

«Санкт-Петербургский государственный политехнический университет»

Институт информационных технологий и управления

Кафедра «Распределенные вычисления и компьютерные сети»

КУРСОВАЯ РАБОТА

Создание простейших классов на языке C++

По дисциплине “Основы программирования”

Выполнил студент
гр. 13507/2

Т.Ф.Сафин

Руководитель
Ст.преподаватель

Т.Н.Самочадина

Санкт-Петербург

2014

Содержание

1	Введение	3
2	Постановка задачи	4
3	Описание разрабатываемых классов. Обоснование способа представления данных	5
3.1	Класс Worker	5
3.2	Класс M_String	5
3.3	Класс Year	6
4	Структура проекта	6
5	Особенности реализации	7
6	Тестирование	8
7	Заключение	9
8	Список используемой литературы	10
9	Список приложений	10

1. 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]

В данной курсовой работе показан пример создания классов с заданной структурой, методами работы с объектами созданных классов.

2. Постановка задачи

Требуется написать программу для хранения и обработки информации о каждом сотруднике в отделе кадров. В ходе написания программы должен быть описан класс, содержащий следующие поля:

- фамилию,
- название занимаемой должности,
- год поступления на работу.

Работа так же должна содержать:

- класс для работы со строками (пользоваться встроенным классом string не разрешается),
- класс для работы с числовыми данными, причем хранение числовых данных (год поступления на работу) должно быть оптимальным,
- реализацию классов, включающую:
 - конструкторы с параметрами и без,
 - деструкторы,
 - перегрузку операторов (ввода\вывода, бинарных, унарных, операторов отношения, постфиксного и префиксного инкремента, и т.д.).

Информация должна быть представлена и храниться в виде массива объектов главного класса (Worker). Кроме того, должна быть реализована функция, определяющая “максимальный” объект, а так же требуется предусмотреть обработку и инициализацию исключительных ситуаций.

3. Описание разрабатываемых классов. Обоснование способа представления данных

3.1. Класс Worker

Для решения поставленной задачи разрабатывается класс Worker. Для корректной абстракции и понижения сложности программы, а так же более легкого чтения кода были разработаны два вспомогательных класса M_String и Year, для хранения строковых и целых (год) значений соответственно. Так, класс Worker имеет три private-поля:

- surName_ типа M_String, для хранения фамилии сотрудника,
- profession_ типа M_String для хранения названия профессии сотрудника,
- year_ типа Year для хранения года поступления сотрудника на работу.

Класс Worker имеет тривиальные методы доступа, методы изменения полей, а так же перегруженные операторы ввода\вывода.

Для более наглядного вывода данных и более понятной работы с программой было решено хранить базу данных сотрудников в файле input.txt. Соответственно, класс Worker имеет перегруженные операторы ввода\вывода для работы с файлами.

3.2. Класс M_String

Вспомогательный класс M_String разработан для работы со строковыми значениями. Идея реализации данного класса заключается в том, что строка хранится как массив символов типа char, заканчивающийся нулевым байтом ('\0'), а сам массив при этом располагается в динамической памяти. Таким образом, класс M_String имеет два private-поля:

- string_ типа char*, для хранения указателя на нулевой элемент массива,
- lenght_ типа unsigned short int, для хранения длины строки.

Класс M_String при этом имеет:

- конструкторы с параметром (в этом случае формируется строка заданной длины),
- без параметров (в этом же случае формируется строка с длиной по умолчанию),
- конструктор копирования (для корректного копирования данных, располагающихся в динамической памяти),
- методы-доступа,
- перегруженные операторы:
 - ввода\вывода,
 - присваивания,
 - отношения.

3.3. Класс Year

Вспомогательный класс Year разработан для работы с целыми значениями, которые представляют собой год поступления сотрудника на работу. Класс поддерживает временные рамки от 1900 до 2099 года.

В связи с требованием хранить числовые данные оптимально, было принято решение “упаковать” число, разбив его на две логически части (подробнее см. раздел 5), таким образом данный класс имеет единственное private-поле year_ типа unsigned char, в котором и хранится упакованное число. Так, экземпляр класса Year будет занимать в памяти 1 байт, что является минимально возможным количеством выделяемой памяти для какой-либо переменной, а оптимальность хранения числа в свою очередь будет максимальна.

Для упаковки и распаковки величин используются private-методы packAndSaveYear() и unpackAndGetYear() соответственно. Приватность данных методов обусловлена тем, что данные методы используются сугубо внутри класса, поэтому их открытие для другой части программы будет нарушать инкапсуляцию, а так же может заставить программиста задумываться о внутренней реализации класса при его использовании, что противоречит основным мотивам использования ООП и будет совершенно лишним при разработке остальной части программы. Таким образом, мы будем работать с переменными типа Year как с обычными целочисленными переменными, не задумываясь о внутренней реализации этого класса, однако при этом хранятся данные значения будут оптимально.

Кроме того, данный класс имеет конструкторы, перегруженные операторы ввода\вывода, сравнения, постфиксного и префиксного инкремента.

4. Структура проекта

Проект реализован в компиляторе Microsoft Visual Studio 2012 на языке C++. Интерфейсы используемых классов описаны в файлах:

Worker.h

M_String.h

Year.h

Реализация используемых классов описана в файлах:

Worker.cpp

M_String.cpp

Year.cpp

Взаимодействие пользователя с консолью происходит с помощью функции “меню”:

mainMenu.cpp

Основной пользовательский функционал (считывание из файла, вывод базы данных на экран, поиск самого молодого специалиста) описан в файлах:

inputFromFile.cpp

showDB.cpp

findAndShowYoungestWorker.cpp

5. Особенности реализации

Большинство из реализованных в данной работе методов тривиальны и не имеют каких-либо тонкостей и особенностей, которые требовали бы дополнительной документации.

В данном разделе будет описан алгоритм упаковки числа при использовании класса `Year`. У данного класса для хранения года используется поле `year_` типа `unsigned char`. Переменная такого типа занимает в памяти 1 байт, т.е. 8 бит. Идея хранения заключается в том, что число (год) делится на две логические части: по две цифры на каждую. Пример: [19][87]. Очевидно, что первая часть этого числа всегда равна либо 19, либо 20, поэтому, для того чтобы хранить первую часть нам достаточно одного бита (установим однозначное соответствие: 1 - это 19, 0 - это 20). Выделим для этого крайний левый (седьмой) бит поля `year_`. Вторая логическая часть числа - это всегда число от 0 до 99. Для хранения такого числа достаточно 7 бит ($2^7=128$, $99<128$). Переведя вторую логическую часть числа в двоичную систему и записав значение в соответствующие биты, получим упакованное число, которое можно однозначно распаковать и получить исходное. При такой работе с битами, будет достаточно следующих операторов:

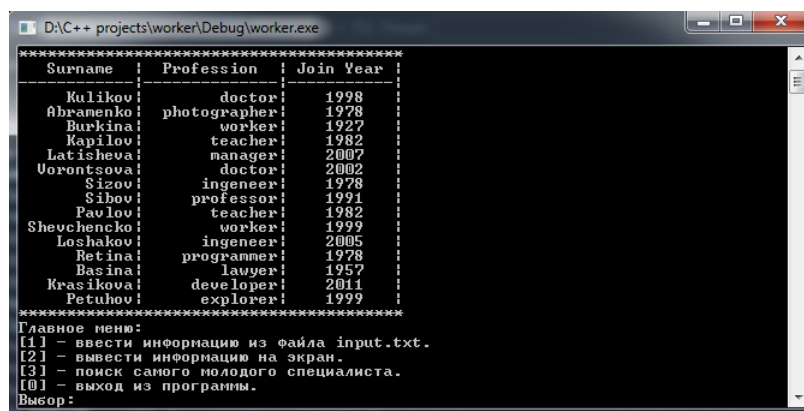
- `<<` - сдвиг влево
- `>>` - сдвиг вправо
- `~` - поразрядная инверсия
- `|` - поразрядное ИЛИ
- `&` - поразрядное И

6. Тестирование

Для тестирования программы будем использовать текстовый файл input.txt следующего содержания:

Kulikov	doctor	1998
Abramenko	photographer	1978
Burkina	worker	1927
Kapilov	teacher	1982
Latisheva	manager	2007
Vorontsova	doctor	2002
Sizov	ingeneer	1978
Sibov	professor	1991
Pavlov	teacher	1982
Shevchencko	worker	1999
Loshakov	ingeneer	2005
Retina	programmer	1978
Basina	lawyer	1957
Krasikova	developer	2011
Petuhov	explorer	1999

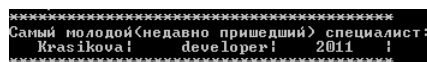
Теперь иницируем ввод базы в память из файла и попробуем вывести на экран:



```
*****
Surname      Profession  Join Year
-----
Kulikov      doctor      1998
Abramenko    photographer 1978
Burkina      worker      1927
Kapilov      teacher     1982
Latisheva    manager     2007
Vorontsova   doctor      2002
Sizov        ingeneer    1978
Sibov        professor   1991
Pavlov       teacher     1982
Shevchencko  worker      1999
Loshakov     ingeneer    2005
Retina       programmer   1978
Basina       lawyer      1957
Krasikova    developer   2011
Petuhov      explorer    1999
*****
Главное меню:
[1] - ввести информацию из файла input.txt.
[2] - вывести информацию на экран.
[3] - поиск самого молодого специалиста.
[0] - выход из программы.
Выбор:
```

Рис. 1.

Проверим поиск самого молодого специалиста по базе:



```
*****
Самый молодой(недавно пришедший) специалист:
Krasikova! developer! 2011 !
*****
```

Рис. 2.

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

7. Заключение

В данной курсовой работе применяется изученный материал об объектно-ориентированном программировании (ООП). Разрабатываемая программа становится понятнее, а ее сложность понижается, благодаря тому, что по ходу программирования с использованием ООП создаются абстракции объектов и отпадает надобность держать в голове сразу несколько вещей - достаточно думать об объекте "в целом". Итогом данной работы являются разработанные средства для управления информацией о рабочих и ее наглядной демонстрацией.

8. Список используемой литературы

1. Войнов П., “Учебный курс. Как работать с битами. Макроопределения”. Ресурс IT- тематики . [”<http://chipenable.ru/index.php/programming-avr/item/4-uchebnyy-kurs-kak-rabotat-s-bitami-makroopredeleniya.html>”]
2. Григорьев А., “Объектно-ориентированное программирование”. Сайт Петрозаводского Государственного Университета. [”http://www.petsu.ru/Chairs/IMO/pascal/theory/part2_3_1”]
3. Павловская Т.А., С\С++. Программирование на языке высокого уровня. СПб.: Питер, 2007 - 464с.

9. Список приложений

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).