```cpp
/*
 * Advanced programming
 * Alireza Akhavan Pour
 * Akhavan@AlirezaWeb.com
 * date: 1394/12/01
 */

int main()
{
    cout<< "In the name of God";
    Lecture_7();
    return 0;
}

string Lecture_7()
{
    cout << "1. decomposition" <<endl;
    cout << "2.string to numeric" <<endl;
    cout << "3.pointer" <<endl;
    return ":)"
}
```

تابع سازنده برای روابط decomposition
(استفاده از یک کلاس در کلاس دیگر)

```cpp
class Circle {
    double radius;
  public:
    Circle(double r) : radius(r) { }
    Circle(){radius=1; }
    double area() {return radius*radius*PI;}
};

class Cylinder {
    Circle base;
    double height;
  public:
    Cylinder(double r, double h)
        : base (r)
    {
        height=h;
    }
    double volume() {return base.area() * height;}
};
```

```cpp
class Circle {
    double radius;
  public:
    Circle(double r) : radius(r) { }
    double area() {return radius*radius*PI;}
};

class Cylinder {
    Circle base;
    double height;
  public:
    Cylinder(double r, double h)
        : base (r)
    {
        height=h;
    }
    double volume() {return base.area() * hei
};
```

```
class Circle {
    double radius;
  public:
    Circle(double r) : radius(r) { }
    Circle() { radius=1;}
    void set_r (int r) {radius=r;}
    double area() {return radius*radius*PI;}
};

class Cylinder {
    Circle base;
    double height;
  public:
    Cylinder(double r, double h)
    {
        height=h;
        base.set_r=r;
    }
    double volume() {return base.area() * height;}
};
```

```
class Circle {
    double radius;
  public:
    Circle(double r) : radius(r) { }
    //Circle() { radius=1;}
    void set_r (int r) {radius=r;}
    double area() {return radius*radius*PI;}
};

class Cylinder {
    Circle base;
    double height;
  public:
    Cylinder(double r, double h)
    {
        height=h;
        base.set_r=r;
    }
    double volume() {return base.area() * height;}
};
```

```
class Circle {
    double radius;
  public:
    //Circle(double r) : radius(r) { }
    //Circle() { radius=1;}
    void set_r (int r) {radius=r;}
    double area() {return radius*radius*PI;}
};

class Cylinder {
    Circle base;
    double height;
  public:
    Cylinder(double r, double h)
    {
        height=h;
        base.set_r=r;
    }
    double volume() {return base.area() * height;}
};
```

در اینجا چون تابع سازنده نداریم، کامپایلر تابع سازنده‌ی پیش فرض (بدون ورودی) برای کلاس ما در نظر میگیرد و تمامی فیلدها را مقدار دهی اولیه با مقادیر نامطلوب میکند.
در نتیجه هنگامی که وارد تابع سازنده ی **Cylander** شده، شئ **base** از کلاس **Circle** با آن تابع سازنده ی پیش فرض ساخته شده و برنامه ارور نمیدهد.

فراموش نکنید که تنها و تنها هنگامی که هیچ تابع سازنده ای تعریف نکنیم خود کامپایلر تابع سازنده پیش فرض تعریف میکند.

0_decomposition.cpp

تبدیل رشته به عدد

www.cplusplus.com/reference/string/

Functions

Convert from strings

stoi C++11	Convert string to integer (function template)
stol C++11	Convert string to long int (function template)
stoul C++11	Convert string to unsigned integer (function template)
stoll C++11	Convert string to long long (function template)
stoull C++11	Convert string to unsigned long long (function template)
stof C++11	Convert string to float (function template)
stod C++11	Convert string to double (function template)
stold C++11	Convert string to long double (function template)

Convert to strings

to_string C++11	Convert numerical value to string (function)
to_wstring C++11	Convert numerical value to wide string (function)

stoi

```
#include <iostream>
#include <string>
using namespace std;


int main()
{
    string s1="salam";
    string s2="125";
    string s3="12.5";
    string s4="123salam";
    string s5="99999999999999999";

    int int1=stoi(s1);
    int int2=stoi(s2);
    int int3=stoi(s3);
    int int4=stoi(s4);
    int int5=stoi(s5);

    system("pause");
    return 0;
}
```

برای استفاده از نوع string و همچنین تابع stoi

خطا
۱۲۵
۱۲
۱۲۳
خطا

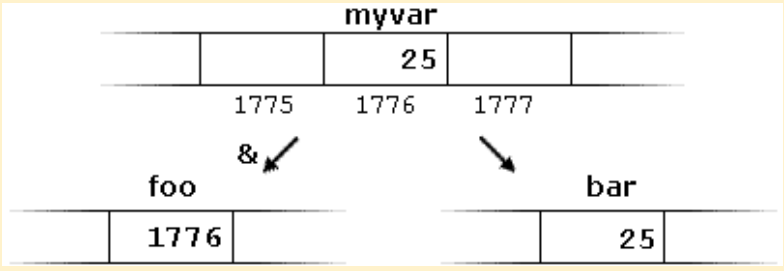
1_string_to_numeric.cpp

اشاره گرها
Address-of operator (&)

✓ آدرس یک متغیر با گذاشتن علامت & قبل از آن متغیر قابل دسترس است

```
foo = &myvar;
```

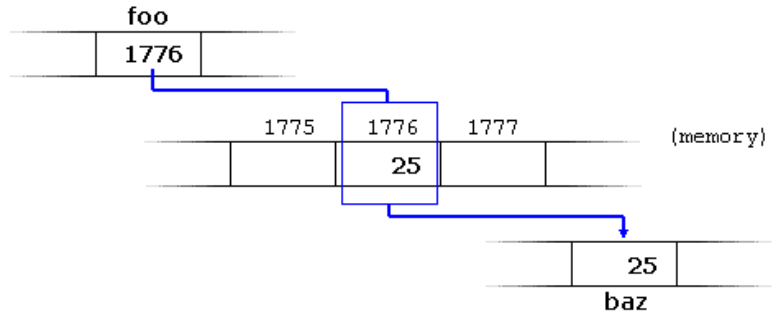
```
myvar = 25;
foo = &myvar;
bar = myvar;
```



اشاره گرها
Dereference operator (*)

✓ با گذاشتن علامت * قبل از متغیری که شماره یا آدرس خانه‌ای از حافظه در آن ذخیره شده است، مقدار آن خانه‌ی حافظه را می‌توانیم بخوانیم.

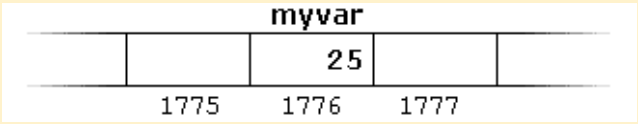
```
baz = *foo;
```



- & is the *address-of operator*, and can be read simply as "address of"
- * is the *dereference operator*, and can be read as "value pointed to by"

سوال:

```
myvar = 25;
foo = &myvar;
```



اگر متغیر myvar را با عدد ۲۵ مقداردهی کنیم، و این متغیر مطابق شکل زیر در خانه‌ی 1776 از حافظه ذخیره شود، بعد از اجرای عبارت foo=&myvar; هر کدام از موارد زیر چه عددی را برمیگرداند؟

- myvar → 25
- &myvar → 1776
- foo → 1776
- *foo → 25

Declaring pointers – تعریف اشاره گرها

```
// my first pointer
#include <iostream>
using namespace std;

int main ()
{
  int firstvalue, secondvalue;
  int * mypointer;

  mypointer = &firstvalue;
  *mypointer = 10;
  mypointer = &secondvalue;
  *mypointer = 20;
  cout << "firstvalue is " << firstvalue << '\n';
  cout << "secondvalue is " << secondvalue << '\n';
  return 0;
}
```

2_pointer.cpp

```cpp
// more pointers
#include <iostream>
using namespace std;

int main ()
{
  int firstvalue = 5, secondvalue = 15;
  int * p1, * p2;

  p1 = &firstvalue;  // p1 = address of firstvalue
  p2 = &secondvalue; // p2 = address of secondvalue
  *p1 = 10;          // value pointed to by p1 = 10
  *p2 = *p1;         // value pointed to by p2 = value pointed to by p1
  p1 = p2;           // p1 = p2 (value of pointer is copied)
  *p1 = 20;          // value pointed to by p1 = 20

  cout << "firstvalue is " << firstvalue << '\n';
  cout << "secondvalue is " << secondvalue << '\n';
  return 0;
}
```

3_pointer-2.cpp