Mechatronic Engineering

Object Oriented Programing and Software Engineering
Laboratory instruction 7
C++ introduction

AGH Kraków, 2020

Materials created for educational purposes.
Dedicated for students attending Software Engineering course.
Author would apreaciate any feedback regarding errors of any kind found in the instruction script.
Please report those to the following email address: danielt@agh.edu.pl

Spis treści

1 Constructors.

Constructor is a special (class) member function, which has the same name as the class. In the constructors body, instructions used to set the initial values of the object elements.

Examples:

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

class liczba {
    int wartosc;
public:
    liczba (int l) { wartosc = l; } // konstruktor
    void wysw() { cout << wartosc << endl; }
};

main() {
    liczba p = liczba(5);
    liczba s(7);
    cout << "piec = ";
    p.wysw();
    cout << "siedem = ";
    s.wysw();
return 0;
}
```

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

class procesor {
      int l_watkow;
      int l_rdzeni;
    public:
      procesor(int, int); //konstruktor
      void wysw_param();
};

procesor::procesor (int a, int b) {
    l_watkow = a;
    l_rdzeni = b;
}
```

```
void procesor::wysw_param() {
    cout << "\nProcesor posiada " << l_watkow << " watkow";
    cout << "\nProcesor posiada " << l_rdzeni << " rdzeni";
}

main() {
    procesor i5(4,4);
    procesor i7(8,4);

    cout << "Program przechowuje i wyswietla informacje na temat "
    << "procesorow\n";

    cout << "\n\ti5\n";

    i5.wysw_param();
    cout << endl;

    cout << "\n\ti7\n";

    i7.wysw_param();
    cout << endl;
return 0;
}
```

A constructor can be overloaded. It is used so in one class can be more than one version of a constructor (with different ammount of parameters)

Example:

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

class procesor {
     int l_watkow;
     int l_rdzeni;
   public:
     procesor (); //konstruktor bezargumentowy
     procesor(int, int); //konstruktor 2 argumentowy
     procesor (int); //konstruktor jednoargumentowy
     void wysw_param();
};
```

```
procesor::procesor (int a, int b) {
    l_watkow = a;
    l_rdzeni = b;
}

procesor::procesor() {
    l_watkow = 1;
    l_rdzeni = 1;
}

procesor::procesor(int a) {
    l_watkow = 1;
    l_rdzeni = a;
}

void procesor::wysw_param() {
    cout << "\nProcesor posiada " << l_watkow << " watkow";
    cout << "\nProcesor posiada " << l_rdzeni << " rdzeni";
}

main() {
    procesor i5(4,4);
    procesor i7(8,4);
    procesor p3;
    procesor c2d(2);

    cout << "Program przechowuje i wyswietla informacje na temat "
    << "procesorow\n";

    cout << "\n\tp3\n";
    p3.wysw_param();
    cout << endl;

    cout << "\n\tc2d\n";
    c2d.wysw_param();
    cout << endl;

    cout << "\n\ti5\n";
    i5.wysw_param();
    cout << endl;

    cout << "\n\ti7\n";
    i7.wysw_param();
```

```
    cout << endl;
return 0;
}
```

2 Destructors.

Destructor is something opposite to the constructor. It is used for class objects deletion. Destructor has the same name as the class and before its name [tilde] sign is placed. Similar to the constructor, destructor doesn't have a return type.

Destructor can be useful
– object was representing a window on a screen which should be closed during the destruction process;
– There was a need to reserve an additional memory (eg. with the use of *new*), and a destructor should release it (eq. with use of *delete*;
– we want to use an auxiliary iterating variable (constructor increases, destructor decreases its value);

Example:

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

class procesor {
      int l_watkow;
      int l_rdzeni;
    public:
      procesor (); //konstruktor bezargumentowy
      procesor(int, int); //konstruktor 2 argumentowy
      procesor (int); //konstruktor jednoargumentowy
      ~procesor(); //destruktor
      void wysw_param();
};

procesor::procesor (int a, int b) {
    l_watkow = a;
    l_rdzeni = b;
}

procesor::procesor() {
```

```
    l_watkow = 1;
    l_rdzeni = 1;
}

procesor::procesor(int a) {
    l_watkow = 1;
    l_rdzeni = a;
}

procesor::~procesor(){
   cout << "Obiekt zostal zniszczony" << endl;
}

void procesor::wysw_param() {
    cout << "\nProcesor posiada " << l_watkow << " watkow";
    cout << "\nProcesor posiada " << l_rdzeni << " rdzeni";
}

main() {
    procesor i5(4,4);
    procesor i7(8,4);
    procesor p3;
    procesor c2d(2);

    cout << "Program przechowuje i wyswietla informacje na temat "
    << "procesorow\n";

    cout << "\n\tp3\n";
    p3.wysw_param();
    cout << endl;

    cout << "\n\tc2d\n";
    c2d.wysw_param();
    cout << endl;

    cout << "\n\ti5\n";
    i5.wysw_param();
    cout << endl;

    cout << "\n\ti7\n";
    i7.wysw_param();
    cout << endl;
return 0;
```

```
}
```

Task

Based on the informations provided in this manual, please create a simple RPG caracter creation program.

Program requirements:
1. The program has options: create new character, load character.
2. The created character is a class object.
3. The character has the following statistics: strength, dexterity, endurance, intelligence, charisma; with values assigned by the constructor.
4. Once you create a new character you can save it to a new text file. The file name should be like the name of the character being created.
5. To load the character, the user should enter the name of the file in which it is stored.