РАЗРАБОТКА ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРИЛОЖЕНИЯ, РЕАЛИЗУЮЩЕГО ФУНКЦИИ ЗАЩИТЫ ДАННЫХ ПОЛЬЗОВАТЕЛЕЙ

Бородина М.С., Малик А.Д.

г. Тамбов Технический колледж ФГБОУ ВПО «Тамбовский государственный технический университет»

Научный руководитель: Мосягина Н.Г., г. Тамбов, преподаватель, Технический колледж ФГОУ ВО «Тамбовский государственный технический университет»

Объектно-ориентированные языки программирования пользуются в последнее время большой популярностью среди программистов. Наиболее распространенным объектно-ориентированным языком программирования, безусловно, является C++. Свободно распространяемые коммерческие системы программирования C++ существуют практически на любой платформе.

В данной статье мы будем рассматривать описание программы, задача которой заключается в автоматизации процесса обработки персональных данных, реализации механизмов защиты данных на основе объектно-ориентированного подхода, также на примере данной программы мы рассмотрим иерархию классов, использование наследования и модификаторов доступа.

В ООП главным элементов является класс, включающий множество объектов с одинаковыми свойствами, операциями и отношениями. Класс имеет внутреннее (реализацию) и внешнее представление – интерфейс. В данной программе реализован класс Adress. Класс Adress является базовым классом программы с помощью которого описывается модель адреса. Ниже представлен код, в котором задаются такие параметры класса как Город, Улица, Дом и Индекс города.

```
class adress{
private:
    char yl[100]; //переменная улица
    char go[100]; //переменная города
    char dom[100]; //переменная дом
    int indexg; //переменная индекса города
    int pass; //переменная пароля
    int sec; //переменная отвечающая за проверку пароля
    int pop; //переменная отвечающая за кол-во попыток
    void genpass(){ //функция отвечающая за генерацию пароля
    pass=60305;  sec=0;pop=3;  }
    voidinyl(){ //функция ввода переменной улица
    cin>>yl;}
    voidingo(){ //функция ввода переменной города
    cin>>go;}
    voidindom(){ //функция ввода переменной дома
    cin>>dom;}
    voidinindex(){ //функция ввода переменной индекса
        cin>>indexg;}
```

Для работы сданными были созданы функции для ввода, вывода, удаления и изменения информации.

Приведем пример фунции для изменения данных inoindex:

```
Void inoindex(){ //функция отдельного ввода индекса
    int de;
    if (sec==0) de=secure();
    if (sec==1 || de==1) {
    indexg=NULL;
    cout<<"Индекс: ";
    in index();
    }
        else {if (pop>0)cout<<”Пароль неправильный - в доступе отказано”<<endl;
    else cout<<»Количество попыток входа превышено - в доступе отказано»<<endl;}
    }
```

Важной задачей является защита информации от несанкционированного просмотра, изменения и удаления данных. Это реализовывается с помощью модификаторов доступа private и public. Изначально, все содержимое класса является доступным для чтения и записи только для него самого. Модификатор доступа public используют для того, чтобы разрешить доступ к данным класса.

Закрытые данные класса размещаются после модификатора доступа private. Если отсутствует модификатор public, то все функции и переменные, по умолчанию являются закрытыми.

Также мы защищаем программу с помощью функции запроса пароля на изменение данных:

```
intsecure(){ //функция отвечающая за защиту изменений класса
        int l=0;
        int pad;

        cout<<"Введите пароль: ";
```

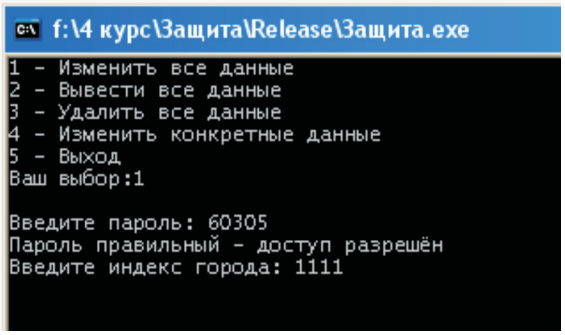
```
                cin>>pad;
                if (pass==pad && pop>0) {sec=0; l=1; cout<<"Пароль правильный – доступ разрешен"<<endl;}
                else pop=pop-1;
                return l;}
```

В файле Защита.cpp мы создали меню с обращениями к функциям классов:

```
    //объявление переменных
    int k=0;
    int n;
    int f;
    int h;
    adressco; //объявление объекта класса
    co.start();//обращение к функции класса
    //меню с обращениями к функциям классов
    while (k==0){
        fo:
        cout<<»1 - Изменить все данные»<<endl;
        cout<<»2 - Вывести все данные»<<endl;
        cout<<»3 - Удалить все данные»<<endl;
        cout<<»4 - Изменить конкретные данные»<<endl;
        cout<<»5 - Выход»<<endl;
        cout<<»Ваш выбор:»;
        cin>>n;
        cout<<endl;
        switch (n){
    case 1: {
                h=co.in();
                break;}
    case 2:{
                h=co.out();
                break;}
    case 3:{
                h=co.resetall();
                break;}

    case 4:{
        cout<<»1 - Изменить индекс»<<endl;
        cout<<»2 - Изменить город»<<endl;
        cout<<»3 - Изменить улицу»<<endl;
        cout<<»4 - Изменить дом»<<endl;
        cout<<»Ваш выбор:»;
                cin>>f;
                switch (f){
                case 1:{ co.inoindex(); goto fo;}
                case 2:{ co.inogo();goto fo;}
                case 3:{ co.inoyl();goto fo;}
                case 4:{ co.inodom();goto fo;}
    default:{cout<<»Неправильное значение меню»<<endl; goto fo;}
                }
                break;}
    case 5:{
                k=1;
                break;}
    default: {cout<<»Неправильное значение меню»<<endl;break;}
        } //switch
    cout<<endl;
    } //while
        return 0;
    }
```

В самом начале программы создается объект класса Adress. Дело в том, что сам класс является только описанием его объекта. Класс Adress является описанием любого адреса, имеющего город, улицу, дом и индекс. Все данные вводятся пользователем вручную. С помощью меню возможно выбрать необходимые действия над данными: изменение, удаление, вывод данных и изменение конкретных данных. При выборе каждого пункта меню, для защиты от несанкционированного доступа в программу, будет запрашиваться пароль, который мы ввели в базовом классе.



Ввод пароля

Данное программное изделие подойдет для любой организации у которой возникла необходимость во временном хранение определенных данных.

Список литературы

1. Шилдт Г. Самоучитель С++. – СПб. : БХВ-Петербург, 2001. – 670 с.

2. Павловская Т.А. С/С++ Программирование на языке высокого уровня. – СПб. : Питер, 2009. – 461 с. : ил.