Государственное бюджетное образовательное учреждение  
высшего образования Московской области  
«Университет «Дубна»  
Филиал «Котельники»

Кафедра «Информационные технологии в управлении»

КУРСОВАЯ РАБОТА

По дисциплине: Объектно-ориентированное программирование

Тема: Создание класса “класс «Дата»”, «Результаты сессии», функция поиска произвольного класса

Ф.И.О. студента Кукушкин Никита Павлович

Группа   ИВТ-21  Направление подготовки «Информатика и вычислительная техника»

Руководитель: к.ф.-м.н., доцент, Орлов Н.Н.    /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /

(ученая степень, должность, ФИО) (подпись)

Выполнил: студент(ка)

          ИВТ-21             группы

                2                    курса

          Кукушкин Н.П.

(Фамилия И.О. студента)

Дата защиты:      24.12.2019

Оценка:

Подпись

(подпись руководителя)

Котельники, 2019

Государственное бюджетное образовательное учреждение  
высшего образования Московской области  
«Университет «Дубна»  
Филиал «Котельники»

Кафедра «Информационные технологии в управлении»

УТВЕРЖДАЮ

Заведующий кафедрой

                                      /  Орлов Н.Н.   /

(подпись) (ФИО)

ЗАДАНИЕ

на курсовую работу (проект)

Тема: Создание класса “класс «Дата»”, «Результаты сессии», функция поиска произвольного класса

По дисциплине: Объектно-ориентированное программирование

Ф.И.О. студента Кукушкин Никита Павлович

Группа   ИВТ-21   Направление подготовки «Информатика и вычислительная техника»

Профильная направленность образовательной программы «Технология разработки

программного обеспечения

Дата выдачи задания    10.09.2019

Дата завершения курсовой работы    24.12.2019

Задание выдал \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /

подпись ФИО

Задание принял \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /

подпись ФИО

Оглавление

[Введение 4](#_Toc27510567)

[1 Создание класса “Прямая линия в пространстве” 5](#_Toc27510568)

[1.1 Теоретическое обоснование 5](#_Toc27510569)

[1.2 Решение задачи 6](#_Toc27510570)

[1.3 Результаты работы программы 11](#_Toc27510571)

[2 Создание класса “Школьные классы” 12](#_Toc27510572)

[2.1 Теоретическое обоснование 12](#_Toc27510573)

[2.2 Решение задачи 13](#_Toc27510574)

[2.3 Результаты работы программы 16](#_Toc27510575)

[3 Создание стека из элементов произвольного класса объектов 17](#_Toc27510576)

[3.1 Теоретическое обоснование 17](#_Toc27510577)

[3.2 Решение задачи 18](#_Toc27510578)

[3.3 Результаты работы программы 22](#_Toc27510579)

[Список литературы 24](#_Toc27510580)

# Введение

В настоящее время, компьютерные технологии, стремительными темпами шагают по древу развития человечества. В настоящее время мощность компактных мобильных телефонов, которыми обладает практически каждый человек в нашем мире, превышает всю мощность сверхдержав 1970-х годов. Фактически, рядом с нами, благодаря развитию технологий, находится переносная мировая библиотека, содержащая все знания человечества

В современном постиндустриальном мире для функционирования различных вещей используются языки программирования, которые позволяют написать программу, выполняющую определённую функцию. История программирования богата различными языками, которые используются и сейчас в различных компаниях: С#, C++, Java, Python, Go и т.д. В данной курсовой работе речь пойдет, наверное, об одном из самых известных и знаковых языков программирования-C++.

Итак, перейдём непосредственно к целям и задачам. Целью курсовой работы по дисциплине “Объектно-ориентированное программирование” является выполнение различных задач, которые включают в себя базу теоретических знании по языку программирования С++.

Главными задачами курсовой работы являются:

I) Работа с классами, включающая в себя теорию из таких дисциплин как “Линейная алгебра” и “Объектно-ориентированное программирование”.

II) Работа с наследованием, включающая важные теоретические знания из дисциплины “Объектно-ориентированное программирование”.

III) Работа с шаблонами, включающая все те же теоретические знания из дисциплины “Объектно-ориентированное программирование”.

# 1 Создание класса “Дата”

**1.1 Теоретическое обоснование**

Перед решением задачи необходимо понять, что же представляет собой “Классы” в языке C++.

Класс - это общая схема или шаблон, на основе которого создаются объекты. Класс содержит описание полей и методов, которые называются членами класса. Поле класса аналогично переменной, а метод- аналог функции. При создании на основе класса объектов, последние получают персональный набор полей и методов- строго в соответствии с описанием класса. Поля объектов имеют значения, а методы выполняют некоторые действия и возвращают результат. Фактически, поля и методы- это соответственно переменные и функции, но только строго «прикрепленные» к определенному объекту. Разные объекты, созданные на основе одного класса, имеют одинаковые наборы полей и методов, при этом значения полей у каждого объекта свои, а методы имеют автоматический доступ к объектам, из которых вызываются. Нет смысла говорить о значении поля или результате вызова метода, если не идентифицирован объект, к которому они принадлежат.

* 1. **Решение задачи**

Необходимо описать класс «Дата». В процессе выполнения этой задачи, нужно предусмотреть конструктор создания объектов этого класса и методы вычисления.

Главная проблема, возникающая при создании данного класса, заключается в том, чтоб вычислить количество дней в месяце. К примеру, если в одном месяце может быть 30 дней, то в следующем уже 31.

В моем варианте, была создана функция, что математически определяла количество дней в заданном месяце.

Сначала бегло освежим в памяти два жизненно необходимых в решении этой задачи оператора: целочисленное деление и остаток от деления.  
  
**Целочисленное деление.** Пример 16/10 = 1  
**Остаток от деления**. Пример 16%10 = 6

Заметим, что остаток от деления имеет равный с делением приоритет.

Итак, применим наш математический аппарат для получения базовой формулы. В обычном месяце 30 или 31 день, так что мы можем использовать для получения поочерёдно 1 или 0, а затем просто прибавить к этому числу константу:

https://habrastorage.org/getpro/habr/post_images/a6c/98b/e16/a6c98be164bb8e5c145849483a5c8aec.png

Получаем таблицу, полужирным выделены корректные значения:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***x*** | **1** | 2 | **3** | **4** | **5** | **6** | **7** | 8 | 9 | 10 | 11 | 12 |
| ***f(x)*** | **31** | 30 | **31** | **30** | **31** | **30** | **31** | 30 | 31 | 30 | 31 | 30 |

Уже есть правильные значения для января и для месяцев с марта по июль включительно. Февраль — особый случай, и с ним мы разберёмся чуть позже. После июля для оставшихся месяцев порядок получения 0 и 1 должен быть изменён на обратный.

Для этого мы может прибавить к делимому 1:

https://habrastorage.org/getpro/habr/post_images/66f/9a2/4f8/66f9a24f8d1be71dda85da88222662ce.png

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***x*** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | **8** | **9** | **10** | **11** | **12** |
| ***f(x)*** | 30 | 31 | 30 | 31 | 30 | 31 | 30 | **31** | **30** | **31** | **30** | **31** |

В итоге получаем такую таблицу:

Теперь правильные значения с августа по декабрь, но, как и предполагалось, значения для прочих месяцев неверны. Давайте посмотрим, как мы можем объединить эти формулы.  
 С помощью маски, используя в делимом X+1 выражение x/8 вместо 1, мы можем заменить порядок получения 0 и 1 формуле на обратный:

https://habrastorage.org/getpro/habr/post_images/f78/2a3/9a3/f782a39a33f0b883d239003211eca9d0.png

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***x*** | **1** | 2 | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |
| ***f(x)*** | **31** | 30 | **31** | **30** | **31** | **30** | **31** | **31** | **30** | **31** | **30** | **31** |

Всё правильно, кроме февраля.

Разберемся с февралем. В любом месяце 30 или 31 день, кроме февраля с его 28 (високосный год выходит за рамки этой задачи). На текущий момент по нашей формуле в нём 30 дней, поэтому неплохо бы вычесть выражение, равное 2 при x=2. Лучшее что мне удалось придумать это 2 mod x, что накладывает маску на все месяцы после февраля:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***x*** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| **2 mod *x*** | 0 | 0 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |

Изменив базовую константу на 28 с добавлением 2 к остальным месяцам получим формулу:

https://habrastorage.org/getpro/habr/post_images/2f0/4de/ff1/2f04deff152bf89e85dffb9649b77ad0.png

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***x*** | 1 | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |
| ***f(x)*** | 29 | **28** | **31** | **30** | **31** | **30** | **31** | **31** | **30** | **31** | **30** | **31** |

К сожалению, январь теперь короче на 2 дня. Но, к счастью, получить выражение, которое будет применяться только для первого месяца очень легко: это округлённое вниз обратное к x число. Умножив его на 2 получаем окончательную формулу:

https://habrastorage.org/getpro/habr/post_images/3d8/a16/999/3d8a169993f90cd39ee7b551380bf8f5.png

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***x*** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |
| ***f(x)*** | **31** | **28** | **31** | **30** | **31** | **30** | **31** | **31** | **30** | **31** | **30** | **31** |

В итоге у меня вышла вот такая функция, для определения количества дней в месяце:

**return 28 + (x + x / 8) % 2 + 2 % x + 1 / x \* 2;**

**Код программы, реализующей класс “Дата”:**

#include <iostream>

#include <string>

using namespace std;

class date

{

public:

    date()

    {

        Day = 0;

        Month = 0;

        Years = 0;

    }

    date(int Day, int Month, int Years)

    {

        this->Day = Day;

        this->Month = Month;

        this->Years = Years;

    }

    void PlussDay(int day)

    {

        Day += day;

    }

    int daysgone(int day, int month, int years)

    {

        int daysgone = 0;

        while ( years != Years )

        {

            if( years < Years )

            {

                daysgone += 365;

                years++;

                continue;

            }

            if( years > Years )

            {

                daysgone -= 365;

                years--;

                continue;

            }

        }

        while( month != Month )

        {

            if( month < Month )

            {

                daysgone += DayINmonth(month);

                month++;

                continue;

            }

            if( month > Month )

            {

                daysgone -= DayINmonth(month);

                month--;

                continue;

            }

        }

        while( day != Day )

        {

            if( day < Day)

            {

                daysgone++;

                day++;

                continue;

            }

            if( day > Day )

            {

                daysgone--;

                day--;

                continue;

            }

        }

        return daysgone;

    }

    date operator + (date& other)

    {

        this->Day += other.Day;

        this->Month += other.Month;

        this->Years += other.Years;

        return \*this;

    }

    date operator - (date& other)

    {

        this->Day -= other.Day;

        this->Month -= other.Month;

        this->Years -= other.Years;

        return \*this;

    }

    date operator = (date& other)

    {

        this->Day = other.Day;

        this->Month = other.Month;

        this->Years = other.Years;

        return \*this;

    }

    void show()

    {

        cout << Day << ":" << Month << ":" << Years;

    }

    int DayINmonth( int x )

    {

        return 28 + (x + x / 8) % 2 + 2 % x + 1 / x \* 2;

    }

private:

    int Day, Month, Years;

};

int main()

{

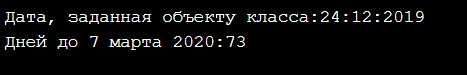
    date day(24, 12, 2019), newday(23, 9, 3041);

   cout << "\n Дата, заданная объекту класса:"; day.show(); cout << endl;

cout << " Дней до 7 марта 2020:" << day.daysgone(7, 3, 2020) << endl;    return 0;

}

* 1. **Результаты работы программы**



**В результате работы представлена работа функции счета**

# 2 Создания приложения «Результаты сессии»

## 2.1 Теоретическое обоснование

Прежде чем приступать к разбору задачи, необходимо понять, что же представляет собой наследование? Наследование- это механизм, позволяющий создавать новые классы на основе уже существующих. Класс, служащий основой для создания другого класса, называется базовым. Класс, который создается на основе базового класса, называется производным классом. Чтобы создать производный класс на основе базового класса в описании производного класса после его имени указывается базовый класс. Между именем производного и базового класса ставится двоеточие и идентификатор определяющий тип наследования. В качестве последнего могут использоваться ключевые слова public(открытое наследование),private(закрытое наследование) и protected(защищенное наследование). Независимо от типа наследования, в производном классе закрытые члены базового класса не наследуется.

## 2.2 Решение задачи

Для создания приложения «Школьные классы» необходимо разработать систему связанных классов, на основании которых создать объекты, согласно представленным ниже таблицам:

**Работники школы:**

|  |  |  |  |
| --- | --- | --- | --- |
| Табельный № | ФИО учителя | Должность /профессия | Пол |
| 100[[1]](#footnote-1) | Петров Станислав Васильевич | Учитель | М |
| 101 | Петрова Валентина Григорьевна | Учитель | Ж |
| 102 | Рыбакова Анна Ивановна | Завуч | Ж |
| 103 | Федоров Юрий Васильевич | Директор | М |
| 104 | Смирнов Антон Юрьевич | Учитель | М |

**Классы:**

|  |  |  |  |
| --- | --- | --- | --- |
| № класса | Табельный № учителя | Специализация класса | Классная комната |
| 10А | 101 | Иностранные языки (английский и немецкий) | 203 |
| 11Б | 100 | Математика и физика | 212 |

**Школьники:**

|  |  |  |  |
| --- | --- | --- | --- |
| № класса | Порядковый № в журнале | ФИО школьника | Пол |
| 10А | 1 | Иванов Сергей Петрович | М |
| 10А | 2 | Костин Петр Васильевич | М |
| 10А | 3 | Матросова Елена Ивановна | Ж |
| 11Б | 1 | Богданов Юрий Сергеевич | М |
| 11Б | 2 | Потапова Юлия Петровна | Ж |
| 11Б | 3 | Сорокина Ольга Петровна | Ж |
| 11Б | 4 | Сидоров Андрей Петрович | М |

**Листинг программы, реализующей класс “Школьные классы”:**

## 2.3 Результаты работы программы

Программа готова. Осталось понять одно: получилось ли то, что требовалось изначально в поставленной задаче.

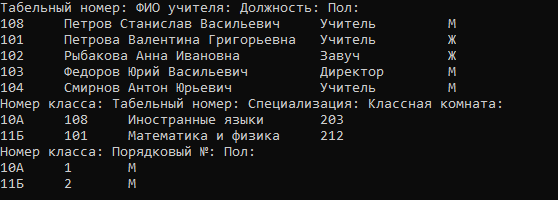


Рисунок 2. Результаты работы программы

Таким образом, реализация таблицы “Школьные классы” проведена успешно.

Вывод: в разделе “Наследование классов” были рассмотрена основная теоретическая часть и реализована программа, содержащая взаимосвязанные таблицы с необходимой информацией.

# 3 Создание стека из элементов произвольного класса объектов

## 3.1 Теоретическое обоснование

Мы достаточно подробно рассмотрели такие темы как “Классы” и “Наследование классов”, но, помимо этого, есть еще одна немаловажная тема под названием “Шаблоны в классах”. Что же такое шаблоны? Шаблоны- это универсальный инструмент языка C++, предназначенный для кодирования обобщенных алгоритмов. Основная идея шаблонов (обобщенных функции) состоит в том, что в них тип данных выступает параметром. Описание шаблонов, как правило, начинается с ключевого слова template, после которого в угловых скобках указывается ключевое поле class и формальное обозначение для типа данных (параметр типа или обобщенный параметр). Во всем остальном шаблон описывается как обычная функция, с поправкой на то, что в описании функции в качестве идентификатора типа можно использовать параметр.

## 3.2 Решение задачи

После рассмотрения теории по шаблонам, приступаем к реализации программы. Условия задачи звучат следующим образом: необходимо создать стек из элементов произвольного класса объектов, функции подсчитывающей количество объектов, обладающих некоторым свойством. Как нам известно, существует библиотека для работы со стеками, упрощающая работу с программой, но мы пойдем другим путем. Для создания шаблонов возьмем за основу 2 класса, один из которых посвящен студентам, а другой-животным.

**Листинг программы, реализующей стек из элементов произвольного класса объектов:**

#include <string>

#include <iostream>

#include <Windows.h>

#define NULL '\0'//определение null в символьном контексте

using namespace std;

//Создаем класс студентов

class student {

//Объявляем модификатор доступа типа private

private:

//Объявляем необходимые переменные

int number;

string name;

public:

//Создаем конструкторы одной переменной

void SetNumber(int number)

{

this->number = number;

}

void SetName(string name)

{

this->name = name;

}

//Создаем конструкторы двух переменных

student(int number, string name) {

this->number = number;

this->name = name;

}

//конструктор с нулевыми параметрами

student() {

this->number = 0;

this->name = NULL;

}

//Создаем функцию вывода

void show() {

cout << "Имя: " << name << "\t Номер: " << number << endl;

}

//Функция для возвращения переменной number

int GetNumber() {

return number;

}

//Функция для возвращения переменной name

string GetName() {

return name;

}

};

//Создаем класс животных

class animals {

//Объявляем модификатор доступа типа private

private:

//Объявляем необходимые переменные

string family;

string name;

public:

//Создаем конструкторы одной переменной

void SetFamily(string family)

{

this->family = family;

}

void SetName(string name)

{

this->name = name;

}

//Создаем конструкторы двух переменных

animals(string family, string name) {

this->family = family;

this->name = name;

}

//конструктор с нулевыми параметрами

animals() {

this->family = NULL;

this->name = NULL;

}

//Создаем функцию вывода

void show() {

cout << "Владелец: " << family << "\t Кличка собаки: " << name << endl;

}

//функция для возвращения значения family

string GetFamily() {

return family;

}

//функция для возвращения значения name

string GetName() {

return name;

}

};

//Создаем шаблон для организации структуры

template <class T>class struct1 {

private:

T s1;

struct1\* next;

public:

struct1(T s, struct1\* d) {

next = d;

s1 = s;

}

T getT() { return s1; }

struct1\* getstruct1() { return next; }

};

//Создаем шаблон для реализации списка с информацией о студентах

template <class T> class spisok1 {

private:

T\* steck1;

int current1;

int hight1;

public:

spisok1(int hight1) {

steck1 = new T[hight1];

current1 = 0;

this->hight1 = hight1;

}

//Функция для проверки

void put(T s1) {

if (current1 != hight1) {

steck1[current1++] = s1;

}

else cout << "Ошибка!\n\n";

}

T get() {

if (current1 > 0) {

T t = T(steck1[--current1]);

return t;}

else {

cout << "Ошибка!\n\n";

return student();

}

}

};

//Создаем шаблон для реализации списка с информацией о животных

template <class T> class spisok2 {

private:

T\* steck2;

int current2;

int hight2;

public:

spisok2(int hight2) {

steck2 = new T[hight2];

current2 = 0;

this->hight2 = hight2;

}

//Функция для проверки

void put(T s2) {

if (current2 != hight2) {

steck2[current2++] = s2;

}

else cout << "Ошибка!\n\n";

}

T get() {

if (current2 > 0) {

T t = T(steck2[--current2]);

return t;

}

else {

cout << "Ошибка!\n\n";

return animals();

}

}

};

//Главная функция программы

int main() {

setlocale(LC\_ALL, "Russian");

spisok1<student> students(2);

//Создание объектов класса students

students.put(student(3, "Иванов Иван"));

students.put(student(10, "Сидорова Юля"));

students.put(student(2, "Петрова Павла"));

student p = students.get();

//отображение значении

p.show();

p = students.get();

p.show();

p = students.get();

p.show();

spisok2<animals> dog(1);

//Создание объектов класса animals

dog.put(animals("Иванов Иван", "Дружок"));

dog.put(animals("Сидорова Юля", "Муска"));

animals t = dog.get();

t.show();

t = dog.get();

t.show();

system("pause");//задержка консольного окна

return 0;}

## 3.3 Результаты работы программы

Программа готова. Осталось узнать, что все же она выдает. При запуске, программа выводит следующее:

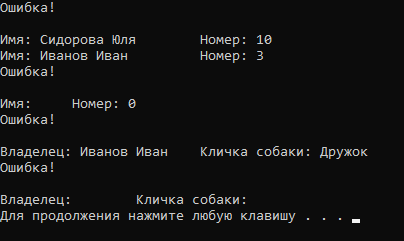


Рисунок 3. Результаты работы программы

Программа выдала значения из созданного списка, но при детальном рассмотрении вывода результатов, может возникнуть следующий вопрос: ”А почему программа выводит какие-то ошибки и не вывела все значения?”. Дело в том, что в стеке присутствует ограниченная память, и был проверен случай, при котором записать все значения не предоставляется возможным.

Вывод: в разделе “Шаблоны классов” была рассмотрена основная теоретическая часть, реализована программа, содержащая стек с необходимыми списками, и сделана проверку в случае, если память стека заполнена.

**Заключение**

В настоящее время, информационные технологии кардинально изменили нашу жизнь и потребность в технике вместе с объёмом производимых технологии становится c каждым днём всё больше и больше. На данный момент, наверное, не осталось такой области, в которой нет применения технологиям. Одной из важных сфер деятельности, в которой информационные технологии получили широкое распространение, является сфера управления. Вдобавок ко всему ранее сказанному, использование разработанных программ значительно упростит управленческий процесс, сократит время проверки расчётов, принятия решений и трудоёмкость работ при проверке результатов.  
В ходе выполнения курсовой работы были рассмотрены следующие разделы:

I.Классы   
 II.Наследование классов

III. Шаблоны классов

Таким образом, мы приходим к следующему выводу: в процессе выполнения курсовой работы, мы точно убедились в том, что классы представляют собой главное инструментальное средство C++ и являются неотъемлемой частью для объектно-ориентированного программирования. Объектно-ориентированное программирование-это способ организации кода в программе путем его группировки в виде объектов, отдельных элементов, которые включают информацию(различные данные) и функции. Объектно-ориентированный подход к организации программы позволяет группировать определенные фрагменты информации вместе с часто используемыми функциями и алгоритмами, связанными с этой информацией.

# Список литературы

1. Ключевые слова подчеркнуты [↑](#footnote-ref-1)