University of Hertfordshire

7COM1025
Programming for Software Engineers
Lecture 11

Dr. Renato C. de Amorim

THE ONE HUNDRED UNDER FIFTY

PROBLEM 11.1

Using vectors, write a program that allows a user to enter as many numbers as he wants, then output the standard deviation of these numbers.

$$stdev = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

CLASSES

```
#include <iostream>
using namespace std;
class Vehicle{
    public:
    int passangers;
    int fuelcap;
    int mpg;
    int range(){
        return mpg*fuelcap; }
    };
int main(){
    Vehicle minivan;
    minivan.passangers=7;
    minivan.fuelcap=16;
    minivan.mpg=21;
    cout<<"Range: "<<minivan.range()<<endl;
    return 0;
}
```

```
#include <iostream>
using namespace std;
class Vehicle{
    public:
    int passangers;
    int fuelcap;
    int mpg;
    int range();
    };
int main(){
    Vehicle minivan;
    minivan.passangers=7;
    minivan.fuelcap=16;
    minivan.mpg=21;
    cout<<"Range: "<<minivan.range()<<endl;
    return 0;
}
int Vehicle::range(){
    return mpg*fuelcap;
}
```

CONSTRUCTOR AND DESTRUCTOR

```
#include <iostream>
using namespace std;
class MyClass{
    public:
        int x;
        MyClass(int i);
        ~MyClass();
    };
MyClass::MyClass(int i){
    x=i;
}
MyClass::~MyClass(){
    cout<<"Destructing... "<<x<<endl;
}
int main()
{
    MyClass ob1(5);
    MyClass ob2(10);
    cout<<ob1.x<<" "<<ob2.x<<endl;
    return 0;
}
```

They are normally specified under public.

A Destructor cannot have parameters

No return type

INLINE FUNCTIONS

A function is expanded inline at the point it is invoked, instead of being called.

Where the inline function is large, the overall size of your program will also increase.

- Best inline functions are those that are small.

inline is a request, not a command.

- Some compilers won't generate inline code if a function contains a loop, switch or a goto.
- Often you can't have inline recursive functions.
- Static variables are frequently disallowed

Inline restrictions are implementation-dependent.

INLINE FUNCTIONS (EXAMPLE)

```
#include <iostream>
using namespace std;
class cl{
    int i;
public:
    int get_i();
    void put_i(int j);
    };
inline int cl::get_i(){
    return i;
}
inline void cl::put_i(int j){
    i=j;
}
int main(){
    cl s;
    s.put_i(10);
    cout<<s.get_i();
    return 0;
}
```

```
#include <iostream>
using namespace std;
class cl{
    int i;
public:
    //automatic inline functions
    int get_i() {return i;}
    void put_i(int j){i=j;}
    };
int main(){
    cl s;
    s.put_i(10);
    cout<<s.get_i();
    return 0;
}
```

PROBLEM 11.2

Create a class MathContainer. It should have methods to:

- Allow the user to include a new number to the container (integer).
- Return the average of the numbers.
- Return the standard deviation of the numbers.
- Return only the even numbers in the container.

ARRAYS OF OBJECTS

```
#include <iostream>
using namespace std;
class MyClass{
    int x;
public:
    void set_x(int i){x=i;}
    int get_x(){return x;}
    };
int main(){
    MyClass obs[4];
    int i;
    for (i=0; i<4;i++)
        obs[i].set_x(i);
    for (i=0;i<4;i++)
        cout<<"obs["<<i<<"].get_x(): "<<obs[i].get_x()<<endl;
    return 0;
}
```

INITIALIZING OBJECT ARRAYS

```
#include <iostream>
using namespace std;
class MyClass{
    int x;
public:
    MyClass(int i){x=i;}
    int get_x(){return x;}
};
int main()
{
    MyClass obs[4]={-1,-2,-3,-4};
    for (register int i=0; i<4; i++)
        cout<<"obs["<<i<<"].get_x(): "<<obs[i].get_x()<<endl;
    return 0;
}
```

INITIALIZING OBJECT ARRAYS

- Two parameters

```
#include <iostream>
using namespace std;
class MyClass{
    int x;
    int y;
public:
    MyClass(int i, int b){x=i;y=b;}
    int get_x(){return x;}
};
int main()
{
    MyClass obs[4]={MyClass(-1,1),MyClass(-2,2),MyClass(-3,3),MyClass(-4,4)};
    for (register int i=0; i<4; i++)
        cout<<"obs["<<i<<"].get_x(): "<<obs[i].get_x()<<endl;
    return 0;
}
```

POINTERS TO OBJECTS

```
#include <iostream>
using namespace std;
class P_example{
    int num;
public:
    void set_num(int val) {num=val;}
    void show_num(){cout<<num<<endl;}
};
int main(){
    P_example ob, *p;
    ob.set_num(1);
    ob.show_num();
    p=&ob;
    p->set_num(20);
    p->show_num();
    return 0;
}
```

POINTERS TO ARRAYS OF OBJECTS

```
#include <iostream>
using namespace std;
class P_example{
    int num;
    static int HowMany;
public:
    P_example(){HowMany++;}
    void set_num(int val) {num=val;}
    void show_num(){cout<<num<<endl;}
    int get_HowMany(){return HowMany;}
};
int P_example::HowMany=0; //note the int!
```

```
int main(){
    P_example ob[10];
    P_example *p=ob;
    int i;
    for (i=0;i<p->get_HowMany();i++,p++)
        p->set_num(i);
    p-=p->get_HowMany();
    for (i=0; i<p->get_HowMany();i++,p++)
        p->show_num();
    return 0;
}
```