"Ss. Cyril and Methodius" University in Skopje
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Lesson 11

Virtual destructor
Example problems

Object oriented programming

Virtual destructor

When and why we need virtual destructor?

```
#include <iostream>
using namespace std;
class Base {
public:
    Base() {
        cout << "Constructing object from Base\n";
    }
    // the destructor
    ~Base() {
        cout << "Destroying object from Base\n";
    }
};
class Derived : public Base {
public:
    Derived() {
        cout << "Constructing object from Derived\n";
    }            ~Derived() {
        cout << "Destroying object from Derived\n";
    }
};
int main() {
    Base *basePointer = new Derived();
    delete basePointer;
}
```

Virtual destructor

When and why we need virtual destructor?

```
#include <iostream>
using namespace std;
class Base {
public:
    Base() {
        cout << "Constructing object from Base\n";
    }
    // the destructor
    virtual ~Base() {
        cout << "Destroying object from Base\n";
    }
};
class Derived : public Base {
public:
    Derived() {
        cout << "Constructing object from Derived\n";
    }           ~Derived() {
        cout << "Destroying object from Derived\n";
    }
};
int main() {
    Base *basePointer = new Derived();
    delete basePointer;
}
```

Problem 1
1/2

Create a hierarchy of classes that represent music and painting work. For each art work we have:

- year (int)
- author (dynamically allocated char array)
- price (float) - should not be negative (Exception)

For each music work we have additionally the genre (char array), and for each painting work we have additionally the technique (char array) and percent of damage (int).

Problem 1
2/2

For each object of the classes should be implemented the following methods:

- `float price()`
  - the initial price of the music work is increased for x% if it dates before the 17th century. The value x can be changed but it's save for all objects.
  - the initial price of the painting work is decreased according to the damage percent.
- operator $>$ - compares by their price
- opearotr $<<$ - outputs the author, year and price

Problem 1
Solution 1/3

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

class ArtWork {
protected:
    int year;
    char* author;
    float price;
public:
    ArtWork(const int year = 0, const char *author = "", const float price = 0)
         {
        this->year = year;
        this->author = new char[strlen(author) + 1];
        strcpy(this->author, author);
        if(price < 0) {
            throw 1;
        }
        this->price = price;
    }

    ArtWork(const ArtWork &a) {
        year = a.year;
        author = new char[strlen(a.author) + 1];
        strcpy(author, a.author);
        price = a.price;
    }
```

Problem 1
Solution 2/3

```
    ArtWork& operator=(const ArtWork &a) {
        if(this == &a) return *this;
        year = a.year;
        delete [] author;
        author = new char[strlen(a.author) + 1];
        strcpy(author, a.author);
        price = a.price;
        return *this;
    }

    virtual float calculate_price() const = 0;

    bool operator>(ArtWork &a) {
        return this->calculate_price() > a.calculate_price();
    }

    friend ostream& operator<<(ostream& out, const ArtWork &a) {
        out << "Author: " << a.author << endl;
        out << "Year: " << a.year << endl;
        out << "Price: " << a.calculate_price() << endl;
        return out;
    }

    ~ArtWork() {
        delete [] author;
    }
};
```

Problem 1
Solution 3/5

```
class PaintWork : public ArtWork {
private:
    char technique[100];
    int damage;
public:
    PaintWork(const int year = 0, const char *author = "", const float price =
         0,
        const char* technique = "", const int damage = 0) : ArtWork(year, author
             , price) {
        strcpy(this->technique, technique);
        this->damage = damage;
    }

    float calculate_price() const {
        return price * (1 - damage / 100.0);
    }
};
```

Problem 1
Solution 4/5

```
class MusicWork : public ArtWork {
private:
    char genre[100];
    static int x;
public:
    MusicWork(const int year = 0, const char *author = "", const float price =
         0,
        const char* genre = "") : ArtWork(year, author, price) {
        strcpy(this->genre, genre);
    }

    float calculate_price() const {
        if(year < 1700) {
            return price * (1 + x / 100.0);
        }
        return price;
    }
};
```

Problem 1
Solution 5/5

```
int MusicWork::x = 10;

int main() {
    MusicWork *mw;
    PaintWork *pw;
    try{
        mw = new MusicWork(1990, "Metalica", -1500, "Heavy Metal");
        pw = new PaintWork(1650, "Leonardo DaVinci", 15000, "Oil", 8);
    } catch(int e) {
        mw = new MusicWork(1990, "Metalica", 600, "Heavy Metal");
        pw = new PaintWork(1650, "Leonardo DaVinci", 15000, "Oil", 8);
    }
    cout << *mw;
    cout << *pw;
    if(*mw > *pw) {
        cout << "Music work: " << mw->calculate_price();
    } else {
        cout << "Paint work: " << pw->calculate_price();
    }
    delete mw;
    delete pw;
    return 0;
}
```

Problem 2

Function template

Write a function templates:

- that will merge two arrays of same type
- that will print elements from array
- that will create a new array which is subarray from the original, defined with start and end.

Problem 2
Solution 1/2

```
#include <iostream>
#include <cstring>
#include <cstdio>
using namespace std;

template <typename T>
void merge_arrays(const T* a1, const int len1, const T* a2, const int len2, T**
     result) {
    *result = new T[len1 + len2];
    int i;
    for(i = 0; i < len1; ++i) {
        (*result)[i] = a1[i];
    }
    for(int j = 0; j < len2; ++j) {
        (*result)[i++] = a2[j];
    }
}

template <typename T>
void print(const T* array, const int len) {
    for(int i = 0; i < len; ++i) {
        cout << array[i];
        if(i != len - 1) {
            cout << " ";
        } else {
            cout << endl;
        }
    }
}
```

Problem 2
Solution 2/2

```
int main() {
    int a1[] = {
        1, 4, 5 ,6, 7
    };

    int a2[] = {
        5, 6, 0
    };
    int* result;
    merge_arrays(a1, 5, a2, 3, &result);
    print(result, 8);
    delete [] result;
    return 0;
}
```

Problem 3

Model a class Bicycle. For each bicycle we have model name, mass and diameter of the wheel (inches).

In bicycle competitions there are three types of races: road races, mountain races and hybrid races. Dependant of the race type there are also three types of bicycles.

Bicycles driven on road have extra info about the width of the tyres, mountain bicycles have number of suspension systems. Create a class hierarchy.

Problem 3

Exceptions

- diameter of the wheel is in range (15 - 29)
- when user tries to create object with invalid value, an object with fixed diameter 21 should be created.

Problem 3

Polymorphism

In each class implment a function `float getRangCoefficient()` that returns the coefficient of ranging the bicycle.

- for road bicycles $(2.5 * width) * 2 + diameter * 0.2$
- for mountain bicycles $number\_of\_suspensions + (27 - diameter) * 0.8$
- for hybrid is the minimum of the road and mountain bicycles

Problem 3

Static function

In the class bicycle define a field with MAX allowed mass for competition. This information is defined by the bicycle federation and it's same for all bicycles. There should be a function that will check if bicycle can compete with definition:

```
bool canCompete(Bicycle &b)
```

Problem 3
Composition

Define a class `Competitor` that has info for name and pointer to dynamically allocated bicycle he/she uses in competitions.

Problem 3
Global function

Define a global function for printing info of competitors by categories and sorted in increasing order by their coeffitient. This defines their starting position. For each competitor print the starting position, name and model of bicycle and rang coeffitient.

Problem 3

Main function

In the main function read data for all competitors of a bicycle race in Skopje. Then print the starting positions of the competitors (only those who can compete) with a maximum allowed mass of 15kg for bicycle.

Materials and Questions

Lectures, exsercises and announcements
courses.finki.ukim.mk

Source code of all examples and problems
github.com/tdelev/OOP/tree/master/latex/src

Office hours
tdelev.github.io/office/