Разработка заданий для изучения объектно-ориентированного программирования

Епрынцев, Д. В.[1], Черноусова, Е. М.[2]
[1]*dimaeprincev@gmail.com*, [2]*lena@info.sgu.ru*
Саратовский государственный университет имени Н.Г. Чернышевского»

Статья посвящена разработке курса задач на тему объектно-ориентированного программирования в школьном курсе информатики.

Ключевые слова: ООП, программирование, курс задач, объектно-ориентированное программирование.

Объектно-ориентированное программирование в настоящее время занимает ведущее место в разработке профессиональных программных средств. Ознакомление с его основами в школьном курсе информатики второго уровня представляется вполне возможным и полезным для тех учащихся, которые в будущем планируют связать свое дальнейшее обучение с программированием. Введение в объектно-ориентированное моделирование в школе может происходить с самого начала изучения информатики, так как в начальном курсе вводятся основные понятия, например, объекта и его свойств.

Актуальность выбранной темы обусловлена тем, что объектно-ориентированный подход дает много преимуществ и используется многими разработчиками, и внедрение ООП в школьный курс поможет ученикам подобраться поближе к тому, чем сегодня занимаются программисты. Для подготовки детей к жизни в современном информационном обществе, в первую очередь необходимо развивать логическое мышление, способности к анализу (вычленению структуры объекта, выявлению взаимосвязей, осознанию принципов организации) и синтезу (созданию новых схем, структур и моделей).

Для того чтобы детям было легче изучать объектно-ориентированное программирование, необходимо правильно подать материал. Необходимо создать такой курс задач, который был бы интересен для детей, связан с их увлечениями и тем, что затрагивает их каждый день.

Объектно-ориентированный подход ценен тем, что не отходит так далеко от привычной для детей картины мира. Есть какой-либо объект. У этого объекта есть свойства, которые характеризуют его. В своем курсе задач мы использовали объекты, которые в свою очередь связаны с современной культурой. Конкретно в данном курсе использовались элементы фэнтези: монстры, волшебники, маги, лучники. Все то, что, скорее всего, знакомо для большинства детей. Будем честны, читать о том, как описывается объект в современных учебниках, достаточно скучно. Например, «объект можно описать с помощью величин. Например, жилой дом можно описать с помощью величин “количество этажей”, “наличие лифта”, “материал, из которого построены стены”» [1]. Мы ни в коем случае не говорим, что это неправильная подача материала. Нам кажется, что намного интереснее будет слушать о том, какие доспехи надеты на рыцарях или какими заклинаниями владеет колдунья.

Итак, объяснение материала начинается с того, что мы вводим понятие класса для учеников. И рассматриваем его на самом простом примере.

Необходимо создать класс Unit с тремя полями: имя, урон и здоровье. Игровая модель позволяет детям охотнее проникнуться изучением предмета и данной темы в целом. Язык, на котором мы будем рассматривать основные принципы объектно-ориентированного программирования, был выбран исходя из того, чтобы код выглядел аккуратнее. Pascal не позволяет добиться такой же компактности, как язык C++.

Листинг 1. Пример программного кода

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

class Unit
{
private:
    string name; //имя
    int health = 0; //здоровье
    int damage = 0; //урон
public:
    void setName (string a) {name = a; }
    void showName() {cout << "Имя: " << name << endl;}
    void setHealth (int a) {health = a; }
    void showHealth () {cout << "Уровень здоровья: " << health << endl;}
    void setDamage (int a) {damage = a;}
    void showDamage () {cout << "Урон: " << damage << endl;}
};

int main()
{
   Unit Frodo;
   Frodo.setName("Frodo");
   Frodo.showName();
   Frodo.setHealth(15);
   Frodo.showHealth();
   Frodo.setDamage(15);
   Frodo.showDamage();
   return 0;}
```

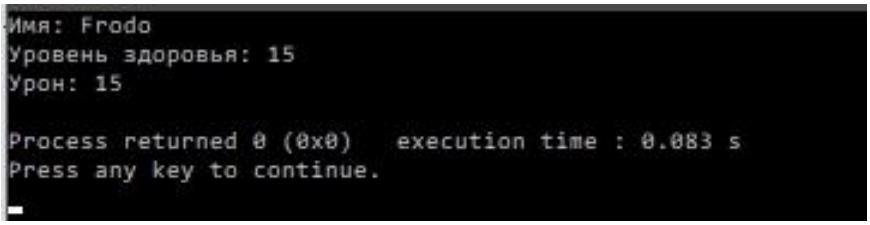

Рис. 1. Результат работы программы

В программном коде отчетливо можно заметить использования элементов вселенной, созданной Дж. Р.Р. Толкином. Все задачи выполнены в подобном стиле, дабы полностью погрузить детей в работу, но и не нагружать их скучной подачей, например, с геометрическими фигурами, которые очень сильно отбивают мотивацию изучать объектно-ориентированное программирование.

В одной из следующих наших задач мы рассмотрели сражения между двумя персонажами. Нам показалось, что написания процесса боя, хоть и достаточно примитивного, будет достаточно забавным и интересным ходом. Огненный монстр сталкивается с волшебником и, если запас маны волшебника недостаточно велик, то он погибает, а в противном случае не дает монстру пройти, выдавая, пожалуй, одну из известнейших цитат фэнтези: «You shall not pass!».

Условие самой задачи выглядит следующим образом:

Создать наследуемые от Unit классы Monster (монстр) и Mage (маг), которые будут содержать в себе одно приватное поле: damage и mana, соответственно. Также у этих классов имеются методы showDamage и showMana, которые выводят на экран уровень наносимого урона объекта класса Monster и количество магической силы персонажа класса Mage. Все остальные методы и параметры наследуются от класса Unit. Если запас магической силы объекта класса Mage меньше 25, то вывести на экран сообщение о том, что сил у героя недостаточно для того, чтобы противостоять монстру, а иначе — «YOU SHALL NOT PASS!» На рисунке 2 изображен результат работы программы.

Листинг 2. Пример программного кода

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

class Unit
{
protected:
    string name;
    int health;
public:
    Unit(): health(30){}
    Unit(int a): health(a) {}
    void setName (string a) {name = a; }
    void showName() {cout << "Имя: " << name << endl;}
    void showHealth () {cout << "Здоровье: " << health << endl;}
    int getHealth () { return health;}
};


class Monster : public Unit
{
private:
     int damage;
public:
     void showDamage(){cout << "Урон монстра: " << damage << endl;}
     int getDamage () { return damage;}
     Monster():damage(20) { }
     Monster(int a):damage(a) { }
     Monster(int a, int b):Unit(a),damage(b) { }
};
```

```
class Mage: public Unit
{
private:
    int mana;
public:
    void showMana(){cout << "Запас силы мага: " << mana << endl;}
    int getMana () { return mana;}
    Mage(): mana (25) {}
    Mage(int a): mana(a) {}
    Mage(int a, int b): Unit(a), mana(b) {}


void shallNotPass ()
{
  if (mana > 25)
  {
     cout << "YOU SHALL NOT PASS!" << endl;
  }
  else
  {
     cout << "Маг бессилен" << endl;
  }
}
};

int main()
{
    Monster Barlog(35, 20);
    Barlog.setName("Barlog");
    Barlog.showName();
    Barlog.showDamage();
    Barlog.showHealth();

    Mage Gandalf(30);
    Gandalf.setName("Gandalf");
    Gandalf.showName();
    Gandalf.showMana();
    Gandalf.showHealth();
    Gandalf.shallNotPass();

    Mage Sauron (15);
    Sauron.setName("Sauron");
    Sauron.showName();
    Sauron.showMana();
    Sauron.showHealth();
    Sauron.shallNotPass();
    cin.get();
    return 0;
}
```

Рис.2. Результат работы программы.

В качестве примера, предлагаем посмотреть на условия задач, которые представлены непосредственно в самом курсе. Первая задача является задачей, для которой не нужно использовать наследование. Вторая задача, в свою очередь, требует использования всех основных принципов ООП:

1. Определить класс Warrior, у которого есть приватные поля damage и health. Создать массив объектов данного класса и вывести на экран информацию о каждом объекте, выбрав сильнейшего (воин с самой большой силой атаки).

2. Создать класс Unit, который содержит в себе такие поля, как health и damage, и функции, позволяющие вывести на экран информацию о данном персонаже, а также определить уровень здоровья и урон, который способен нанести объект класса Unit. Создать производные классы: Monster (дополнительное поле: ярость) и Hero (дополнительное поле: храбрость). Написать метод, который будет определять убежит герой или сразится с монстром (если храбрость героя меньше 10 и ярость монстра больше 15, то герой сбегает).

В дальнейшем планируется создать полноценный электронный курс, демонстрирующий понятия: классы, объекты, наследование, инкапсуляция, полиморфизм, виртуальные функции, чистые виртуальные функции, абстрактные классы. А также содержащий набор задач для самостоятельной работы учеников.

Список литературы

[1] *Горячев А.В., Суворова Н.И., Спиридонова Т.Ю.* Информатика в играх и задачах 5-й класс. Учебное пособие, контрольные работы и тесты. Изд. 2-е, испр. — М.:Баласс, 2013. — 160 с.

[2] *Лаптев В.В.* C++. Объектно-ориентированное программирование: Учебное пособие. - СПб.: Питер, 2008. 463 с.

[3] *Stroustrup, B.* A tour of C++: 2013, 171 с.

[4] *Страуструп, Б.* Язык программирования C++: Специальное издание, 2008. 1055 с.

[5] *Черноусова Е.М.* «Изучение объектно-ориентированного программирования в школьном курсе информатики и ИКТ" Информационные технологии в образовании: Материалы VII Всерос. научно-практ. конф. – Саратов: ООО «Издательский центр «Наука»», 2015. - С. 133-138.