برنامه زیر کدام است؟

```

template <class A>
void f(A x) { cout<<x; }
void main() {
    int b=2; char c='M';
    f(b);
    f(c);}

```

پاسخ : جواب 2M است.

۵۹- اگر از ورودی اعداد 1 2 3 وارد شوند، خروجی کدام است؟

```

template <class T>
T f( T v1, T v2, T v3 )
{
    T m = v1;
    if ( v2 > m ) m = v2;
    if ( v3 > m ) m = v3;
    return m;
}

int main() {
    int a,b,c;
    cin >> a >> b >> c;
    cout << f( a,b,c); }

```

پاسخ : جواب 3 است.

۶۰- خروجی چیست؟

```

template < class T >
void f( const T *x, const int n ){
    for ( int i = 0; i < n; i++ )
        cout << x[i] << " ";
int main(){
    const int x = 3, y = 2, z = 5;
    int a[x] = { 1, 2, 3 };
    double b[y] = { 1.5, 2.5 };
    char c[z] = "HELLO";
    f(a,x);
    f(b,y);
    f(c, z);}

```

پاسخ : جواب 1 2 3 1.5 2.5 HELLO است.

۶۱- خروجی برنامه زیر کدام است؟

```

class c{ protected: int a; };
void main(){c x;
    cout<<x.a;}

```

پاسخ : خطا دارد.

۶۲- خروجی برنامه زیر کدام است؟

```

class a{
    private:
        int x;
    public:
        static int c;
        a() { ++c; cout<<'(';}
        a(int m) { x=m;cout<<'*'; }
        ~a() { cout<<")"; }
};
int a::c=0;

```

```

void main(){
    a w,x,t(1);
    cout<<w.c;
    { a y(2),z; cout<<w.c; }
    cout<<w.c<<x;}

```

پاسخ : جواب (((*2*(3))3))) است.

۶۳- خروجی برنامه زیر کدام است؟

```
class a{
public:
    static int c;
    a() {c=2; }
    ~a() {--c; }
    ~a() {++c; }
};
int a::c=8;
void main(){
    a x;
    cout<<x.c;
    a y;
    cout<<x.c;
}
```

پاسخ : خطا دارد.

۶۴- کدام خط(ها) دارای خطا است؟

```
class t{
    int a,b;
public :
    t(int x=2 , int y=3) : a(x) , b(++y) {p(); } //1
    p() { cout<<a+b; } //2
};
void main(){
    t d;
    d.t(); //3
}
```

پاسخ : جواب ۳ است.

۶۵- خروجی برنامه زیر کدام است؟

```
class a{
    int x;
    public:
        int f(int n) { cout<< n+1;}
};
class b : public a{
    int y;
    public:
        int g(int n) { f(5); cout<<n+2;}
};
void main() {
    b t;
    t.f(1);
    t.g(2);
}
```

پاسخ : جواب 264 است.

۶۶- خروجی برنامه زیر کدام است؟

```
class a{
    public:
        int f() { cout<<'1'; }
};
class b : a{
    private: int t;
};
void main(){
    b x;
    a y;
    x.f();
    y.f();
}
```

پاسخ : خطا دارد.

۶۷- اعضای قسمت `protected` کلاس `c2` عبارتند از :

```
class c1{
    private    : int  a;
    protected : int  b;
    public    : int  c;
};

class c2 : protected c1{
    private    : int  d;
    protected : int  e;
    public    : int  f;
};
```

پاسخ : جواب e,b,c است.

۶۸- اعضای قسمت `public` کلاس `c2` عبارتند از :

```
class c1{
    private    : int  a;
    protected : int  b;
    public    : int  c;
};

class c2 : public c1{
    private    : int  d;
    protected : int  e;
    public    : int  f;
};
```

پاسخ : جواب c,f است.

۶۹- کدام خط(ها) خطا دارد؟

```

class a{
    protected: int x;
    public:    f(int i) {x=i;}
};
class b : private a{
    public: void g() { y=x+1;} //1
    int y;
};
class c : public b{
    int z;
    public: void h() {z=y+2;} //2
};
main(){
    b m;
    c n;
    m.g(); //3
    n.f(1); //4
}

```

پاسخ : خط ۴ .

۷۰- خروجی چیست؟

```

class a{
    int i;
    public: a(int x){i=x;}
           ~a() {cout<<i;}
};
class b : public a{
    int j;
    public:    b(int x, int y) : a(x) { j=y ;cout<<j; }
};
main() {
    b ob1(1,2);
    b ob2(3,4);
}

```

پاسخ : جواب ۲۴۳۱ است.

۷۱- خروجی کدام است؟

```

class c1{
    int a;
    public : c1(int x) { a=x;}
            int f() { return a; }
};
class c2{
    int b;
    public : c2(int x) { b=x; }
};
class c3 : public c1, public c2{
    int c;
    public : c3 ( int x , int y , int z) : c1(z) ,c2(y) { c=x;}
            void p () { cout <<f () + c; }
};
main() {
    c3 ob(1,2,3);
    ob.p();
}

```

پاسخ : جواب 4 است.

۷۲- خروجی کدام است؟

```

class a{
    public: int i;
            a(int x) { i=x; cout<<i;}
};
class b: public a{
    public : b(int x) : a(x++) { }
};
class c : public b{
    public : c( int x ) : b(++x) { }
};
main() {
    c ob(1);
}

```

پاسخ : جواب 2 است.

۷۳- خروجی برنامه زیر کدام است؟

```

class a{
    int x; int y;
public:
    a(int i, int j) {x=++i; y=i+j;}
    int f() { return x;}
};
int main(){
    int i;
    a ob[2]={a(3,1),a(2,4)};
    int s=0;
    for(i=0 ; i<2 ;i++)
        s+=ob[i].f();
    cout<<s;}

```

پاسخ : جواب 7 است.

۷۴- خروجی چیست؟

```

class c{
    int a;
public:
    c(int x)      { a=x;}
    friend int h(c) {c z; cout<<z.a; };
};
main(){
    c k(5);
    h(k);}

```

پاسخ : خطا دارد.

۷۵- خروجی برنامه زیر کدام است؟

```

class a{
    int x;
public:
    friend void g() { cout<<'1';} //1
};
void main(){
    a t;
    g(); //2
    t.g(); //3 }

```

پاسخ : خط ۳ خطا دارد.

۷۶- خروجی برنامه زیر کدام است؟

```
class a {
    int x;
public:
    a() : x(2) {cout<<x<<"*";}
    friend void f(a &, int);
    void p() { cout << ++x ; }
};
```

```
void f(a &k , int n)
{ k.x = --n; }
```

```
int main(){
    a ob;
    ob.p();
    f ( ob,7);
    ob.p();
}
```

پاسخ : جواب 2*38 است.

۷۷- خروجی چیست؟

```
class c{
    int a;
public:
    c(int x) { a=x;}
    friend class d;
};
```

```
class d{
public :
    int f(c z) { cout<<z.a+3;}
};
```

```
main(){
    c ob1(5);
    d ob2;
    ob2.f(ob1);
}
```

پاسخ : جواب 8 است.

۷۸- خروجی چیست؟

```

class c{
    static int a;
    int b;
public:   f(int x,int y) { a=x;b=y;}
        void g() { cout<<a+b++;}
};
int c::a;
main(){
    c o1,o2;
    o1.f(5,4);
    o1.g();
    o2.f(2,6);
    o2.g();
    o1.g();
}

```

پاسخ : جواب 987 است.

۷۹- خروجی چیست؟

```

class c{
public:
    static int a;
};
int c::a;
main(){
    c ::a=1;
    cout<<c::a++;
    c x;
    cout<<x.a;
}

```

پاسخ : جواب 12 است.

۸۰- خروجی چیست؟

```

class a{
    public: static int c;
           a() {c++;}
           ~a() { c++;}
};
int a::c;
void f() { a b; cout<<a::c; }
main(){
    a ob1,ob2;
    f();
    cout<<a::c;
    a ob3,ob4;
    cout<<a::c;
    f();
    cout<<a::c;
}

```

پاسخ: جواب 34678 است.

۸۱- کدام خط خطا دارد؟

```

class c{
    int a;
    static int b;
public:
    int d;
    static g() //1
    {
        a=1; //2
        b=2; //3
        d=3; //4
    }
};
int c::a=4; //5
main(){
    c ob;
    ob.g();
}

```

پاسخ: خط های 2,4,5

۸۲- خروجی چیست؟

```

class c {
    int x;
    const int c;
public:
    c(int c=0, int i=1);
    void f() { x=++c; }
    void p() const;
};
c::c(int c, int i) : x(c), c(i) { x = c; c = i; }
void c::p() const { cout << x << c; }
int main() {
    c a(1,2);
    a.f(); a.p(); a.f(); a.p();
}

```

پاسخ : خطا دارد.

۸۳- کدام خط(ها) دارای خطا می باشد؟

```

class a{
    static int x;
    const int y;
public:
    a(int m=1) : x(m) { cout<<x; } //1
    static int f() const { return ++x; } //2
    g() { cout<<x; return y; } //3
};

```

پاسخ : خط های ۲ و ۳.

۸۴- خروجی چیست؟

```

class c{
    int x;
public:
    c()      { x=1;}
    c(int y) { x=++y;}
    int fun() { return x--;}
};
int main() {
    c z[3]={1,2,3};
    c *q;
    int i;
    q= z;
    for(i=0 ; i<3 ; i++)
    {
        cout<<q->fun();
        q++;
    }
}

```

پاسخ : جواب 234 است.

۸۵- خروجی برنامه زیر کدام است؟

```

class count{
public :
    int x;
    void p() { cout<<++x;}
};

void main() {
    count c,*k=&c;
    c.x=2;
    c.p();
    k->x=4;
    k->p();
    (++*k).x=1;
    (*k).p();
}

```

پاسخ : جواب 352 است.

۸۶- خروجی چیست؟

```

class a{
public:
    a(int x) {y=x+2;}
    int f() { return ++y;}
    int y;
};
int main(){
    a ob(1);
    int a::*p1;
    p1=&a::y;
    cout<<ob.*p1;
    int (a::*p2)();
    p2 = & a::f;
    cout<<(ob.*p2)();
}

```

پاسخ : جواب 34 است.

۸۷- خروجی برنامه زیر کدام است؟

```

void h (int x) { cout<< x+2; }
void h(int y, int z) { cout<<y*z; }
void main(){
    void (*p) (int);
    void (*q) (int,int);
    p=h;
    q=h;
    p(1);
    q(2,3);
}

```

پاسخ : جواب 36 است.

۸۸- خروجی چیست؟

```

class a{
    public : virtual void f(){ cout<<'1'; }
};
class b : public a{
    public : void f() {cout<<'2'; }
};
class c : public a{
    public: void f() {cout<<'3'; }
};
main(){
    a *p,x;
    b ob1;
    c ob2;
    p=&x;    p->f();
    p=&ob1;  p->f();
    p=&ob2;  p->f();}

```

پاسخ : جواب 123 است.

۸۹- خروجی چیست؟

```

class c{
    int a;
    int b;
public:
    c() {}
    c(int x ,int y) { a=x; b=y; }
    void f()      { cout<<a<<b; }
    c operator -(c op2);
c c::operator -(c op2)
{
    c t;
    t.a=op2.a-a;
    t.b=op2.b-b;
    return t;
}
main () {
    c ob1(1,2);
    c ob2(3,4);
    ob1=ob1-ob2;
    ob1.f();}

```

پاسخ : جواب 2-2- است.

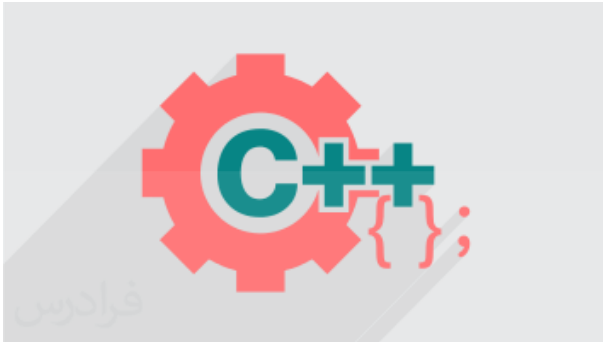
دسته بندی موضوعی آموزش های فرادرس، در ادامه آمده است:

 <p>مهندسی برق الکترونیک و رباتیک</p>	 <p>هوش مصنوعی</p>	 <p>آموزش‌های دانشگاهی و تخصصی</p>	 <p>برنامه‌نویسی</p>
<p>مهندسی برق الکترونیک و رباتیک - کلیک (+)</p>	<p>هوش مصنوعی - کلیک (+)</p>	<p>آموزش‌های دانشگاهی و تخصصی - کلیک (+)</p>	<p>برنامه‌نویسی - کلیک (+)</p>
 <p>مهارت‌های دانشگاهی</p>	 <p>مباحث مشترک</p>	 <p>دروس دانشگاهی</p>	 <p>مهندسی نرم‌افزار</p>
<p>مهارت‌های دانشگاهی - کلیک (+)</p>	<p>مباحث مشترک - کلیک (+)</p>	<p>دروس دانشگاهی - کلیک (+)</p>	<p>مهندسی نرم‌افزار - کلیک (+)</p>
 <p>آموزش‌های عمومی</p>	 <p>طراحی و توسعه وب</p>	 <p>نرم‌افزارهای عمومی</p>	 <p>نرم‌افزارهای تخصصی</p>
<p>آموزش‌های عمومی - کلیک (+)</p>	<p>طراحی و توسعه وب - کلیک (+)</p>	<p>نرم‌افزارهای عمومی - کلیک (+)</p>	<p>نرم‌افزارهای تخصصی - کلیک (+)</p>

منبع مطالعاتی تکمیلی مرتبط با این کتاب

آموزش ویدئویی برنامه‌نویسی C++

یادگیری یک زبان برنامه‌نویسی دید ما را به علم کامپیوتر بسیار گسترده‌تر می‌کند. از طرفی برنامه‌نویسی نحوه فکر کردن را به ما می‌آموزد. عمومیت زبان C++ در میان زبان‌های برنامه‌نویسی بسیار بالا است و می‌تواند به عنوان اولین زبان یاد گرفته شود و به پیش‌نیاز دیگر احتیاج نباشد.



در این آموزش، مفاهیم شیء‌گرایی در ++C آموزش داده شده است. مفاهیمی مانند: کلاس، تابع دوست، کلاس دوست، ارث بری، پلی مورفیزم (Polymorphism)، سر بارگذاری، قالب. در این آموزش، ابتدا مفاهیم به صورت پاورپوینت آموزش داده می‌شود و سپس این مفاهیم در محیط عملی با مثال‌های متنوعی بررسی می‌شوند. از نقاط قوت این آموزش، تدریس مطالب شیء‌گرایی با ساده‌ترین روش و با ذکر مثال‌های بسیار است.

مدرس: مهندس فرشید شیرافکن

مدت زمان: ۸ ساعت

faradars.org/fvcp95042

جهت مشاهده آموزش ویدئویی این آموزش [کلیک](#) کنید.