Mechatronic Engineering

Object Oriented Programing and Software Engineering
Laboratory instruction 6
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 Classes.

Class is a data type that similarly to structures is composed of many data types. By creation of a class, programmer creates new data type that can be a model of realistic object.

1.1 Class definition.

```
class our_type
{
//class body
};
```

As one can see, class definition is simple. It is important to remember about adding a semicolon after the closing bracket.

1.2 Class elements.

To create a class object one has to proceed as with variable declaration:

```
our_type name_of_the_object;
```

In C++ it is possible to crate a pointer or reference in the class type:

```
our_type * ptr; // pointer

our_type any_object;
our_type &pnickname = any_object // reference
```

In the shown class definition one can find a blank space (commented as class body). Inside it the elements of the class are declared. Class elements can be any data:

```
class processor{
Public:
   int n_cores;
   float clck_sp;
   int n_thr;
   char name[80];
};
```

In order to refer to object elements one has to use the following notation:
– object.element
– pointer -> element
– reference.element

Załóżmy, że tworzymy następujące obiekty i przeprowadzamy na nich operacje:

```
processor i5;          // definition of object specimen
processor * wsk;       //definition of pointer
processor & ipiatka;  // definition of reference
```

To associate a value to element n_ cores in i5 object it is necessary to use following methods:

```
i5.n_cores = 4;

wsk = &i5;
wsk->n_cores = 4;

ifive.n_cores=4;
```

Functions also can be used as classes elements. They are called member functions. They are used to work with member data.

```
class mob_phone {
public:
   void call(int num);
   void vibrate(bool vib);
   int ph_num;
   bool vibration;
   char cont_name[80];
   char disp_call_name(char cont[]);
};
```

In the above definitione one can see, that function declarations are mixed with data declarations. This is happening because the range of validity scope for class data is covering the whole class.
It is possible to use as an element for the class elements from other classes.

1.3 Labels.

In the above example one could notice notation public. It was so called class label. Classes can have three types of labels:

– **private** – this means that elements declared in its range will be only available to use inside of the class that they are declared. This means that only member functions of this class will be able to use them;
– **public** – this means that elements declared in its range will be available inside the class and beyond it;
– **protected** – similar to private. It provides members to the classes derived from the class in which it is used.
In default if no labels will occur inside the class, members will have private labels.

1.4 Member functions.

The member function is used to perform operations on class data members. It is also the only (though not quite) tool for performing operations on private label components. The function is called for a particular object of the class.

```
object_name.name_of_member_function(argument);
```

For the following class [1]:

```
class person {
      char name[80];
      int age;
   public:
      void memo(char *, int);
      void write();
};
```

oraz obiekty:

```
person student1, student2, student3;
```

to use the member function:

```
student1.memo("Jan Kowalski", 23);
```

[1]Source: J. Grębosz, Symfonia C++

Member functions can be defined in two places:
– inside class definition;
– after declaration inside class definition outside the class body. It is necessary to add before the name of the function **name_ of_ the _ class::**
Example:

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

class processor {
      int n_thread;
      int n_core;
    public:
      int proc_param(int thr, int cores) {
         n_thread = thr;
         n_core = cores;
      }
      void disp_param();
};

void procesor::disp_param() {
    cout << "\nProcessor has " << n_thread << " threads";
    cout << "\nProcesor has " << n_core << " cores";
}

main() {
    processor i5, i7;
    cout << "Program stores and displays the processors "
    << "data\n";

    i5.proc_param(4, 4);
    i7.proc_param(8, 4);
    cout << "\n\ti5\n";

    i5.disp_param();
    cout << endl;
    cout << "\n\ti7\n";

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

Task

Based on the informations provided in this manual, please improve the simple utility interface of the snack and beverage vending machine created on previous labratories.

Program requirements:
1. Rebuild the vending machine software to object oriented program (add classes and objects).