Mechatronic Engineering

Object Oriented Programing and Software Engineering
Laboratory instruction 3
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 Inheritance.

Inheritance allows one class to inherit an element from another class. The inheritance class is called the derived class, while the class from which the element is inherited, the base class.
Why use inheritance? Assume that the classes we created for the program have the same functions or variables. Instead of writing several times the same code, it is enough to apply inheritance. It will shorten our code and make the program more transparent.

Incheritance definition:

```
class Base {
   //class body
};
class Derived : inheritance_type Base {
   //class body
};
```

There are three types of inheritance. Public, protected and private. The differences relate to the access rights to the inherited components.

Public (public) – Public base components of the base class are inherited as public, and protected components are protected;
Protected (protected) – Public components are inherited as protected and protected components are protected;
Private (private) – Private inheritance is the default (when no inheritance type is specified). Public constituents are inherited as private and protected as private;

Components that have the private attribute are always inherited as private, and the derived class does not have access to them. Access is only possible through friendly classes.
The inherited components can be used just like those defined directly in a derived class.

Example:

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

class hero {
```

```
    int str, dex, vit, int, cha;
public:
    void przyp_atr (int x) {
        str = x; dex = x; vit = x; int = x;
        cha = x;
    }

    void wysw_atr() {
        cout<<"\t\t\tHero stats:"<<endl;
        cout<<"\tStrength: "<<str<<endl;
        cout<<"\tDexterity: "<<dex<<endl;
        cout<<"\tInteligence: "<<int<<endl;
        cout<<"\tVitality: "<<vit<<endl;
        cout<<"\tCharisma: "<<cha<<endl;
    }
};

class knight : public hero {
public:
    knight (int l){
        przyp_atr(l);
        wysw_atr();
    }
};

main() {
    int b;
    do {
        if (b > 9) {
            cout<<"Learn to read! the number supposed to be "
            <<"from 0 to 9"<<endl;
        }
        cout << "Your knight is being created. "
        << "Enter a number from 0 to 9: "<<endl;
        cin>>b;
    } while (b>9);

knight r1(b);

return 0;
}
```

2 Header files.

You can use header files for better organization of the code. These are external source files where class and their members declarations are stored. This will store only main function in the main source file. Headers have *.h or * .hpp extensions. For the header file to be properly attached to the program use the directive:

```
#include "name_of_header_file"
```

Task

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

Program requirements:
1. Add option to generate a list of monsters to the program (Five monsters per program cycle).
2. Equip the program with „monsters" class, wich will inherit member elements from the main hero class (the one with the atributes).
3. Monster stats are randomly generated in the range of 1 to 10
4. Once you generate a new list of monsters, you can save it to a new text file.