**Лабораторна робота №2**

**Тема.** Перевантаження операцій.

**Мета.** Одержати практичні навички створення абстрактних типів даних і перевантаження операцій у мові С++.

**Постановка задачі:** Розширити функціональність класу, реалізовану в попередній роботі, згідно тогож варіанту.

Завдання 1:

В клас Rectangle додати:

* + Перевантаження:
    - операції ++ (--): одночасно збільшує (зменшує) значення полів а і b;
    - констант true і false: звернення до екземпляра класу дає значення true, якщо прямокутник із заданими довжинами сторін є квадратом, інакше false;
    - операції \*: одночасно домножает поля а і b на скаляр;
    - перетворення типу Rectangle в string (і навпаки).

Головна програма:

Rectangle.h:

#pragma once

#include <iostream>

#include <string>

#include <iomanip>

using namespace std;

class Rectangle

{

private:

double a, b;

public:

Rectangle();

Rectangle(double a, double b);

double P() const;

double S() const;

void setSizes();

void setA();

void setB();

void getSizes() const;

double getA() const;

double getB() const;

bool IsSquare() const;

void WhileNotNormal(double& num);

Rectangle& operator++();

Rectangle& operator--();

Rectangle operator++(int);

Rectangle operator--(int);

operator bool() const;

Rectangle operator\*(int scalar) const;

operator string() const;

Rectangle operator+(const Rectangle& r2) const;

Rectangle operator+(int side) const;

friend Rectangle operator+(int side, const Rectangle& r2);

Rectangle(const string& str);

friend ostream& operator<<(ostream& out, const Rectangle& r);

static void Show();

friend istream& operator>>(istream& in, Rectangle& r);

double& operator[](int index);

Rectangle& operator +=(double num);

Rectangle& operator +=(int num);

Rectangle& operator +=(const Rectangle& r);

friend bool operator ==(const Rectangle& r1, const Rectangle& r2);

friend bool operator >=(const Rectangle& r1, const Rectangle& r2);

friend bool operator <=(const Rectangle& r1, const Rectangle& r2);

friend bool operator >(const Rectangle& r1, const Rectangle& r2);

friend bool operator <(const Rectangle& r1, const Rectangle& r2);

friend bool operator !=(const Rectangle& r1, const Rectangle& r2);

Rectangle& operator =(const Rectangle& r);

};

Rectangle.cpp:

#include "Rectangle.h"

void Rectangle::WhileNotNormal(double& num) {

while (num <= 0) {

cout << "Невiрно задано сторону введiть знову: ";

cin >> num;

}

}

Rectangle::Rectangle(double a, double b) {

WhileNotNormal(a);

WhileNotNormal(b);

this->a = a;

this->b = b;

}

Rectangle::Rectangle() { a = 0; b = 0; }

Rectangle::Rectangle(const string& str) {

int cPos = str.find("|");

if (cPos != string::npos) {

string num1 = str.substr(0, cPos);

string num2 = str.substr(cPos + 1);

a = stod(num1);

b = stod(num2);

}

else

a = b = 0;

}

void Rectangle::Show()

{

cout << setw(15) << "1 сторона" << setw(5) << "|" << setw(15) << "2 сторона" << endl;

}

double Rectangle::P() const {

if (a == 0 || b == 0) {

cout << "Недостатньо значень сторiн для периметру" << endl;

return 0;

}

return 2 \* (a + b);

}

double Rectangle::S() const {

if (a == 0 || b == 0) {

cout << "Недостатньо значень сторiн для площi." << endl;

return 0;

}

return a \* b;

}

void Rectangle::setA() {

cout << "Введiть сторону a: ";

cin >> this->a;

WhileNotNormal(a);

}

void Rectangle::setB() {

cout << "Введiть сторону b: ";

cin >> this->b;

WhileNotNormal(b);

}

void Rectangle::setSizes() {

this->setA();

this->setB();

}

double Rectangle::getA() const {

return a;

}

double Rectangle::getB() const {

return b;

}

void Rectangle::getSizes() const {

if (IsSquare())

cout << "Квадрат\n";

else

cout << "Прямокуник\n";

cout << "Довжина сторони a: " << getA() << ", дожина сторни b:" << getB() << endl;

}

bool Rectangle::IsSquare() const {

if (a == 0 || b == 0) {

cout << "Недостатньо значень сторiн для визначення того чи квадрат." << endl;

return false;

}

return a == b;

}

Rectangle& Rectangle::operator++() {

a += 1.0; b += 1.0;

return \*this;

}

Rectangle& Rectangle::operator--() {

if (a > 1.0 && b > 1.0) {

a -= 1; b -= 1;

}

else

throw exception("Вихiд за межi сторiн");

return \*this;

}

Rectangle Rectangle::operator++(int) {

Rectangle temp = \*this;

a += 1; b += 1;

return temp;

}

Rectangle Rectangle::operator--(int) {

Rectangle temp = \*this;

if (a > 1.0 && b > 1.0) {

a -= 1; b -= 1;

}

else

throw exception("Вихiд за можлве значення");

return temp;

}

Rectangle::operator bool() const { return a == b && a != 0; }

Rectangle::operator string() const {

string num = to\_string(a) + "|" + to\_string(b);

return num;

}

Rectangle Rectangle::operator+(const Rectangle& r2) const

{

return Rectangle(a + r2.a, b + r2.b);

}

Rectangle Rectangle::operator+(int side) const

{

return Rectangle(a + side, b + side);

}

Rectangle Rectangle::operator\*(int scalar) const {

double x1 = a, x2 = b;

if (scalar > 0) {

x1 \*= scalar;

x2 \*= scalar;

}

else throw exception("Не можна множити на вiд'ємну сторону");

return Rectangle(x1, x2);

}

Rectangle operator+(int side, const Rectangle& r2)

{

return Rectangle(r2.a + side, r2.b + side);

}

double& Rectangle::operator [](int index)

{

if (index == 0)

return a;

else if (index == 1)

return b;

else

throw out\_of\_range ("Невiрний iндекс");

}

Rectangle& Rectangle::operator+=(double num)

{

a += num; b += num;

return \*this;

}

Rectangle& Rectangle::operator+=(int num)

{

a += num; b += num;

return \*this;

}

Rectangle& Rectangle::operator+=(const Rectangle& r)

{

a += r.a;

b += r.b;

return \*this;

}

ostream& operator<<(ostream& out, const Rectangle& r) {

out << setw(15) << r.a << setw(5) << "|" << setw(15) << r.b;

return out;

}

istream& operator>>(istream& in, Rectangle& r)

{

double x1 = -1, x2 = -1;

while (x1 <= 0 || x2 <= 0 )

if (!(in >> x1 && in >> x2))

return in;

r.a = x1;

r.b = x2;

return in;

}

bool operator==(const Rectangle& r1, const Rectangle& r2)

{

return r1.a == r2.a && r1.b == r2.b;

}

bool operator>=(const Rectangle& r1, const Rectangle& r2)

{

return r1.S() >= r2.S();

}

bool operator<=(const Rectangle& r1, const Rectangle& r2)

{

return r1.S() <= r2.S();

}

bool operator>(const Rectangle& r1, const Rectangle& r2)

{

return !(r1 <= r2);

}

bool operator<(const Rectangle& r1, const Rectangle& r2)

{

return !(r1 >= r2);

}

bool operator!=(const Rectangle& r1, const Rectangle& r2)

{

return !(r1 == r2);

}

Rectangle& Rectangle::operator=(const Rectangle& r) {

if (\*this == r)

return \*this;

a = r.a;

b = r.b;

return \*this;

}

Main.cpp:

#include "Rectangle.h"

int main() {

setlocale(0, "UKR");

//Основні елементи

Rectangle rect = Rectangle(5, 5);

string r1STR = rect;

cout << "Str = " << r1STR << endl;

Rectangle::Show();

cout << rect << "(Об'єкт rect - початкове значення)" << endl;

cout << rect++ << "(rect++)" << endl;

cout << rect-- << "(rect--)"<< endl;

cout << ++rect << "(++rect)" << endl;

cout << --rect << "(--rect)" << endl;

cout << "rect\*5: " << endl;

Rectangle::Show();

cout << rect \* 5 << endl;

cout << endl;

cout << "Рядкове значення (r):" << endl;

Rectangle r = r1STR;

cout << "r квадрат? ";

if (r)

cout << "Квадрат" << endl;

Rectangle::Show();

cout << r << endl;

Rectangle rectangle1 = r1STR;

Rectangle rectangle2(10, 10);

cout << endl;

Rectangle::Show();

cout << rectangle1 << " (rectngle1 початкове значення)" << endl;

cout << rectangle2 << " (rectngle2 почакове значення)" << endl;

cout << rectangle1 + rectangle2 << " (Сума двох прямокутникiв)"<<endl;

cout << 10 + rectangle1 << " (Число + прямокутник:)" << endl;

cout << rectangle1 + 10 << " (Прямокутник + число)" << endl;

//Індексація та +=

Rectangle rr;

Rectangle::Show();

cout << rr << "(пустий об'єкт rr)" << endl;

cout << "Введiть значення для полiв Rectangle (Перша сторона, друга сторона): "; cin >> rr;

cout << rr << "(Об'єкт rr піля введення)" << endl;

(rr += 5)+=10;

cout << rr << "Об'єкт rr пiсля операцiї (rr += 5)+=10" << endl;

cout << "Доступ до полiв за iндексом: " << endl;

cout << "rr[0] = " << rr[0] << endl;

rr[0] = 10;

cout << "rr[0] має дорiвнювати 10: " << rr[0] << endl;

//Логіні операції

Rectangle r1(10, 20), r2(5, 10);

Rectangle::Show();

cout << r1 << "(Об'єкт r1)" << endl;

cout << r2 << "(Об'єкт r2)" << endl;

bool a = r1 == r2;

cout << "r1 == r2 " << a << endl;

a = r1 >= r2;

cout << "r1 >= r2 " << a << endl;

a = r1 > r2;

cout << "r1 > r2 " << a << endl;

a = r1 <= r2;

cout << "r1 <= r2 " << a << endl;

a = r1 < r2;

cout << "r1 < r2 " << a << endl;

a = r1 != r2;

cout << "r1 != r2 " << a << endl;

// Перевизначені оператори потоку

cout << "Створено масив об'єктiв:" << endl;

Rectangle rectangles[5];

for (int i = 0; i < 5; i++) {

cout << "Введiть " << i + 1 << " прямокутник: ";

cin >> rectangles[i];

}

Rectangle::Show();

for (int i = 0; i < 5; i++)

cout << rectangles[i] << endl;

}

Тестовий набір

|  |  |  |
| --- | --- | --- |
| № | Програма | Тествоий набір |
| 1 |  | Rectangle rect = new Rectangle(5,5),  Перетворено в рядок:  string r1STR = rect;  Та виведено:  cout << "Str = " << r1STR << endl;  Використано інкремент та декремент (постфіксний, префіксний) для rect:  cout << rect++;  cout << rect--;  cout << ++rect;  cout << --rect;  Виконано збільшення сторін на 5:  cout << rect \* 5;  Створено об’єкт з рядка:  Rectangle r = r1STR;  Перевірено на те чи r квадрат  if (r)  cout << "Квадрат" << endl;  Виведено r:  cout << r;  Створено два об’єкти для виконання дій:  Rectangle rectangle1 = r1STR;  Rectangle rectangle2(10, 10);  Виконано дії із об’єктами:  cout << rectangle1 + rectangle2;  cout << 10 + rectangle1;  cout << rectangle1 + 10;  Створено порожній об’єкт та виведено його:  Rectangle rr;  cout << rr;  Введено поля об’єкту rr;  cin >> rr;  Виконано операцію для об’єкту: (rr += 5)+=10;  Отримано значення за індексом, змінено його та виведено нове:  cout << "rr[0]  rr[0] = 10;  cout << "rr[0] ;  Створено два об’єкти: Rectangle r1(10, 20), r2(5, 10);  bool a = r1 == r2;  Виконано логічні перевірки над о’бєктами  cout << "r1 == r2 " << a;  a = r1 >= r2;  cout << "r1 >= r2 " << a;  a = r1 > r2;  cout << "r1 > r2 " << a;  a = r1 <= r2;  cout << "r1 <= r2 " << a;  a = r1 < r2;  cout << "r1 < r2 " << a;  a = r1 != r2;  cout << "r1 != r2 " << a;  Створено та виведено масив об’єктів.  Rectangle rectangles[5];  for (int i = 0; i < 5; i++) {  cin >> rectangles[i];  }  Rectangle::Show();  for (int i = 0; i < 5; i++)  cout << rectangles[i]; |

Завдання 2: Додати нові функціональні можливості до класу (C# Windows Form) за допомогою перевантаження операторів.

Form1.cs:

using System;

using System.Collections.Generic;

using System.Collections.Specialized;

using System.ComponentModel;

using System.Data;

using System.Diagnostics.Eventing.Reader;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Runtime.InteropServices;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Xml;

namespace LabRob1

{

    public partial class Form1 : Form

    {

        DataTable dt = new DataTable();

        DataTable dt1;

        List<Appliances> list;

        string filter = "Binary file (\*.BIN)|\*.BIN";

        CheckBox[] chk;

        Control[] textBoxes;

        public Form1()

        {

            InitializeComponent();

            dt.Columns.Add("ID", typeof(string));

            dt.Columns.Add("Ім'я", typeof(string));

            dt.Columns.Add("Бренд", typeof(string));

            dt.Columns.Add("Ціна", typeof(double));

            dt.Columns.Add("Рік випуску", typeof(int));

            dt.Columns.Add("Тип спожвання", typeof(string));

            dt.Columns.Add("Потужність", typeof(int));

            dt.Columns.Add("Витратність (на год.)", typeof(double));

            dt1 = dt.Copy();

            dataGridView1.DataSource = dt;

            list = new List<Appliances>();

            saveFileDialog1.Filter = filter;

            saveFileDialog1.Title = "Збереження в бінарний файл";

            openFileDialog1.Filter = filter;

            openFileDialog1.Title = "Відкриття бінарного файлу";

            chk = new CheckBox[] { checkBox9, checkBox10, checkBox11, checkBox12, checkBox13, checkBox14, checkBox15 };

            textBoxes = new Control[] { textBox19, textBox25, textBox24, textBox22, textBox21, comboBox13, textBox20 };

        }

        // Створення

        private void toolStripButton5\_Click(object sender, EventArgs e)

        {

            if (IsEmpty(textBox10) || IsEmpty(textBox8) || IsEmpty(textBox9) || IsEmpty(textBox11) || IsEmpty(textBox12) || IsEmpty(textBox23) || IsEmpty(comboBox11))

                return;

            list.Add(new Appliances(textBox10.Text, textBox8.Text, textBox9.Text, double.Parse(textBox11.Text), short.Parse(textBox12.Text), comboBox11.Text, short.Parse(textBox23.Text)));

        }

        // Введення в поле

        private void comboBox11\_KeyPress(object sender, KeyPressEventArgs e)

        {

            e.Handled = true;

        }

        private void textBox11\_KeyPress(object sender, KeyPressEventArgs e)

        {

            TextBox textBox = sender as TextBox;

            if (char.IsDigit(e.KeyChar) || char.IsControl(e.KeyChar))

                return;

            if (e.KeyChar == ',' && !textBox.Text.Contains(',') && textBox.SelectionStart != 0)

                return;

            e.Handled = true;

        }

        private void textBox23\_KeyPress(object sender, KeyPressEventArgs e)

        {

            if (char.IsDigit(e.KeyChar) || char.IsControl(e.KeyChar))

                return;

            e.Handled = true;

        }

        private bool IsEmpty(Control t)

        {

            if (string.IsNullOrEmpty(t.Text))

            {

                t.Focus();

                return true;

            }

            return false;

        }

        private void AddToTable(DataTable t, Appliances a)

        {

            t.Rows.Add(a.Id, a.Name, a.Brand, a.Price, a.Year, a.EnergyClass, a.Power, a.EnergyCost);

        }

        private void UpdateDataGridView1()

        {

            dt.Clear();

              foreach (Appliances a in list)

                     AddToTable(dt,a);

            dataGridView1.DataSource = null;

            dataGridView1.DataSource = dt;

        }

        // Знижка

        private void toolStripButton6\_Click(object sender, EventArgs e)

        {

            foreach (DataGridViewRow row in dataGridView1.SelectedRows)

                if (row.Index < list.Count)

                    list[row.Index].Discount();

            UpdateDataGridView1();

        }

        private void вивестиМасивToolStripMenuItem\_Click(object sender, EventArgs e)

        {

            UpdateDataGridView1();

        }

        private void Clear(params Control[] controls)

        {

            foreach (Control c in controls)

                c.Text = "";

        }

        // Збереженн  - читання

        private void зберегтиToolStripMenuItem\_Click(object sender, EventArgs e)

        {

            if (list.Count > 0 && saveFileDialog1.ShowDialog() == DialogResult.OK)

            {

                try

                {

                    string path = saveFileDialog1.FileName;

                    using (FileStream fs = new FileStream(path, FileMode.Create))

                    using (BinaryWriter br = new BinaryWriter(fs))

                    {

                        foreach (Appliances a in list)

                        {

                            br.Write(a.Id);

                            br.Write(a.Name);

                            br.Write(a.Brand);

                            br.Write(a.Price);

                            br.Write(a.Year);

                            br.Write(a.EnergyClass);

                            br.Write(a.Power);

                        }

                    }

                }

                catch { MessageBox.Show("Помилка запису в файл", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error); return; }

            }

        }

        private void відкритиToolStripMenuItem\_Click(object sender, EventArgs e)

        {

            if (openFileDialog1.ShowDialog() == DialogResult.OK)

            {

                try

                {

                    string path = openFileDialog1.FileName;

                    using (FileStream fs = new FileStream(path, FileMode.Open))

                    using (BinaryReader br = new BinaryReader(fs))

                    {

                        list.Clear();

                        while (br.BaseStream.Position < br.BaseStream.Length)

                        {

                            string id = br.ReadString();

                            string name = br.ReadString();

                            string brand = br.ReadString();

                            double price = br.ReadDouble();

                            short year = br.ReadInt16();

                            string type = br.ReadString();

                            short power = br.ReadInt16();

                            list.Add(new Appliances(id, name, brand, price, year, type, power));

                        }

                    }

                }

                catch { MessageBox.Show("Помилка читання файлу.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error); return; }

                UpdateDataGridView1();

            }

        }

        // Виделення

        private void dataGridView1\_CellClick(object sender, DataGridViewCellEventArgs e)

        {

            int index = e.RowIndex;

            if (index <0)

                return;

            textBox13.Text = list[index].Id;

            textBox18.Text = list[index].Name;

            textBox17.Text = list[index].Brand;

            textBox16.Text = list[index].Price.ToString();

            textBox15.Text = list[index].Year.ToString();

            comboBox12.Text = list[index].EnergyClass;

            textBox14.Text = list[index].Power.ToString();

        }

        private void toolStripButton1\_Click(object sender, EventArgs e)

        {

            List<Appliances> delete = new List<Appliances>();

            foreach (DataGridViewRow row in dataGridView1.SelectedRows)

            {

                int index = row.Index;

                if (index >= 0 && index < list.Count)

                    if (MessageBox.Show($"Видаляти {list[index].Name} {list[index].Brand} {list[index].Year}?", "Питання", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

                        delete.Add(list[index]);

            }

            foreach (Appliances a in delete)

                list.Remove(a);

            Clear(textBox18, textBox17, textBox16, textBox15, textBox14, textBox13, comboBox12);

            UpdateDataGridView1();

        }

        // Поява/зникнення другої таблиці

        private void tabControl2\_TabIndexChanged(object sender, EventArgs e)

        {

            TabControl tabControl = (TabControl)sender;

            string text = tabControl.SelectedTab.Text;

            if (text == "Пошук")

            {

                smallSize();

            }

            else

            {

                bigSize();

            }

            toolStripButton11.Visible = false;

        }

        // Розміри вікна

        public void smallSize()

        {

            dataGridView2.Visible = true;

            label39.Visible = true;

            dataGridView1.Height = 230;

        }

        public void bigSize()

        {

            dataGridView1.Height = 470;

            dataGridView2.Visible = false;

            label39.Visible = false;

        }

        // Пошук

        private void toolStripButton2\_Click(object sender, EventArgs e)

        {

            bool any = false;

            foreach (CheckBox c in chk)

                any |= c.Checked;

            if (!any)

                return;

            dt1.Rows.Clear();

            foreach (DataRow row in dt.Rows)

            {

                bool find = true;

                for (int i = 0; i < chk.Length; i++)

                {

                    if (chk[i].Checked)

                        if (textBoxes[i].Text != row[i].ToString())

                        {

                            find = false;

                            break;

                        }

                }

                if (find)

                    dt1.Rows.Add(row.ItemArray);

            }

            if(dt1.Rows.Count == 0)

            {

                MessageBox.Show("Елементів не знайдено.", "Повідомлення", MessageBoxButtons.OK, MessageBoxIcon.Information);

                return;

            }

            dataGridView2.DataSource = dt1;

        // Конструктор зазамовчуванням

        private void toolStripButton7\_Click(object sender, EventArgs e)

        {

            list.Add(new Appliances());

        }

        // Текстове повідомлення

        private void toolStripButton8\_MouseHover(object sender, EventArgs e)

        {

            textBox30.Clear();

            textBox30.Text = "Кнопка \"Арифметичні\"" + Environment.NewLine + "Виберіть рядок/рядки таблиці," + Environment.NewLine + "Введіть число в поле та натисніть кнопку для виконання дії позначеної зліва від полів," + Environment.NewLine + "Можна використовувати декілька полів та виконати арифметичні операції.";

        }

        private void toolStripButton9\_MouseHover(object sender, EventArgs e)

        {

            textBox30.Clear();

            textBox30.Text = "Кнопка \"Пошук\"" + Environment.NewLine + "Виберіть рядок таблиці" + Environment.NewLine + "Натисніть кнопку," + Environment.NewLine + "Вам буде показано список всіх товарів із ціною вибраного," + Environment.NewLine + "(Якщо вирати декілька рядків, то будуть виведені товари, які входять в перелік цін вибраних товарів)";

        }

        private void toolStripButton10\_MouseHover(object sender, EventArgs e)

        {

            textBox30.Clear();

            textBox30.Text = "Кнопка \"Перевірка\"" + Environment.NewLine + "Виберіть рядок таблиці," + Environment.NewLine + "Натисніть на кнопку," + Environment.NewLine + "В результаті ви маєте отримати повідомлення про те чи товар поточного року випуску." + Environment.NewLine + "(Можна виділити декілька об'єктів, тобі вам потрібно буде вибрати перевірку чи всі об'єкти поточного року, чи хоча б один із вибраного).";

        }

        private void toolStripButton11\_MouseHover(object sender, EventArgs e)

        {

            textBox30.Text = "Кнопка \"Сховати\"" + Environment.NewLine + "При відображенні таблиці з результатами може виникнути потреба приховати її," + Environment.NewLine + "Якщо в вас з'явилась така потреба, можна натиснути на цю кнопку.";

        }

        private void toolStripButton12\_MouseHover(object sender, EventArgs e)

        {

            textBox30.Text = "Кнопка \"Найдорожчі\"" + Environment.NewLine + "Вам необхідно виділити рядки, які вас цікавлять," + Environment.NewLine + "потім натиснути кноку і ви отримаєте список товарів із нійбільшою ціною.";

        }

        // Арфметичні операції

        private void toolStripButton8\_Click(object sender, EventArgs e)

        {

            if (dataGridView1.SelectedRows.Count == 0)

                return;

            double add = -1, div = -1, sub = -1, mul = -1;

            if (!IsEmpty(textBox26))

                add = double.Parse(textBox26.Text);

            if (!IsEmpty(textBox27))

                sub = double.Parse(textBox27.Text);

            if(!IsEmpty(textBox28))

                mul = double.Parse(textBox28.Text);

            if(!IsEmpty(textBox29))

                div = double.Parse(textBox29.Text);

            foreach(DataGridViewRow row in dataGridView1.SelectedRows)

            {

                int index = row.Index;

                if(mul >= 0)

                    list[index] \*= mul;

                if(div > 0)

                    list[index] /= div;

                if(add > 0)

                    list[index] += add;

                if(sub > 0 && sub <= list[index].Price)

                    list[index] -= sub;

            }

            UpdateDataGridView1();

        }

       // Однакової ціни

        private void toolStripButton9\_Click(object sender, EventArgs e)

        {

            dt1.Clear();

            List <Appliances> find = new List<Appliances>();

            if (dataGridView1.SelectedRows.Count == 0)

                return;

            smallSize();

            toolStripButton11.Visible = true;

            foreach(DataGridViewRow row in dataGridView1.SelectedRows)

            {

                int index = row.Index;

               if (index >= list.Count)

                    continue;

                find.Add(list[index]);

            }

            deleteFromList(find);

            foreach(Appliances a in list)

                foreach(Appliances b in find)

                    if (a == b)

                        AddToTable(dt1, a);

            dataGridView2.DataSource = null;

            dataGridView2.DataSource = dt1;

        }

        private void deleteFromList(List <Appliances> l)

        {

            for(int i = 0; i < l.Count - 1; i++)

                for (int j = i +1; j  < l.Count; j++)

                    if (l[i] == l[j])

                    {

                        l.RemoveAt(j);

                        j--;

                    }

        }

        // Поточного року

        private void toolStripButton11\_Click(object sender, EventArgs e)

        {

            bigSize();

            toolStripButton11.Visible = false;

        }

        // Виправити цикл, бо коли вибирати останійелемент (пустий) і передонстанній, то результат не правильний

        private void toolStripButton10\_Click(object sender, EventArgs e)

        {

            int count = dataGridView1.SelectedRows.Count;

            if (count == 0)

                return;

            int index = dataGridView1.SelectedRows[0].Index;

            if (count == 1 && index < list.Count && index >= 0)

                if (list[index])

                    MessageBox.Show("Вибраний товар поточного року випуску.", "Порівняння", MessageBoxButtons.OK, MessageBoxIcon.Information);

                else

                    MessageBox.Show("Вибраний товар не поточного року випуску.", "Порівняння", MessageBoxButtons.OK, MessageBoxIcon.Information);

            else

            {

                MessageBoxMy messageBoxMy = new MessageBoxMy();

                DialogResult dr = messageBoxMy.ShowDialog();

                if (dr != DialogResult.Cancel)

                {

                    dt1.Clear();

                    smallSize();

                    toolStripButton11.Visible = true;

                    bool one = false, all = true;

                    for (int i = 0; i < dataGridView1.SelectedRows.Count; i++)

                    {

                        int indexRow = dataGridView1.SelectedRows[i].Index;

                        if (indexRow >= list.Count)

                            continue;

                        if (list[indexRow])

                            AddToTable(dt1, list[indexRow]);

                        if (i < dataGridView1.SelectedRows.Count - 1)

                        {

                            if (list[i] || list[i + 1])

                                one = true;

                            if (!(list[i] && list[i + 1]))

                                all = false;

                        }

                    }

                    dataGridView2.DataSource = null;

                    dataGridView2.DataSource = dt1;

                    if (dr == DialogResult.OK && dt1.Rows.Count > 0)

                    {

                        if (all)

                            MessageBox.Show("Всі елемент випущені поточного року", "Поівдомлення", MessageBoxButtons.OK, MessageBoxIcon.Information);

                        else

                            MessageBox.Show("Не всі елемент випущені поточного року", "Поівдомлення", MessageBoxButtons.OK, MessageBoxIcon.Information);

                    }

                    else if (dr == DialogResult.Yes)

                    {

                        if (one)

                            MessageBox.Show("Хоча б один елемент випущений поточного року", "Поівдомлення", MessageBoxButtons.OK, MessageBoxIcon.Information);

                        else

                            MessageBox.Show("Ні одного елементу немає поточного року", "Поівдомлення", MessageBoxButtons.OK, MessageBoxIcon.Information);

                    }

                    else

                        MessageBox.Show("Ні одного елементу немає поточного року", "Поівдомлення", MessageBoxButtons.OK, MessageBoxIcon.Information);

                }

            }

        }

        // Пошук максимального значення.

        private void toolStripButton12\_Click(object sender, EventArgs e)

        {

            if (dataGridView1.SelectedRows.Count == 0)

                return;

            List<Appliances> appliances = new List<Appliances>();

            foreach(DataGridViewRow row in dataGridView1.SelectedRows)

                appliances.Add(list[row.Index]);

            Appliances a = Max(appliances);

            dt1.Clear();

            foreach(Appliances app in appliances)

            {

                if (app == a)

                    AddToTable(dt1,app);

            }

            smallSize();

            toolStripButton11.Visible = true;

            dataGridView2.DataSource = null;

           dataGridView2.DataSource = dt1;

        }

        private Appliances Max(List<Appliances> appliances)

        {

            Appliances max = appliances[0];

            for(int i = 1; i < appliances.Count; i++)

            {

                if(max < appliances[i])

                    max = appliances[i];

            }

            return max;

        }

        // Виведення ціни.

        private void dataGridView1\_KeyPress(object sender, KeyPressEventArgs e)

        {

            if (char.ToUpper(e.KeyChar) == 'I')

            {

                DataGridView dataGridView = sender as DataGridView;

                int count = dataGridView.SelectedRows.Count;

                if (count == 0)

                    return;

                string all = "";

                foreach (DataGridViewRow row in dataGridView.SelectedRows)

                {

                    int index = row.Index;

                    double price = list[index];

                    string name = list[index];

                    all += name + " - " + price + "\n";

                }

                MessageBox.Show(all, "Інформація про вибрані товари", MessageBoxButtons.OK, MessageBoxIcon.Information);

            }

if(char.ToUpper(e.KeyChar) == 'P' || char.ToUpper(e.KeyChar) == 'M')

           {

               foreach(DataGridViewRow row in dataGridView1.SelectedRows)

               {

                   int index = row.Index;

                   if(index < 0)

                       continue;

                   if (char.ToUpper(e.KeyChar) == 'P')

                       list[index]++;

                   else

                       list[index]--;

               }

               UpdateDataGridView1();

           }

        }

    }

}

MessageBoxMy.cs:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace LabRob1

{

    public partial class MessageBoxMy : Form

    {

        bool all = false, one = false;

        public MessageBoxMy()

        {

            InitializeComponent();

        }

        private void button1\_Click(object sender, EventArgs e)

        {

            all = true;

            this.Close();

        }

        private void button2\_Click(object sender, EventArgs e)

        {

            one = true;

            this.Close();

        }

        private void MessageBoxMy\_FormClosed(object sender, FormClosedEventArgs e)

        {

            if(all)

                DialogResult = DialogResult.OK;

            else if(one)

                DialogResult= DialogResult.Yes;

            else

                DialogResult = DialogResult.Cancel;

        }

    }

}

Apliance.cs:

using Microsoft.SqlServer.Server;

using System;

using System.CodeDom;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Runtime.Remoting.Messaging;

using System.Text;

using System.Threading.Tasks;

namespace LabRob1

{

    public class Appliances

    {

        static int id = 1;

        public string Id { get; private set; }

        public string Name { get; set; }

        public string Brand { get; set; }

        public double Price { get; set; }

        public short Year { get; set; }

        public string EnergyClass { get; set; }

        public short Power { get; set; }

        public double EnergyCost { get; private set; }

        public Appliances(): this(id.ToString(),"Ім'я " + id, "Бренд " + id, 0, 0, "No",0) { }

        public Appliances(string Id, string Name, string Brand, double Price, short Year, string EnergyClass, short Power)

        {

            this.Id = Id;

            this.Name = Name;

            this.Brand = Brand;

            this.Price = Price;

            this.Year = Year;

            this.EnergyClass = EnergyClass;

            this.Power = Power;

            id++;

            CalculateCost();

        }

        public Appliances(Appliances other)

        {

            this.Id = other.Id;

            this.Name = other.Name;

            this.Brand = other.Brand;

            this.Price = other.Price;

            this.Year = other.Year;

            this.EnergyClass = other.EnergyClass;

            this.Power = other.Power;

            this.EnergyCost = other.EnergyCost;

        }

        // Вартість за годину роботи

        private void CalculateCost()

        {

            double priceKW = 3.25;

            EnergyCost = (double)Power \* priceKW /1000;

        }

        bool disc = false;

        public void Discount()

        {

            double discount = 0;

            switch (EnergyClass)

            {

                case "A+++": discount = 0.2;break;

                case "A++": discount = 0.1;break;

                case "A+": discount = 0.1;break;

            }

            if(disc == false)

            {

                disc = true;

                Price -= Price \* discount;

            }

        }

     // Оператори:

        public static Appliances operator+(Appliances a, double num)

        {

            return new Appliances(a) { Price = a.Price + num };

        }

        public static Appliances operator-(Appliances a, double num)

        {

            return new Appliances(a) { Price = a.Price - num };

        }

        public static Appliances operator \*(Appliances a, double num)

        {

            return new Appliances(a) { Price = a.Price \* num };

        }

        public static Appliances operator /(Appliances a, double num)

        {

            return new Appliances(a) { Price = a.Price / num };

        }

        // Логічні оператор для первірки того чи новий об'єкт.

        public static bool operator true(Appliances a)

        {

            int now = DateTime.Now.Year;

            return a.Year == now;

        }

        public static bool operator false(Appliances a)

        {

            int now = DateTime.Now.Year;

            return a.Year != now;

        }

        public static Appliances operator &(Appliances a1, Appliances a2)

        {

            int now = DateTime.Now.Year;

            if((a1.Year == now) && (a2.Year == now))

                return a1;

            return new Appliances();

        }

        public static Appliances operator |(Appliances a1, Appliances a2)

        {

            int now = DateTime.Now.Year;

            if (a1.Year == now)

                return a1;

            if (now == a2.Year)

                return a2;

            return new Appliances();

        }

        public static bool operator!(Appliances a)

        {

            return a.Year != DateTime.Now.Year;

        }

        // Переведенн типів, для того, щоб отримувати ціну Товару

        public static implicit operator double(Appliances a)

        {

            return a.Price;

        }

        public static implicit operator string(Appliances a)

        {

            return a.Name + " " + a.Brand;

        }

        // Перевантаження операторів порівняння:

        //      Щоб знаходти об'єкти із максимальною ціною (>,<) - Додати кнопку та опис кнопки

        //      Щоб знаходити всі товари із ціною вираного товару (==, !=)

        public static bool operator >(Appliances a1, Appliances a2)

        {

            return a1.Price > a2.Price;

        }

        public static bool operator <(Appliances a1, Appliances a2)

        {

            return a2 > a1;

        }

        public static bool operator ==(Appliances a1, Appliances a2)

        {

            return a1.Price == a2.Price;

        }

        public static bool operator !=(Appliances a1,Appliances a2)

        {

            return !(a1 == a2);

        }

public static Appliances operator ++(Appliances a)

       {

           a.Price += 1000;

           return a;

       }

       public static Appliances operator --(Appliances a)

       {

           a.Price -= 1000;

           return a;

       }

    }

}

Тестування:

|  |  |  |
| --- | --- | --- |
| № | Програма | Тестовий набір |
| 1 |  | Відкрито файл із тестовими наборами:  Змінено ціну першого товару на 1000  List[idex]+=  1000  Виконано зміну ціни  одночасно  декількома  операторами  (((  List[index]\*= 2,1) /=3) +=1000) -= 2000  Застосовано  Оператори до всіх товарів.  for(int i = 0; i <count; i++){  List[index]\*= 2,1) /=3) +=1000) -= 2000  } |
| 2 |  | Застосовано пошук по ціні другого елементу  Obj2 == Obj2  Застосовано пошук товарів, в яких ціна така як і в першого об’єкту  list[1] == list[1],  list[1] == list[5]  Застосовано пошук товарів, в яких ціна така як і вибраного  діапазону  list[1] == list[1],  list[1] == list[5],  list[2] == list[2],  list[3] == list[3]  list[4] == list[4] |
| 3 |  | Перевірено те чи товар поточного року.  list[3] = false  Перевірено чи всі товари поточного року та виведено товар, який поточного.  All = false;  list[3] = true;  Перевірено чи хоча б один товар поточного року.  One = true;  list[3] = true;  Зролено перевіркуна коректність роботи  list[5] = false |
| 4 |  | Виведено найдорожчий товар із списку  list[3] > list[1],  list[3] > list[2],  list[3] > list[4]  Виведено інформацію про вибраний список товарів. |
| 5 |  | Знову відкрито тестовий набір.  Застосовано оператор інкремену до 4 і 5 товарів.  list[4]++,  list[5]++.  Декілька разів засосовано оператор декрименту до 10 елементу.  List[10]-- |

Висновок: засвоєно перевантаження оператоів в C# та C++, виконано перевантаження бінарних, унарних, логічних операторів, операторів переведення типу, арифметичних операторів, операторів присвоєння та операторів потоку.