**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»**

**ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ**

**Кафедра системного програмування та спеціалізованих комп’ютерних систем**

**Лабораторна робота №1**

**з дисципліни**

**«Об’єктно-орієнтоване програмування»**

**Тема: «Розробка простого класу»**

**Варіант 4(19)**

**Виконав: студент 2 курсу**

**ФПМ, групи КВ-11**

**Чебан М.Д.**

**Перевірив(-ла)**

**Київ – 2021**

**Завдання:**

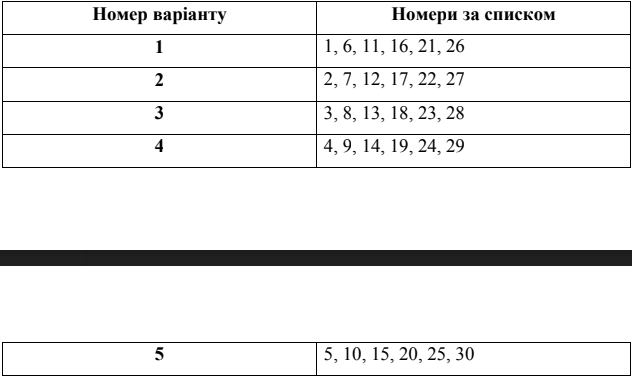
Розробити клас згідно до варіанту. Не використовувати default-реалізацію конструкторів та операторів (реалізувати самостійно). Обробка помилкових ситуацій не є обов’язковою в рамках даної роботи. Пропонується самостійно розробити інтерфейс класу, зокрема, виходячи з власних міркувань зручності його подальшого використання (бажано зробити використання класу по можливості інтуїтивно зрозумілим).

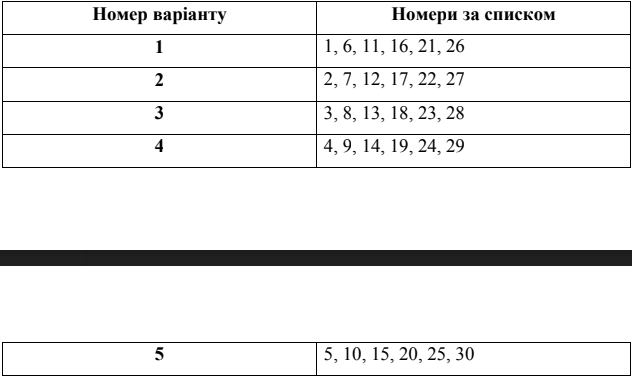
Також надати код, що демонструє спосіб використання та можливості розробленого класу (набір тестових прикладів).

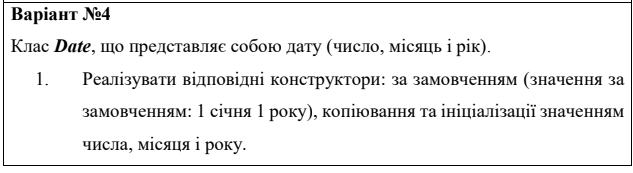
Номер варіанту визначається відповідно до номеру за списком за наступною формулою:

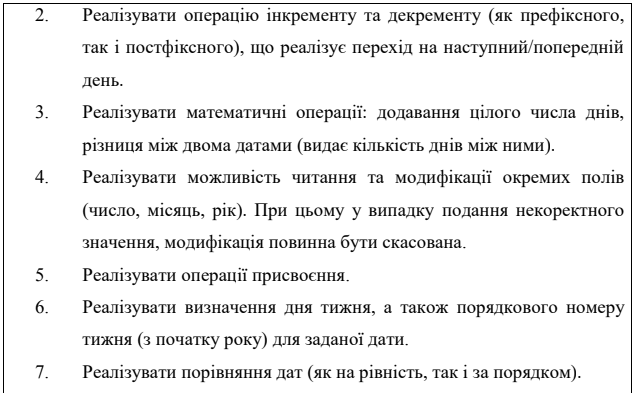
НомерВаріанту = (НомерЗаСписком – 1) mod 5 + 1

або за таблицею:





Завдання за варіантом №4:



Код програми:

Хедер файл.

Date.h

#**pragma once**

**class** Date

{

**private**:

**int** day;

**int** month;

**int** year;

**public**:

Date(); //конструктор за замовченням

Date(**int** d, **int** m, **int** y); // конструктор з параметрами

Date(**const** Date& obj); // конструктор копіювання

**bool** IfLeapYear(**int** y); // метод для з'ясування чи �" рік високосним

**bool** IfCorrectDate(**int** d, **int** m, **int** y); // метод для з'ясування чи �" дата коректною

**void** Input(**int** d, **int** m, **int** y); // метод для заповнення всіх полів одразу полів

Date& **operator**=(**const** Date& obj); // перевантаження оператору присво�"ння з копіюванням

Date& **operator**++ (); // перевантаження префіксного інкрименту

Date **operator**++ (**int** number); // перевантаження постфіксного інкрименту

Date& **operator**-- (); // перевантаження префіксного декрименту

Date **operator**-- (**int** number); // перевантаження постфіксного декрименту

**void** Print(); // метод виведення/друкування дати

**int** GetDay(); // інспектор/геттер для дня

**int** GetMonth(); // інспектор/геттер для місяці

**int** GetYear(); // інспектор/геттер для року

**void** SetDay(**int** d); // модифікатор/сеттер для дня

**void** SetMonth(**int** m); // модифікатор/сеттер для місяця

**void** SetYear(**int** y); // модифікатор/сеттер для року

**void** **operator**+(**const** **int** number); // перевантажений оператор + для додавання задано�- кількості днів

**bool** **operator**==(Date date2); // перевантажений оператор == для перевірки дат на рівність

**bool** **operator**>(Date date2); // перевантажений оператор > для порівняння дат

**bool** **operator**<(Date date2); // перевантажений оператор < для порівняння дат

**int** **operator**-(Date date2); // перевантажений оператор - для віднімання дат(визначення кількості днів між датми)

**int** CaculateWeekDay(**int** d, **int** m, **int** y); // метод для з'ясування дня тижня

**int** DayOfWeekNumber(); // метод для обрахування порядкового номеру дня тижня з початку року

};

cpp файл

Date.cpp

#include "Date.h"

#include <iostream>

#include <Windows.h>

#include <iomanip>

#include <conio.h>

**using** **namespace** std;

Date::Date() //конструктор за замовченням

{

day = 1;

month = 1;

year = 1;

}

**bool** Date::IfLeapYear(**int** y) // метод для з'ясування чи �" рік високосним

{

**if** ((y % 4 == 0 && y % 100 != 0) || y % 400 == 0)

**return** **true**;

**else** **return** **false**;

}

**bool** Date::IfCorrectDate(**int** d, **int** m, **int** y) // метод для з'ясування чи �" дата коректною

{

**int** flag = 0;

**if** (m == 1 || m == 3 || m == 5 || m == 7 || m == 8 || m == 10 || m == 12)

{

**if** (d < 32 && d > 0)

flag++;

}

**if** (m == 4 || m == 6 || m == 9 || m == 11)

{

**if** (d < 31 && d > 0)

flag++;

}

**if** (m == 2)

{

**bool** leap = IfLeapYear(y);

**if** (leap == **true**)

{

**if** (d < 30 && d > 0)

flag++;

}

**else**

{

**if** (d < 29 && d > 0)

flag++;

}

}

**if** (flag == 1)

**return** **true**;

**else** **return** **false**;

}

**void** Date::Input(**int** d, **int** m, **int** y) // метод для заповнення всіх полів одразу полів

{

**if** (IfCorrectDate(d, m, y) == **true**) // перевірка на коректність дати

{

year = y;

day = d;

month = m;

}

**else** // якщо дата не �" коректною, поля заповнюються за замовченням

{

cout << "\nIncorrect values! Default values are used now(1,1,1);\n";

day = 1;

month = 1;

year = 1;

}

}

Date::Date(**int** d, **int** m, **int** y) // конструктор з параметрами

{

**if** (IfCorrectDate(d, m, y))

{

year = y;

day = d;

month = m;

}

**else** // якщо дата не �" коректною, поля заповнюються за замовченням

{

cout << "\nIncorrect values! Default constructor is called;\n";

Date();

}

}

Date::Date(**const** Date& obj) // конструктор копіювання

{

day = obj.day;

month = obj.month;

year = obj.year;

}

Date& Date::**operator**=(**const** Date& obj) // перевантаження оператору присво�"ння з копіюванням

{

**if** (**this** == &obj)

**return** \***this**;

day = obj.day;

month = obj.month;

year = obj.year;

**return** \***this**;

}

Date& Date::**operator**++ () // перевантаження префіксного інкрименту

{

day++;

**if** (month == 12 && day == 32)// початок перевірки для переходу на наступний місяць або рік

{

day = 1;

month = 1;

year += 1;

}

**else** **if** (month == 1 || month == 3 || month == 5 || month == 7 || month == 8 || month == 10)

{

**if** (day == 32)

{

month++;

day = 1;

}

}

**else** **if** (month == 4 || month == 6 || month == 9 || month == 11)

{

**if** (day == 31)

{

month++;

day = 1;

}

}

**else** **if** (month == 2)

{

**bool** leap = IfLeapYear(year);

**if** (leap == **true**)

{

**if** (day == 30)

{

month++;

day = 1;

}

}

**else**

{

**if** (day == 29)

{

month++;

day = 1;

}

}

}

**return** \***this**;

}

Date Date::**operator**++ (**int** number) // перевантаження постфіксного інкрименту

{

Date tmp = \***this**;

**this**->day++;

**if** (month == 12 && day == 32) // початок перевірки для переходу на наступний місяць або рік

{

**this**->day = 1;

**this**->month = 1;

**this**->year += 1;

}

**else** **if** (month == 1 || month == 3 || month == 5 || month == 7 || month == 8 || month == 10)

{

**if** (day == 32)

{

month++;

**this**->day = 1;

}

}

**else** **if** (month == 4 || month == 6 || month == 9 || month == 11)

{

**if** (day == 31)

{

month++;

**this**->day = 1;

}

}

**else** **if** (month == 2)

{

**bool** leap = IfLeapYear(year);

**if** (leap == **true**)

{

**if** (day == 30)

{

month++;

**this**->day = 1;

}

}

**else**

{

**if** (day == 29)

{

month++;

**this**->day = 1;

}

}

}

**return** tmp;

}

Date& Date::**operator**-- () // перевантаження префіксного декрименту

{

day--;

**if** (month == 1 && day == 0) // початок перевірки для переходу на попередній місяць або рік

{

day = 31;

month = 12;

year -= 1;

}

**else** **if** (day == 0)

{

**if** (month - 1 == 1 || month - 1 == 3 || month - 1 == 5 || month - 1 == 7 || month - 1 == 8 || month - 1 == 10)

{

month--;

day = 31;

}

**else** **if** (month - 1 == 4 || month - 1 == 6 || month - 1 == 9 || month - 1 == 11)

{

month--;

day = 30;

}

**else** **if** (month - 1 == 2)

{

**bool** leap = IfLeapYear(year);

**if** (leap == **true**)

{

day = 29;

month--;

}

**else**

{

day = 28;

month--;

}

}

}

**return** \***this**;

}

Date Date::**operator**-- (**int** number) // перевантаження постфіксного декрименту

{

Date tmp = \***this**;

**this**->day--;

**if** (month == 1 && day == 0)// початок перевірки для переходу на попередній місяць або рік

{

**this**->day = 31;

**this**->month = 12;

**this**->year -= 1;

}

**else** **if** (**this**->day == 0)

{

**if** (month - 1 == 1 || month - 1 == 3 || month - 1 == 5 || month - 1 == 7 || month - 1 == 8 || month - 1 == 10)

{

**this**->month--;

**this**->day = 31;

}

**else** **if** (month - 1 == 4 || month - 1 == 6 || month - 1 == 9 || month - 1 == 11)

{

**this**->month--;

**this**->day = 30;

}

**else** **if** (month - 1 == 2)

{

**bool** leap = IfLeapYear(year);

**if** (leap == **true**)

{

**this**->day = 29;

**this**->month--;

}

**else**

{

**this**->day = 28;

**this**->month--;

}

}

}

**return** tmp;

}

**void** Date::Print() // метод виведення/друкування дати

{

cout << **char**(201);

**for** (**int** i = 0; i < 20; i++)

{

**if** (i == 5 || i == 13)

cout << **char**(203);

**else** cout << **char**(205);

}

cout << **char**(187) << endl;

cout << **char**(186) << " Day " << **char**(186) << " Month " << **char**(186) << " Year " << **char**(186) << endl;

cout << **char**(204);

**for** (**int** i = 0; i < 20; i++)

{

**if** (i == 5 || i == 13)

cout << **char**(206);

**else** cout << **char**(205);

}

cout << **char**(185) << endl;

cout << **char**(186) << " ";

cout << setw(2) << day;

cout << " " << **char**(186) << " ";

cout << setw(2) << month << " " << **char**(186) << " ";

cout << setw(4) << year << " " << **char**(186) << endl;

cout << **char**(200);

**for** (**int** i = 0; i < 20; i++)

{

**if** (i == 5 || i == 13)

cout << **char**(202);

**else** cout << **char**(205);

}

cout << **char**(188) << endl;

}

**int** Date::GetDay() // інспектор/геттер для дня

{

**return** day;

}

**int** Date::GetMonth() // інспектор/геттер для місяці

{

**return** month;

}

**int** Date::GetYear() // інспектор/геттер для року

{

**return** year;

}

**void** Date::SetDay(**int** d) // модифікатор/сеттер для дня

{

**int** tmp = day;

day = d;

**if** (IfCorrectDate(day, month, year) == **false**)

{

cout << "Incorrect value! Modification is canceled(DAY)\n";

day = tmp;

system("pause");

}

}

**void** Date::SetMonth(**int** m) // модифікатор/сеттер для місяця

{

**int** tmp = month;

month = m;

**if** (IfCorrectDate(day, month, year) == **false**)

{

cout << "Incorrect value! Modification is canceled(MONTH)\n";

month = tmp;

system("pause");

}

}

**void** Date::SetYear(**int** y) // модифікатор/сеттер для року

{

**int** tmp = year;

year = y;

**if** (IfCorrectDate(day, month, year) == **false**)

{

cout << "Incorrect value! Modification is canceled(YEAR)\n";

year = tmp;

system("pause");

}

}

**void** Date::**operator**+(**const** **int** number) // перевантажений оператор + для додавання задано�- кількості днів

{

day += number;

**while** (IfCorrectDate(day, month, year) != **true**)

{

**if** (month == 12 && day > 31)

{

day = day - 31;

month = 1;

year += 1;

}

**else** **if** (month == 1 || month == 3 || month == 5 || month == 7 || month == 8 || month == 10)

{

**if** (day > 31)

{

month++;

day = day - 31;

}

}

**else** **if** (month == 4 || month == 6 || month == 9 || month == 11)

{

**if** (day > 30)

{

month++;

day = day - 30;

}

}

**else** **if** (month == 2)

{

**bool** leap = IfLeapYear(year);

**if** (leap == **true**)

{

**if** (day > 29)

{

month++;

day -= 29;

}

}

**else**

{

**if** (day > 28)

{

month++;

day = day - 28;

}

}

}

}

}

**bool** Date::**operator**==(Date date2) // перевантажений оператор == для перевірки дат на рівність

{

**if** (year == date2.year && month == date2.month && day == date2.day)

**return** **true**;

**else** **return** **false**;

}

**bool** Date::**operator**>(Date date2) // перевантажений оператор > для порівняння дат

{

**if** (year > date2.year)

{

**return** **true**;

}

**else** **if** (year == date2.year && month > date2.month)

{

**return** **true**;

}

**if** (year == date2.year && month == date2.month && day > date2.day)

{

**return** **true**;

}

**else** **return** **false**;

}

**bool** Date::**operator**<(Date date2) // перевантажений оператор < для порівняння дат

{

**if** (year < date2.year)

{

**return** **true**;

}

**else** **if** (year == date2.year && month < date2.month)

{

**return** **true**;

}

**if** (year == date2.year && month == date2.month && day < date2.day)

{

**return** **true**;

}

**else** **return** **false**;

}

**int** Date::**operator**-(Date date2) // перевантажений оператор - для віднімання дат(визначення кількості днів між датми)

{

**if** (\***this** == date2)

{

**return** 0;

}

**int** count = 0;

Date tmp = \***this**;

**if** (\***this** > date2)

{

**while** ((\***this** == date2) == **false**)

{

--\* **this**;

count++;

}

}

**else**

{

**while** ((\***this** == date2) == **false**)

{

++\* **this**;

count++;

}

}

\***this** = tmp;

**return** count;

}

**int** Date::CaculateWeekDay(**int** d, **int** m, **int** y) // метод для з'ясування дня тижня

{

**if** (m == 1 || m == 2) {

m += 12;

y--;

}

**int** iWeek = (d + 2 \* m + 3 \* (m + 1) / 5 + y + y / 4 - y / 100 + y / 400) % 7; //Формула розрахунку Кіма Ларсона

**switch** (iWeek)

{

**case** 0:

cout << " Monday \n" << endl;

**break**;

**case** 1:

cout << " Tuesday \n" << endl;

**break**;

**case** 2:

cout << " Wednesday \n" << endl;

**break**;

**case** 3:

cout << " Thursday \n" << endl;

**break**;

**case** 4:

cout << " Friday \n" << endl;

**break**;

**case** 5:

cout << " Saturday \n" << endl;

**break**;

**case** 6:

cout << " Sunday \n" << endl;

**break**;

}

**return** iWeek;

}

**int** Date::DayOfWeekNumber() // метод для обрахування порядкового номеру дня тижня з початку року

{

Date tmp(1, 1, year);

**int** count = 1;

**while** (tmp < \***this**)

{

tmp + 7;

count++;

}

**return** count;

}

файл із функцією main

Lab1.cpp

// Lab1.cpp : Этот файл содержит функцию "main". �-десь начинается и заканчивается выполнение программы.

//

#include <iostream>

#include <Windows.h>

#include <iomanip>

#include <conio.h>

#include "Date.h"

**using** **namespace** std;

**int** main()

{

Date dt, dt1, dt2;

**bool** f = 1;

**int** d = 22, m = 11, y = 2003;

**int** d1, m1, y1, n;

Date newdt(d, m, y);

Date copydt(newdt);

**while** (f == 1)

{

system("cls");

**int** choice;

cout << "Enter the choice: \n";

cout << " Press ENTER to print the date(object newdt) \n";

cout << " Press ESCAPE to finish the program \n";

cout << "1 - Test of constructors; \n";

cout << "2 - Increment and decrement operators; \n";

cout << "3 - Adding an integer number of days, difference between two dates; \n";

cout << "4 - Setters and getters(object dt); \n";

cout << "5 - Assignment operations; \n";

cout << "6 - Define day and number of the week(object newdt); \n";

cout << "7 - Comparison of dates \n";

choice = \_getch();

**switch** (choice)

{

**case** VK\_RETURN:

{

system("cls");

newdt.Print();

system("pause");

**break**;

}

**case** VK\_ESCAPE:

{

f = 0;

**break**;

}

**case** '1':

system("cls");

cout << "Task1.\n";

cout << "Implement the corresponding constructors:" << endl;

cout << "by default (default value: January 1 of " << endl;

cout << "year 1), copying and initialization with " << endl;

cout << "a value day, month and year." << endl << endl;;

cout << "Default constructor(dt):\n";

dt.Print();

cout << "\n\Constructor with parametres(newdt):\n";

newdt.Print();

cout << "\n\Copy constructor:(copydt, copied from newdt)\n";

copydt.Print();

system("pause");

**break**;

**case** '2':

system("cls");

cout << "Task2.\n";

cout << "Implement increment and decrement operations" << endl;

cout << "(both prefix and postfix), which implements " << endl;

cout << "the transition to the next / previous day. " << endl << endl;

cout << "Enter the new date: \n";

cout << "Day: ";

cin >> d;

cout << "Month: ";

cin >> m;

cout << "Year: ";

cin >> y;

dt.Input(d, m, y);

cout << endl << endl;

cout << "increments: \n";

cout << "In code: dt++.Print();" << endl;;

dt++.Print();

cout << endl;

cout << "In code: dt.Print();" << endl;;

dt.Print();

cout << endl;

cout << "In code: (++dt).Print();" << endl;;

(++dt).Print();

cout << endl;

cout << "decrements: \n";

cout << "In code: dt--.Print();" << endl;;

dt--.Print();

cout << endl;

cout << "In code: dt.Print();" << endl;;

dt.Print();

cout << endl;

cout << "In code: (--dt).Print();" << endl;;

(--dt).Print();

cout << endl;

system("pause");

**break**;

**case** '3':

system("cls");

cout << "Task3.\n";

cout << "Implement mathematical operations: adding an " << endl;

cout << "integer number of days, difference between two " << endl;

cout << "dates(gives the number of days between them)." << endl << endl;

cout << "\n newdt: \n";

newdt.Print();

cout << "\nEnter the number of days(n): ";

cin >> n;

newdt + n;

cout << "\nResult(int code: newdt + n;):\n";

newdt.Print();

cout << "\n\nThe number of days between two dates\n\n";

cout << "Enter the first date(Date1): \n";

cout << "Day: ";

cin >> d;

cout << "Month: ";

cin >> m;

cout << "Year: ";

cin >> y;

dt1.Input(d, m, y);

cout << "\nEnter the second date(Date2): \n";

cout << "Day: ";

cin >> d;

cout << "Month: ";

cin >> m;

cout << "Year: ";

cin >> y;

dt2.Input(d, m, y);

n = dt1 - dt2;

cout << "\nNumber of days between Date1 and Date2 = " << n << endl << endl;

system("pause");

**break**;

**case** '4':

system("cls");

cout << "Task4.\n";

cout << "Implement the possibility of reading and modifying " << endl;

cout << "individual fields (date, month, year). At the same " << endl;

cout << "time, in the case of submission of incorrect value, " << endl;

cout << "the modification must be canceled. " << endl << endl;

cout << "\n dt: \n";

dt.Print();

cout << "\nEnter the day: ";

cin >> d;

dt.SetDay(d);

dt.Print();

cout << "\n\nEnter the month: ";

cin >> m;

dt.SetMonth(m);

dt.Print();

cout << "\n\nEnter the year: ";

cin >> y;

dt.SetYear(y);

dt.Print();

cout << "\nGetters: \n\n";

cout << "Day: " << dt.GetDay() << endl;

cout << "Month: " << dt.GetMonth() << endl;

cout << "Year: " << dt.GetYear() << endl;

system("pause");

**break**;

**case** '5':

system("cls");

cout << "Task4.\n";

cout << "Implement assignment operations." << endl << endl;

cout << "dt:\n";

dt.Print();

cout << "newdt:\n";

newdt.Print();

cout << "In code: newdt = dt; newdt.Print(); \nResult:\n";

newdt = dt;

newdt.Print();

system("pause");

**break**;

**case** '6':

system("cls");

cout << "Task6.\n";

cout << "Implement the definition of the day of " << endl;

cout << "the week, as well as the serial number " << endl;

cout << "of the week (from the beginning of the " << endl;

cout << "year) for a given date. " << endl << endl;

cout << "\n newdt: \n";

newdt.Print();

cout << endl;

newdt.CaculateWeekDay(newdt.GetDay(), newdt.GetMonth(), newdt.GetYear());

cout << "The number of week in this year = " << newdt.DayOfWeekNumber() << endl << endl;

system("pause");

**break**;

**case** '7':

system("cls");

cout << "Task7.\n";

cout << "Implement date comparison" << endl;

cout << "(both for equality and in order)." << endl << endl;

cout << "Enter the first date: \n";

cout << "Day: ";

cin >> d;

cout << "Month: ";

cin >> m;

cout << "Year: ";

cin >> y;

dt1.Input(d, m, y);

cout << "\nEnter the second date: \n";

cout << "Day: ";

cin >> d;

cout << "Month: ";

cin >> m;

cout << "Year: ";

cin >> y;

dt2.Input(d, m, y);

**if** (dt1 > dt2)

{

cout << "\nDate1 > Date2\n";

}

**else** **if** (dt1 < dt2)

{

cout << "\nDate1 < Date2\n";

}

**else** **if** (dt1 == dt2)

{

cout << "\nDate1 == Date2\n";

}

system("pause");

**break**;

}

}

cout << "\n\n\nKV-11, Cheban Maksym Dmytrovych, Variant 19(4)\n\n";

}

Скріншоти із тестуванням програми:

