# Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"

# Факультет інформатики та обчислювальної техніки Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 9 з дисципліни

«Основи програмування

»

Варіант 23

Виконав студент Панченко Сергій Віталійович

(шифр, прізвище, ім'я, по батькові)

Перевірив

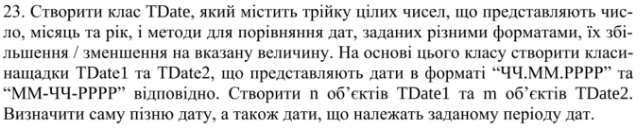
( прізвище, ім'я, по батькові)

Київ 202

## Лабораторна робота 5

**Мета:** вивчити механізми створення класів та об’єктів з використанням успадкування та поліморфізму

**Постановка задачі:**

****

C:

#include <iostream>

#include<vector>

#include<algorithm>

using namespace std;

class TDate {

protected:

class Month {

public:

int number;

int days;

string name;

Month() {}

Month(string name, int number, int days) {

this->number = number;

this->days = days;

}

};

Month months[12] = {

Month("January", 1, 31),

Month("February", 2, 28),

Month("March", 3, 31),

Month("April", 4, 30),

Month("May", 5, 31),

Month("June", 6, 30),

Month("July", 7, 31),

Month("August", 8, 31),

Month("September", 9, 30),

Month("October", 10, 31),

Month("November", 11, 30),

Month("December", 12, 31),

};

int day;

int month;

int year;

static bool IsFourth(int year) {

bool res = 0;

if (year % 4 == 0 || (year % 100 == 0 && year % 400 == 0)) {

res = 1;

}

return res;

}

int GetDays(int year, int month) {

int daysmonth = months[month - 1].days;

if (months[month-1].number==2&&IsFourth(year)) {

daysmonth = 29;

}

return daysmonth;

}

public:

TDate() {

this->day = 1;

this->month = months[0].number;

this->year = 1;

}

TDate(int day, int month, int year) {

this->day = day;

this->month = months[month - 1].number;

this->year = year;

}

int GetDay() {

return this->day;

}

int GetMonth() {

return this->month;

}

int GetYear() {

return this->year;

}

TDate& operator+=(int value) {

this->day += value;

while (this->day > GetDays(this->year, this->month)) {

this->day -= GetDays(this->year, this->month);

this->month++;

if (this->month > 12) {

this->year++;

this->month = months[0].number;

}

}

return \*this;

}

TDate& operator-=(int value)

{

this->day -= value;

while (this->day <= 0) {

this->month--;

if (this->month == 0) {

this->year--;

this->month = months[11].number;

}

this->day += GetDays(this->year, this->month);

}

return \*this;

}

bool operator >(const TDate& d) const {

bool res = 0;

if (this->year > d.year) {

res = 1;

}

else if (this->year == d.year) {

if (this->month > d.month) {

res = 1;

}

else if (this->month== d.month) {

if (this->day > d.day) {

res = 1;

}

}

}

return res;

}

bool operator ==(const TDate& d) const {

return (this->day == d.day && this->month == d.month && this->year == d.year);

}

bool operator<(const TDate& d) const

{

if ((!(\*this > d) && !(\*this == d))) {

return true;

}

return false;

}

bool operator>=(const TDate& d) const

{

if ((\*this == d) || (\*this > d)) {

return true;

}

return false;

}

bool operator<=(const TDate& d) const

{

if ((\*this == d) || (\*this < d)) {

return true;

}

return false;

}

bool operator!=(const TDate& d) const

{

if (!(\*this == d)) {

return true;

}

return false;

}

virtual string date\_string() {

char buf[255];

string date\_str;

sprintf\_s(buf, "%d:", this->day);

if (this->day < 10)

date\_str += "0";

date\_str += buf;

sprintf\_s(buf, "%d:", this->month);

if (this->month < 10)

date\_str += "0";

date\_str += buf;

sprintf\_s(buf, "%d", this->year);

if (this->year < 1000) {

int temp = this->year;

while (temp < 1000) {

temp \*= 10;

date\_str += "0";

}

}

date\_str += buf;

return date\_str;

}

static bool TDateComparator(TDate\* d1, TDate\* d2) {

return \*d1 < \*d2;

}

};

class TDate1 : public TDate {

public:

TDate1() :TDate() {}

TDate1(int day, int month, int year) :TDate(day, month, year) {}

string date\_string() override {

char buf[255];

string date\_str;

sprintf\_s(buf, "%d.", this->day);

if (this->day < 10)

date\_str += "0";

date\_str += buf;

sprintf\_s(buf, "%d.", this->month);

if (this->month < 10)

date\_str += "0";

date\_str += buf;

sprintf\_s(buf, "%d", this->year);

if (this->year < 1000) {

int temp = this->year;

while (temp < 1000) {

temp \*= 10;

date\_str += "0";

}

}

date\_str += buf;

return date\_str;

}

};

class TDate2: public TDate{

public:

TDate2() :TDate() {}

TDate2(int day, int month, int year) :TDate(day, month, year) {}

string date\_string() override {

char buf[255];

string date\_str;

sprintf\_s(buf, "%d-", this->month);

if (this->month < 10)

date\_str += "0";

date\_str += buf;

sprintf\_s(buf, "%d-", this->day);

if (this->day < 10)

date\_str += "0";

date\_str += buf;

sprintf\_s(buf, "%d", this->year);

if (this->year < 1000) {

int temp = this->year;

while (temp < 1000) {

temp \*= 10;

date\_str += "0";

}

}

date\_str += buf;

return date\_str;

}

};

void EnterArgs(int& day, int& month, int& year) {

cout << "Enter year: ";

cin >> year;

cin.ignore();

cout << "Enter month: ";

cin >> month;

cin.ignore();

cout << "Enter day: ";

cin >> day;

cin.ignore();

}

int main()

{

srand(time(NULL));

int year1 = 0;

int month1 = 0;

int day1 = 0;

int year2 = 0;

int month2 = 0;

int day2 = 0;

cout<<"Enter interval"<<endl;

cout << "\tEnter the first date" << endl;

EnterArgs(day1, month1, year1);

cout << "\tEnter the second date" << endl;

EnterArgs(day2, month2, year2);

TDate d1 = TDate(day1, month1, year1);

TDate d2 = TDate(day2, month2, year2);

vector<TDate\*>dates;

for (int i = 0; i < 10; i++)

{

dates.push\_back(new TDate1(1 + rand() % 28, 1 + rand() % 12, year1 - 3 + rand() % (year2 - year1 + 7)));

dates.push\_back(new TDate2(1 + rand() % 28, 1 + rand() % 12, year1 - 3 + rand() % (year2 - year1 + 7)));

}

cout <<endl<< "Dates" << endl;

for (int i = 0; i < dates.size(); i++)

{

cout << dates[i]->date\_string() << endl;

}

cout << endl << "Between" << endl;

for (int i = 0; i < dates.size(); i++)

{

if (\*dates[i] >= d1 && \*dates[i] <= d2) {

cout << dates[i]->date\_string() << endl;

}

}

cout <<endl<< "The Latest" << endl;

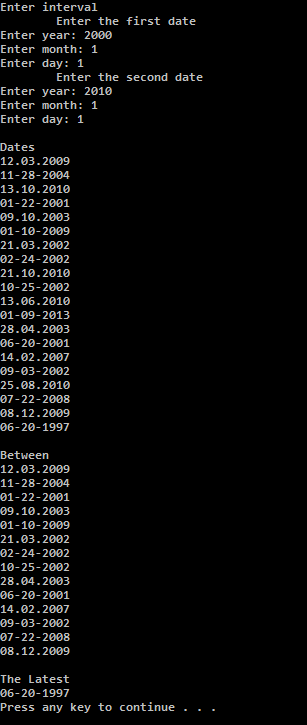
cout << (\*min\_element(dates.begin(), dates.end(), TDate::TDateComparator))->date\_string() << endl;

system("pause");

return 0;

}

**Результат:**

****

**Висновок:** під час лабораторної роботи навчилися працювати з класами та використовувати успадкування й поліморфізм. Оскільки програма створюює об’єкти класу, перезавантажує оператори та порівнює дати, виконуючи умови поставленої до варіанту задачі, то лабораторна виконана правильно.