**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСТИЕТ**

**Лабораторная работа №1**

**по дисциплине**

**«Программирование»**

Разработка классов, создание конструкторов и деструкторов.

Использование статических членов класса

Вариант 1

Группа: **АВТ-907**

Студент: **Шурубей А.С.**

Преподаватель: **Булатов А. Д.**

НОВОСИБИРСК 2020

**Задание****:**

Построить класс Дата-Время. Класс должен включать следующие поля: день, месяц, год, часы, минуты, строковое представление даты. Класс должен обеспечивать простейшие функции для работы с данными класса: увеличение/уменьшение на 1 день, час, минуту, секунду и т.д., методы изменения объекта, вывода на экран объекта.

**Теоретические сведения:**

**Класс**

Класс - это определяемый пользователем тип. Описание класса очень похоже на описание структуры в Си. В этом смысле класс является расширением понятия структуры. В простейшем случае класс можно определить с помощью конструкции:

***тип\_класса имя\_класса {список\_членов\_класса};***

где

*тип\_класса* – одно из служебных слов **class, struct, union**;

*имя\_класса* – идентификатор;

*список\_членов\_класса* – определения и описания, типизированных данных и принадлежащих классу функций.

*Функции* – это методы класса, определяющие операции над объектом.

*Данные* – это поля объекта, образующие его структуру. Значения полей определяет состояние объекта.

Для описания объекта класса (экземпляра класса) используется конструкция

***имя\_класса имя\_объекта;***

Для изменения видимости компонент в определении класса можно использовать спецификаторы доступа: **public, private, protected**.

Общедоступные (public) компоненты класса доступны в любой части программы. Они могут использоваться любой функцией как внутри данного класса, так и вне его. Доступ извне осуществляется через имя объекта:

***имя\_объекта.имя\_члена\_класса***

***ссылка\_на\_объект.имя\_члена\_класса***

***указатель\_на\_объект->имя\_члена\_класса***

Собственные (private) компоненты класса локализованы в классе и не доступны извне. Они могут использоваться функциями – членами данного класса и функциями – “друзьями” того класса, в котором они описаны.

Защищенные (protected) компоненты доступны внутри класса и в производных классах.

**Конструктор**

Конструктор выделяет память для объекта и инициализирует данные − члены класса.

Конструктор имеет ряд особенностей:

* Для конструктора не определяется тип возвращаемого значения. Даже тип void не допустим.
* Указатель на конструктор не может быть определен, и соответственно нельзя получить адрес конструктора.
* Конструкторы не наследуются.
* Конструкторы не могут быть описаны с ключевыми словами virtual, static, const, mutuable, valatile.

Конструктор всегда существует для любого класса, если он не определен явно, то он создается автоматически. По умолчанию создается конструктор без параметров и конструктор копирования. Если конструктор описан явно, то конструктор по умолчанию не создается. По умолчанию конструкторы создаются общедоступными (public).

Параметром конструктора не может быть его собственный класс, но может быть ссылка на него (T&). Без явного указания программиста конструктор всегда автоматически вызывается при определении (создании) объекта. В этом случае вызывается конструктор без параметров. Для явного вызова конструктора используются две формы:

***имя\_класса имя\_объекта (фактические\_параметры);***

***имя\_класса (фактические\_параметры);***

**Деструктор**

Для многих типов также требуется обратное действие, деструктор, чтобы обеспечить соответствующую очистку объектов этого типа. Имя деструктора для класса X есть ~X() («дополнение конструктора»). В частности, многие типы используют некоторый объем памяти из свободной памяти, которая выделяется конструктором и освобождается деструктором.

**Пример работы программы:**

****

**Фрагменты программы:**

**Объявление конструкторов, методов и переменных**

#pragma once

class Date\_time

{

public:

Date\_time();

Date\_time(int \_day, int \_month, int \_year, int \_sec, int \_min, int \_hour); //Конструктор с параметрами

~Date\_time(); // Деструктор

int dateToDays();

int getDay();

int getMonth();

int getYear();

void setDay(int d);

void setMonth(int m);

void setYear(int y);

void nextDay();

void lastDay();

void print();

int getSec();

int getMin();

int getHour();

void setSec(int s);

void setMin(int mi);

void setHour(int h);

void nextMin();

void lastMin();

private:

int sec;

int min;

int hour;

int day;

int month;

int year;

const int mDays[12] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

};

**Функции класса:**

#include "Date\_time.h"

#include <iostream>

#include <string>

using namespace std;

Date\_time::Date\_time() {}

Date\_time::Date\_time(int \_day, int \_month, int \_year, int \_sec, int \_min, int \_hour)

{

day = \_day;

month = \_month;

year = \_year;

sec = \_sec;

min = \_min;

hour = \_hour;

}

Date\_time::~Date\_time() {}

int Date\_time::dateToDays()

{

int days = 0;

days = day;

for (int i = 0; i < month - 1; i++) {

days += mDays[i];

}

days += year \* 365 + year / 4 - year / 100 + year / 400;

return days;

}

int Date\_time::getDay()

{

return day;

}

int Date\_time::getMonth()

{

return month;

}

int Date\_time::getYear()

{

return year;

}

void Date\_time::setYear(int y)

{

if (y >= 1100 && y <= 2100)

year = y;

}

void Date\_time::setMonth(int m)

{

if (m <= 12)

month = m;

}

void Date\_time::setDay(int d)

{

if (d % 4 == 0 && month == 2 && d <= 29)

day = d;

else if (d <= mDays[month - 1])

day = d;

}

void Date\_time::nextDay()

{

if (day + 1 <= mDays[month]) day++;

else if (month < 12) { day = 1; month++; }

else { day = 1; month = 1; year++; }

}

void Date\_time::lastDay()

{

if (day - 1 >= 1) day--;

else if (day < 1) { day = mDays[month - 1]; month--; }

else { day = 31; month = 12; year--; }

}

int Date\_time::getSec()

{

return sec;

}

int Date\_time::getMin()

{

return min;

}

int Date\_time::getHour()

{

return hour;

}

void Date\_time::setSec(int s)

{ if (s < 60)

sec = s;

}

void Date\_time::setMin(int mi)

{

if (mi < 60)

min = mi;

}

void Date\_time::setHour(int h)

{

if (h < 24)

hour = h;

}

void Date\_time::nextMin()

{

if ((min + 1) < 60) min++;

else if ((hour + 1) < 24 && (min + 1) == 60) { hour += 1; min = 0; }

else { hour = 0; min = 0;}

}

void Date\_time::lastMin()

{

if ((min - 1) >= 0) min--;

else if ((min - 1) < 0 && (hour - 1) > 0) { hour--; min = 59; }

else { hour = 23; min = 59; }

}

void Date\_time::print() {

cout << day << "." << month << "." << year << endl;

cout << hour << ":" << min << ":" << sec << endl;

}

**Функция main ()**

#include "Date\_time.h"

#include <iostream>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Rus");

Date\_time primer1;

primer1.setYear(2019);

primer1.setMonth(5);

primer1.setDay(21);

primer1.nextDay();

primer1.setHour(19);

primer1.setMin(59);

primer1.setSec(13);

primer1.nextMin();

primer1.print();

Date\_time primer2;

primer2.setYear(2020);

primer2.setMonth(2);

primer2.setDay(29);

primer2.lastDay();

primer2.setHour(5);

primer2.setMin(0);

primer2.setSec(25);

primer2.lastMin();

primer2.print();

}

**Вывод:**

Было изучено создание классов и их реализации. При работе программы было замечено, что в конце её выполнения вызываются деструкторы, которые уничтожают объекты созданного класса. Помимо этого, экземпляр класса также уничтожается после вывода. Были изучены конструкторы, которые позволяют инициализировать класс без лишних вызываний метода.