Mechatronic Engineering

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

AGH Kraków, 2021

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

Table of contents

1 Constructors.

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

Examples:

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

class number {
   int value;
public:
   number (int l) { value = l; } // constructor
   void disp() { cout << value << endl; }
};

main() {
   number p = number(5);
   number s(7);
   cout << "five = ";
   p.disp();
   cout << "seven = ";
   s.disp();
return 0;
}
```

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

class processor {
     int n_thr;
     int n_cores;
   public:
     processor(int, int); //constructor
     void disp_param();
};

processor::processor (int a, int b) {
   n_thr = a;
   n_cores = b;
}
```

```
void processor::disp_param() {
   cout << "\nProcessor has " << n_thr << " threads";
   cout << "\nProcessor has " << n_cores << " cores";
}

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

   cout << "This software stores and displays the
   information about "
   << "processors\n";

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

   i5.disp_param();

   cout << endl; cout

   << "\n\ti7\n";

   i7.disp_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 number of parameters)

Example:

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

class processor {
     int n_thr;
     int n_cores;
   public:
     processor (); //no argument constructor
     processor(int, int); //2 argument constructor
     processor (int); //1 argument constructor
     void disp_param();
};
```

```
processor::processor (int a, int b) {
   n_thr = a;
   n_cores = b;
}

processor::processor() {
   n_thr = 1;
   n_cores = 1;
}

processor::processor(int a) {
   n_thr = 1;
   n_cores = a;
}

void processor::disp_param() {
   cout << "\nProcessor has " << n_thr << " threads";
   cout << "\nProcessor has " << n_cores << " cores";
}

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

   cout << "This software stores and displays the information
   about "
   << "processors\n";

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

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

   cout << "\n\ti5\n";
   i5.disp_param();
   cout << endl;
   cout << "\n\ti7\n";
   i7.disp_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 for:

- 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 processor {
     int n_thr;
     int n_cores;
   public:
     processor (); // no argument constructor
     processor(int, int); //2 argument constructor
     processor (int); //1 argument constructor
     ~processor(); //destructor
     void disp_param();
};

processor::processor (int a, int b) {
   n_thr = a;
   n_cores = b;
```

```
}

processor::processor() {
      n_thr = 1;
      n_cores = 1;
}

processor::processor(int a) {
   n_thr = 1;
   n_cores = a;
}

processor::~processor(){
  cout << "The object was deleted" << endl;
}

void processor::disp_param() {
   cout << "\nProcessor has " << n_thr << " threads";
   cout << "\nProcessor has " << n_cores << " cores";
}

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

   cout << "This software stores and displays information
   about "
   << "processors\n";

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

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

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

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

return 0;

}
```

Task

Based on the information provided in this manual, please create a simple RPG character 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.