Міністерство освіти і науки України

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

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

Звіт

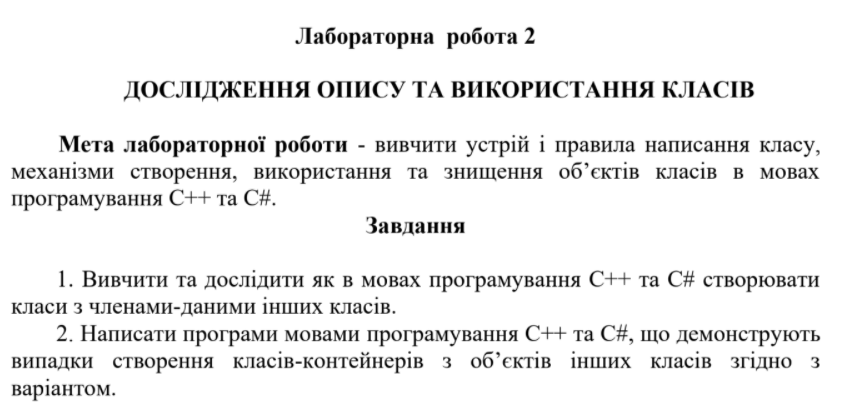
з лабораторної роботи № 2 з дисципліни

«Програмування - 2. Структури даних та алгоритми»

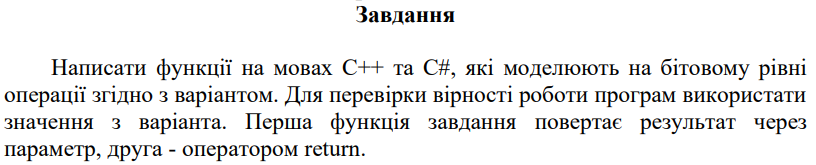
«ДОСЛІДЖЕННЯ ОПИСУ ТА ВИКОРИСТАННЯ КЛАСІВ У МОВАХ С++ ТА С#»

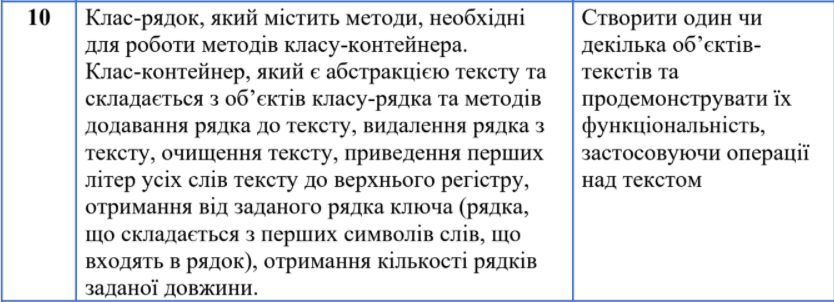
Виконав студент Кубай Д. І.

Київ 2021

****

**Варіант 10**





**Код на мові С++**

**file “Library.h”**

#pragma once

#include <iostream>

#include <string>

using namespace std;

class Line //lines class

{

public:

Line(); //default constructor

Line(char\* line); //own constructor

char\* GetLine(); //get the line

int GetLength(); //get lien length

private:

int \_length; //length

char\* \_arr; //line

int FindLength(char\* line); //search the length

};

class LineCollection //lines class - container

{

public:

LineCollection(Line line); //container

void AddLine(Line line); //add line to container

char\*\* GetLines(); //to print the container lines

int GetLinesNumber(); //return the number of lines

bool DeleteLine(int n); //deletes some of lines

void DeleteAll(); //deletes all lines

int HowMuch(int size); //number of the lines with some size

void ToUpRegister(); //gets first letters to upper register

char\*\* KeyWords(); //make key-words

private:

Line\* CopyArray(Line\* lines, int oldSize, int newSize); //copy array

int \_linesNumber = 0; //number of lines

Line\* \_myLines; //container of lines

};

char\* LineInput(); //string input

void PrintLines(LineCollection lines, string h); //print key -words

void PrintLines(LineCollection lines); //print lines

**file “Source.cpp”**

#include "Library.h"

int main()

{

cout << ("Input your first line:\n");

char\* line = LineInput(); //first line create

Line \_line = Line(line);

delete[] line;

LineCollection lines = LineCollection(\_line); //container create

bool skip = false;

cout << ("To finish the running press 0\n"); //menu

cout << ("To add new string press 1\n");

cout << ("To delete the string press 2\n");

cout << ("To delete all strings press 3\n");

cout << ("To find how much lines with some length the text has press 4\n");

cout << ("To get the first letter to upper register press 5\n");

cout << ("To make the key-words press 6\n");

cout << ("To stop the program press Enter\n");

cout << ("Select the operation:\n");

while (!skip) //operation choice

{

char key;

cin >> key;

cin.get();

int n = key - 48;

if (n == 0) //stop the running

{

skip = true;

break;

}

else if (n == 1) //add the line

{

cout << ("Write your line:\n");

char\* line = LineInput();

\_line = Line(line);

delete[] line;

lines.AddLine(\_line);

cout << ("\nYour lines:\n");

PrintLines(lines);

cout << ("\nSelect the operation:\n");

}

else if (n == 2) //delete some line

{

int number;

cout << ("Select the Line:\n");

cin >> number;

bool check = lines.DeleteLine(number);

if (!check) cout << ("The Line is Missing\n");

else

{

cout << ("\nYour lines:\n");

PrintLines(lines);

}

cout << ("\nSelect the operation:\n");

}

else if (n == 3) //deletes all lines

{

lines.DeleteAll();

cout << ("\nYour lines:\n");

PrintLines(lines);

cout << ("\nSelect the operation:\n");

}

else if (n == 4) //searching lines with some length

{

cout << ("Write the length of searching lines\n");

int number;

cin >> number;

number = lines.HowMuch(number);

cout << ("\nThe number of lines with that length: ") << number;

cout << ("\nSelect the operation:\n");

}

else if (n == 5) //gets first letters to upper register

{

lines.ToUpRegister();

cout << ("\nYour lines:\n");

PrintLines(lines);

cout << ("\nSelect the operation:\n");

}

else if (n == 6) //make key - words

{

cout << ("\nKey-words:\n");

PrintLines(lines, "keys");

cout << ("\nSelect the operation:\n");

}

}

}

**file “Library.cpp”**

#include "Library.h"

//////////////

/////Line

/////////////

Line::Line() //default constructor

{

\_arr = new char[1];

\_length = 0;

\_arr[0] = '\n';

}

Line::Line(char\* line) //own constructor

{

\_length = FindLength(line);

\_arr = new char[\_length + 1];

for (int j = 0; j <= \_length; j++)

{

\_arr[j] = line[j];

}

\_arr[\_length] = '\n';

}

char\* Line::GetLine() //get line

{

return \_arr;

}

int Line::GetLength() //get length

{

return \_length;

}

int Line::FindLength(char\* line) //search the length

{

int i = 0;

while (line[i] != '\n')

{

i++;

}

return i;

}

///////////

///LinesCollection

///////////

LineCollection::LineCollection(Line line) //container construct

{

\_linesNumber = 1;

\_myLines = new Line[1];

\_myLines[0] = line;

}

void LineCollection::AddLine(Line line) //add line to container

{

Line\* copy = new Line[\_linesNumber];

for (int i = 0; i < \_linesNumber; i++)

{

copy[i] = \_myLines[i];

}

\_linesNumber++;

\_myLines = new Line[\_linesNumber];

for (int i = 0; i < (\_linesNumber - 1); i++)

{

\_myLines[i] = copy[i];

}

\_myLines[\_linesNumber - 1] = line;

delete[] copy;

}

char\*\* LineCollection::GetLines() //to print lines

{

char\*\* text = new char\* [\_linesNumber];

for (int i = 0; i < \_linesNumber; i++)

{

int length = \_myLines[i].GetLength();

char\* str = \_myLines[i].GetLine();

text[i] = new char[length + 1];

for (int j = 0; j <= length; j++)

{

text[i][j] = str[j];

}

}

return text;

}

int LineCollection::GetLinesNumber() //lines number

{

return \_linesNumber;

}

bool LineCollection::DeleteLine(int n) //delete some line

{

if (n >= \_linesNumber)

{

return false;

}

else

{

\_linesNumber--;

for (int i = n; i < \_linesNumber; i++)

{

\_myLines[i] = \_myLines[i + 1];

}

Line\* copy = CopyArray(\_myLines, \_linesNumber, \_linesNumber);

delete[] \_myLines;

\_myLines = new Line[\_linesNumber];

\_myLines = CopyArray(copy, \_linesNumber, \_linesNumber);

delete[] copy;

return true;

}

}

void LineCollection::DeleteAll() //clear the container

{

delete[] \_myLines;

\_myLines = new Line[1];

\_linesNumber = 0;

}

int LineCollection::HowMuch(int size) //searhs how much lines with some size do we have

{

int k = 0;

for (int i = 0; i < \_linesNumber; i++)

{

if (size == \_myLines[i].GetLength()) k++;

}

return k;

}

void LineCollection::ToUpRegister() //changes first letter to upper register

{

for (int i = 0; i < \_linesNumber; i++)

{

int length = \_myLines[i].GetLength();

char\* str = \_myLines[i].GetLine();

for (int j = 0; j < length; j++)

{

int k = (str[j]);

if (j == 0 && k != ' ' && k > 96 && k < 123)

{

str[j] = (k - 32);

}

else if (j > 0 && (str[j - 1]) == ' ' && k != ' ' && k > 96 && k < 123)

{

str[j] = (k - 32);

}

}

}

}

char\*\* LineCollection::KeyWords() //make the key - words

{

char\*\* \_keyWords = new char\* [\_linesNumber];

for (int i = 0; i < \_linesNumber; i++)

{

int length = \_myLines[i].GetLength();

char\* str = \_myLines[i].GetLine();

int index = 0, size = 0;

for (int j = 0; j < length; j++)

{

int k = (str[j]);

if (j == 0 && k != ' ')

{

size++;

}

else if (j > 0 && (str[j - 1]) == ' ' && k != ' ')

{

size++;

}

}

\_keyWords[i] = new char[size + 1];

for (int j = 0; j < length; j++)

{

int k = (str[j]);

if (j == 0 && k != ' ')

{

\_keyWords[i][index] = str[j];

index++;

}

else if (j > 0 && (str[j - 1]) == ' ' && k != ' ')

{

\_keyWords[i][index] = str[j];

index++;

}

}

\_keyWords[i][size] = '\n';

}

return \_keyWords;

}

Line\* LineCollection::CopyArray(Line\* lines, int oldSize, int newSize) // makes a copy of array

{

Line\* copy = new Line[newSize];

for (int n = 0; n < oldSize; n++)

{

copy[n] = lines[n];

}

return copy;

}

///////////

///functions

///////////

char\* LineInput() //string input

{

char\* line = new char[1];

char\* copy = new char[1];

char a;

a = cin.get();

int i = 0;

while (a != '\n')

{

line = new char[i + 2];

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

{

line[j] = copy[j];

}

line[i] = a;

copy = new char[i + 1];

for (int j = 0; j < (i + 1); j++)

{

copy[j] = line[j];

}

i++;

a = cin.get();

}

line[i] = '\n';

return line;

}

void PrintLines(LineCollection lines) //lines print

{

char\*\* text = lines.GetLines();

int linesNumber = lines.GetLinesNumber();

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

{

int j = 0;

while (text[i][j] != '\n')

{

cout << (text[i][j]);

j++;

}

cout << endl;

}

}

void PrintLines(LineCollection lines, string h) //keywords print

{

char\*\* text = lines.KeyWords();

int linesNumber = lines.GetLinesNumber();

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

{

int j = 0;

while (text[i][j] != '\n')

{

cout << (text[i][j]);

j++;

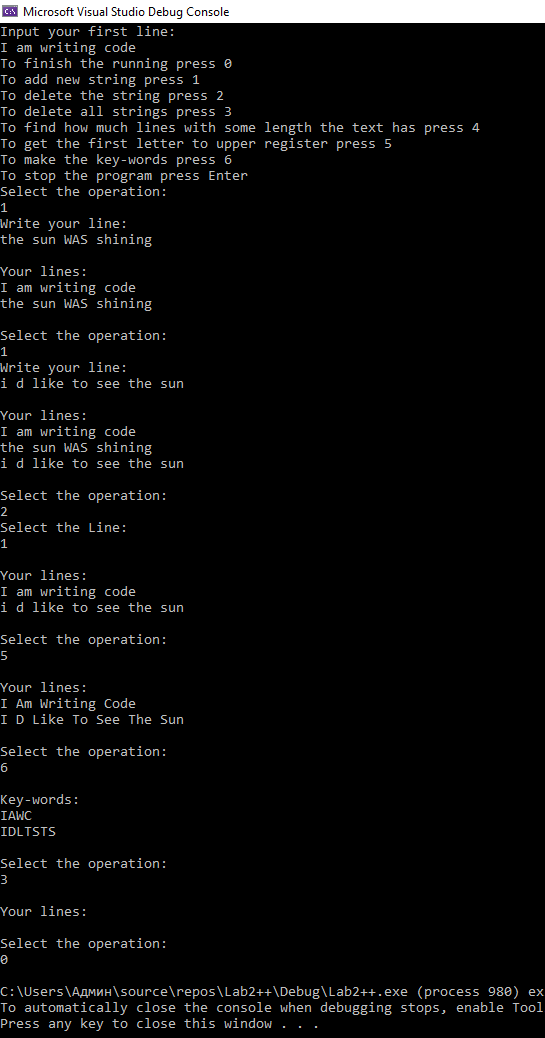
}

cout << endl;

}

}

**Вивід програми (C++)**

****

**Код на мові С#**

**file “Program.cs”**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using TextLibrary;

namespace Lab3

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Input your first line:");

char[] line = LineInput(); //input first line

Line \_line = new Line(line); //creating of the first line class instance

LineCollection lines = new LineCollection(\_line); //out container

bool skip = false;

Console.WriteLine("To finish the running press 0"); //menu

Console.WriteLine("To add new string press 1");

Console.WriteLine("To delete the string press 2");

Console.WriteLine("To delete all strings press 3");

Console.WriteLine("To find how much lines with some length the text has press 4");

Console.WriteLine("To get the first letter to upper register press 5");

Console.WriteLine("To make the key-words press 6");

Console.WriteLine("To stop the program press Enter");

Console.ReadLine();

while (!skip) //operation choice

{

string key;

Console.WriteLine("Select the operation:");

key = Console.ReadLine();

switch (key)

{

case "0": //finish the running press

skip = true;

break;

case "1":

Console.WriteLine("Write your line:"); //add new string

line = LineInput();

\_line = new Line(line);

lines.AddLine(\_line);

Console.WriteLine("Your lines:");

PrintLines(lines);

Console.WriteLine();

Console.ReadLine();

break;

case "2":

int n;

Console.WriteLine("Select the Line:"); //delete the string

n = Convert.ToInt32(Console.ReadLine());

bool check = lines.DeleteLine(n);

if (!check) Console.WriteLine("The Line is Missing");

else

{

Console.WriteLine("Your lines:");

PrintLines(lines);

}

Console.WriteLine();

break;

case "3": //delete all strings

lines.DeleteAll();

Console.WriteLine("Your lines:");

PrintLines(lines);

Console.WriteLine();

break;

case "4": //finds how much lines with some length the text has

Console.WriteLine("Write the length of searching lines");

n = Convert.ToInt32(Console.ReadLine());

n = lines.HowMuch(n);

Console.WriteLine("The text has {0} lines with that length", n);

Console.WriteLine();

break;

case "5":

lines.ToUpRegister(); //gets first letters to upper register

Console.WriteLine("Your lines:");

PrintLines(lines);

Console.WriteLine();

break;

case "6": //make key - words

Console.WriteLine("Key-words:");

PrintLines(lines, "keys");

Console.WriteLine();

break;

default: //defauls (stop)

skip = true;

break;

}

}

}

static public char[] LineInput() //string input

{

char[] line = new char[1];

char[] copy = new char[1];

char a;

a = Convert.ToChar(Console.Read());

int i = 0;

while (a != '\r')

{

line = new char[i + 2];

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

{

line[j] = copy[j];

}

line[i] = a;

copy = new char[i+1];

for(int j = 0; j < (i+1); j++)

{

copy[j] = line[j];

}

i++;

a = Convert.ToChar(Console.Read());

}

line[i] = '\0';

return line;

}

static public void PrintLines(LineCollection lines) //print lines

{

char[][] text = lines.GetLines();

int linesNumber = lines.GetLinesNumber();

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

{

int j = 0;

while(text[i][j] != '\0')

{

Console.Write(text[i][j]);

j++;

}

Console.WriteLine();

}

}

static public void PrintLines(LineCollection lines, string h) //print key-words

{

char[][] text = lines.KeyWords();

int linesNumber = lines.GetLinesNumber();

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

{

int j = 0;

while (text[i][j] != '\0')

{

Console.Write(text[i][j]);

j++;

}

Console.WriteLine();

}

}

}

}

**Library “TextLibrary”**

**Class “Line.cs”**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TextLibrary

{

public class Line //lines class

{

private int \_length; //length

private char[] \_arr; //array

public Line() //line default constructor

{

\_arr = new char[1];

\_length = 0;

\_arr[0] = '\0';

}

public Line(char[] line) //own line constructor

{

\_length = FindLength(line);

\_arr = new char[\_length + 1];

for(int j = 0; j <= \_length; j ++)

{

\_arr[j] = line[j];

}

\_arr[\_length] = '\0';

}

public char[] GetLine() //get the line

{

return \_arr;

}

public int GetLength() //get the length

{

return \_length;

}

private int FindLength(char[] line) //searchs the length

{

int i = 0;

while (line[i] != '\0')

{

i++;

}

return i;

}

}

}

**Class “LineCollection.cs”**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TextLibrary

{

public class LineCollection //class container for lines

{

private int \_linesNumber = 0; //number of lines

private Line[] \_myLines; //container

public LineCollection(Line line) //container

{

\_linesNumber = 1;

\_myLines = new Line[1];

\_myLines[0] = line;

}

public void AddLine(Line line) //adds line in container

{

Line[] copy = new Line[\_linesNumber];

for (int i = 0; i < \_linesNumber; i++)

{

copy[i] = \_myLines[i];

}

\_linesNumber++;

\_myLines = new Line[\_linesNumber];

for (int i = 0; i < (\_linesNumber - 1); i++)

{

\_myLines[i] = copy[i];

}

\_myLines[\_linesNumber - 1] = line;

}

public char[][] GetLines() //get lines from container

{

char[][] text = new char[\_linesNumber][];

for (int i = 0; i < \_linesNumber; i++)

{

int length = \_myLines[i].GetLength();

char[] str = \_myLines[i].GetLine();

text[i] = new char[length + 1];

for (int j = 0; j <= length; j++)

{

text[i][j] = str[j];

}

}

return text;

}

public int GetLinesNumber() //gets number of lines

{

return \_linesNumber;

}

public bool DeleteLine(int n) //delete some line

{

if(n >= \_linesNumber)

{

return false;

}

else

{

\_linesNumber--;

for (int i = n; i < \_linesNumber; i++)

{

\_myLines[i] = \_myLines[i + 1];

}

Line[] copy = CopyArray(\_myLines, \_linesNumber, \_linesNumber);

\_myLines = new Line[\_linesNumber];

\_myLines = CopyArray(copy, \_linesNumber, \_linesNumber);

return true;

}

}

public void DeleteAll() //delets all lines

{

\_myLines = new Line[1];

\_linesNumber = 0;

}

public int HowMuch(int size)

{

int k = 0;

for(int i = 0; i < \_linesNumber; i++)

{

if (size == \_myLines[i].GetLength()) k++;

}

return k;

}

public void ToUpRegister() //get first letters to upper register

{

for (int i = 0; i < \_linesNumber; i++)

{

int length = \_myLines[i].GetLength();

char[] str = \_myLines[i].GetLine();

for(int j = 0; j < length; j++)

{

int k = Convert.ToInt32(str[j]);

if (j == 0 && k != ' ' && k > 96 && k < 123)

{

str[j] = Convert.ToChar(k - 32);

}

else if(j > 0 && Convert.ToInt32(str[j-1]) == ' ' && k != ' ' && k > 96 && k < 123)

{

str[j] = Convert.ToChar(k - 32);

}

}

}

}

public char[][] KeyWords() //make key-words

{

char[][] \_keyWords = new char[\_linesNumber][];

for (int i = 0; i < \_linesNumber; i++)

{

int length = \_myLines[i].GetLength();

char[] str = \_myLines[i].GetLine();

int index = 0, size = 0;

for (int j = 0; j < length; j++)

{

int k = Convert.ToInt32(str[j]);

if (j == 0 && k != ' ')

{

size++;

}

else if (j > 0 && Convert.ToInt32(str[j - 1]) == ' ' && k != ' ')

{

size++;

}

}

\_keyWords[i] = new char[size + 1];

for (int j = 0; j < length; j++)

{

int k = Convert.ToInt32(str[j]);

if (j == 0 && k != ' ')

{

\_keyWords[i][index] = str[j];

index++;

}

else if (j > 0 && Convert.ToInt32(str[j - 1]) == ' ' && k != ' ')

{

\_keyWords[i][index] = str[j];

index++;

}

}

\_keyWords[i][size] = '\0';

}

return \_keyWords;

}

private Line[] CopyArray(Line[] lines, int oldSize, int newSize) //make the copy

{

Line[] copy = new Line[newSize];

for(int n = 0; n < oldSize; n++)

{

copy[n] = lines[n];

}

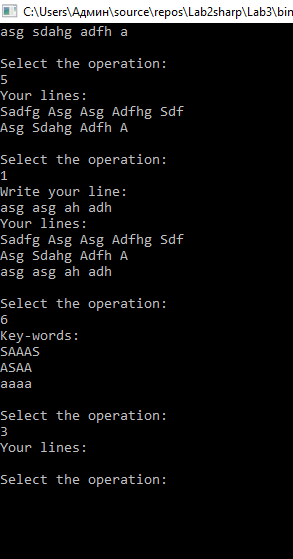
return copy;

}

}

}

**Вивід програми (C#)**

****

**Висновок**

Я створив програми на C++ і C#, що демонструють створення класів контейнерів інших класів. Програми зчитують рядок необмеженого розміру, створюють з ним екземпляр, який поміщається у клас-контейнер. Клас – контейнер виконує різні дії з текстом, як наприклад видалення певного рядка, заміна маленьких перших букв великими, створення слів – ключів і т. д..