Санкт-Петербургский государственный политехнический университет

Институт информационных технологий и управления

Кафедра «Распределенные вычисления и компьютерные сети»

К У Р С О В А Я Р А Б О Т А

Создание простейшего класса на языке C++

по дисциплине «Основы программирования»

Выполнил
студент гр.13507/1 В.Б. Борисов

Руководитель
Ст. преподаватель Т.Н.Самочадина

«___» _____ 2014 г.

Санкт-Петербург
2014

СОДЕРЖАНИЕ

Введение

Объектно-ориентированное программирование (ООП) - это результат естественной эволюции более ранних методологий программирования. Потребность в ООП связана со стремительным усложнением приложений и отсюда, как следствие, недостаточной надежностью программ и выразительными способностями языков программирования.

ООП - это моделирование объектов посредством иерархически связанных классов. Малозначащие детали объекта скрыты от нас, и если мы даем команду, например, переместить объект, то он ”знает”, как он это делает. Переход от традиционного программирования к ООП на начальном этапе характерен тем, что под объектами в программе подразумеваются конкретные физические объекты. В этом случае легче дается понимание различных действий над ними. В качестве примера можно выбрать простые графические фигуры, поскольку каждая фигура представляет реальный объект на экране. Его всегда можно отобразить и, тем самым, проверить моделируемые действия над объектом в явном виде. После того, как определены простейшие графические объекты, достаточно легко можно формировать более сложные на основе уже имеющихся. В ООП такому сложному графическому объекту соответствует список примитивных объектов, к которому применимы все те же действия, что и к составляющим его элементам: отображение, стирание с экрана, перемещение в заданном направлении. [2]

Данная курсовая работа посвящена разработке и реализации программы на языке C++ с использованием класса динамических массивов структур. Программа обеспечивает возможность создание файла и хранения в нем информации об инициалах абонента, номера телефона и тарифа, а также выполнение требуемых функции по работе с этими данными.

1. Детальная постановка задачи

Необходимо описать класс «абонент», который содержит следующие поля:

- Фамилия имя отчество
- Номер телефона
- Тариф

Помимо описания полей класса, работа должна содержать:

1. Конструктор с параметрами, без параметров и конструктор копирования
2. Дружественные функции ввода/вывода
3. Перегрузку одного из бинарных операторов как метод класса и через дружественную функцию
4. Перегрузку одного из операторов отношения как метод класса и через дружественную функцию
5. Перегрузку оператора присваивания
6. Перегрузку префиксного и постфиксного инкремента как метод класса и через дружественную функцию
7. Перегрузку операторов << и >>
8. Создание массива из объектов класса
9. Функцию, определяющую максимальный объект
10. Предусмотреть обработку и инициализацию исключительных данных

Основная часть

2. Описание разрабатываемого класса. Обоснование способа представления данных.

Класс описан в отдельном заголовочном файле «Subscriber.h».
Как уже было сказано, класс «абонент» содержит 3 поля (методы имеют открытый доступ, а свойства класса закрытый доступ), двое из которых представлены в динамической памяти. Это позволяет по ходу работы программы контролировать и корректировать объём используемой памяти и, следовательно, обрабатывать большие объёмы данных, обходя ограниченность физической памяти машины.

Выделяется память с помощью оператора «new», а освобождается — с помощью оператора «delete».

В описании класса также присутствуют различные перегруженные и дружественные функции:

1. Перегрузка бинарного оператора

Для перегрузки бинарной операции внутри класса необходимо создать функцию-метод:
(общий вид)

```
type operator symbols(type1 parametr)
{
операторы;
}
```

Здесь **type** — тип возвращаемого операцией значения, **operator** — служебное слово, **symbols** — перегруженная операция, **type1** - тип второго операнда, первым операндом является экземпляр текущего класса, **parametr** — имя переменной второго операнда.

В программе данная перегруженная функция представлена следующим образом:

```
Subscriber & Subscriber::operator +(short tariff)
{
    tariff_ += tariff;
    tariff_ %= 1000;
    return *this;
};
```

2. Перегрузка оператора присваивания

Оператор присваивания обязательно определяется в виде функции класса, потому что он неразрывно связан с объектом, находящимся слева от "=". Определение оператора присваивания в глобальном виде сделало бы возможным переопределение стандартного поведения оператора "=".

```
type operator symbols(type1 parametr)
{
операторы;
}
```

В программе данная перегруженная функция представлена следующим образом:

```
Subscriber & Subscriber::operator =(const Subscriber &ob){
if (this!=&ob)
{
    fullName_ = ob.fullName_;
    number_ = ob.number_;
    tariff_ = ob.tariff_;
}
return *this;
};
```

3. Дружественные функции

Для создания дружественной функции необходимо описать её внутри класса и приписать к функции ключевое слово «friend». В качестве параметра ей должен передаваться объект или ссылка на объект класса.

```
class A
{
public:
friend void in(A &ob);
};
```

В программе данная дружественная функция представлена следующим образом:

```
void out(const Subscriber &ob)
{
    cout << "ФИО название: " << ob.fullName_ << "\n";
    cout << "Номер телефона : " << ob.number_ << "\n";
    cout << "Тариф : " << ob.tariff_ << "\n";
};
```

3. Структура проекта

Проект реализован в компиляторе Microsoft Visual Studio 2013 на языке C++. Используемый класс описан в файле.

Subscriber.h

В классе Subscriber.h также присутствуют:

Перегруженные функции

1) Subscriber &operator +(short tariff);
2) friend Subscriber &operator -(Subscriber &ob, short tariff);
3) bool operator >(const Subscriber &ob);
4) friend bool operator <(const Subscriber &ob1, const Subscriber &ob2);
5) Subscriber & operator =(const Subscriber &ob);
6) Subscriber &operator ++(int);
7) friend Subscriber &operator --(Subscriber &ob);
8) friend const ostream& operator << (ostream & os, const Subscriber & ob);

Дружественные функции

1) friend void in(Subscriber &ob);
2) friend void out(const Subscriber &ob);

Реализация используемого класса описана в файле

Subscriber.cpp

Взаимодействие пользователя с окном ввода происходит с помощью функции

main.cpp

Основной пользовательский функционал (ввод/вывод данных) происходит с помощью основных операторов:

1) cin
2) cout

Поиск максимального объекта осуществляется в файле

maxobject.cpp

Во время работы программы действует «обработка ошибок», которая в нужный момент приостанавливает дальнейший ввод данных, это позволяет избежать неверного вывода информации. (Например: Номер телефона не может содержать знак минус или тариф не может быть меньше 0)

Пример кода:

```
try {
    if (!cin) throw "Введено не число\n";
    if (size < 0) throw "число должно быть >=0\n";
   }
    catch (char* err){
            cerr << err;
    }
```

4. Особенности реализации отдельных методов

В этом разделе мы рассмотрим методы, которые принимают участие в процессе создания программы. Большинство из реализованных в данной работе методов тривиальны и не имеют каких-либо тонкостей и особенностей, которые требовали бы дополнительной документации.

Возьмем, например: перегрузку постфиксного инкремента

Для полного понимания представляется код:

```
Subscriber & Subscriber::operator ++(int)

{

    Subscriber temp = *this;

    this -> tariff_+=100;

    if (tariff_ > 1000)

            tariff_ = 0;

    return temp;

};
```

Эта часть кода добавляет к тарифу 100 рублей. В коде присутствует указатель this, который в свою очередь содержит адрес объекта, который и увеличит свое значение на 100. Операция постфиксного инкремента и декремента должны иметь первый параметр типа «int». Он используется только для того чтобы отличить их от префиксной формы.

Рассмотрим следующую перегрузку

```
Subscriber & Subscriber::operator =(const Subscriber &ob){
if (this!=&ob)
{
    fullName_ = ob.fullName_;
    number_ = ob.number_;
    tariff_ = ob.tariff_;
}
return *this;
        };
```

Эта перегрузка присваивания. Эта операция вызывается каждый раз, когда одному существующему объекту присваивается значение другого. Чтобы сохранить значение присваивания, операция-функция должна возвращать ссылку на объект, для которого она вызвана, и принимать в качестве параметра единственный аргумент – ссылку на присваиваемый объект. Для возврата текущего объекта из функции члена используем оператор return «*this».

5. Отладка и тестирование

Требуется протестировать программу на различных тестах. Результатом программы должен быть – вывод на экран абонента с наивысшим тарифом.

При создании программы регулярно использовался «Локальный отладчик Windows» с конфигурации решения «Debug», что позволило отследить и устранить ошибки.

Тест план с результатами выполнения тестов:

	Вводимые значения	Вывод
1)	«Введите количество абонентов» -3	Сообщение: « число должно быть >=1»
2)	«Введите количество абонентов» Petrov	Сообщение: «Введено не число»
3)	«Введите количество абонентов» 2 «Введите информацию о абоненте(ах)» Абонент[0] Borisov 79217429364 1000 Абонент[1] Bondar 79818227380 1300	Информация о максимальном счете ФИО название: Bondar Номер телефона : 79818227380 Тариф : 1300

Итак, программа корректно принимает данные на вход, обрабатывает их, и выводит полностью корректный ответ.

Заключение

В заключение проведенной работы можно сделать основные выводы. Класс представляет собой главное инструментальное средство C++ для объектно-ориентированного программирования. Мы узнали, что класс очень похож на структуру, в которой сгруппированы элементы, соответствующие данным о некотором объекте, и оперирующие этими данными функции (называемые методами*).* Объект представляет собой некоторую сущность, например телефон. Класс C++ позволяет программам определять все атрибуты объекта. В случае, когда объектом является телефон, класс может содержать такие элементы данных, как номер и тип телефона, его тариф и т.д. На основе этого была написана программа, которую можно найти в (Приложении 1).

Поставленная задача решена. Алгоритм реализован на языке C++. Проделаны тесты, результаты которых приведены в таблице.

Список использованных источников

1. Павловская Т.А. C/C++. Программирование на языке высокого уровня.- СПб.: Питер, 2011 – 461с.: ил

2. Григорьев А., “Объектно-ориентированное программирование”. Сайт Петрозаводского Государственного Университета [“http://www.petrsu.ru”]

Приложения

1. Приложение 1. Текст программы (сдано в электронном формате).
2. Приложение 2. Протокол отладки (сдано в электронном формате).
3. Приложение 3. Слайды презентации (сдано в электронном формате).