

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота №** **5**

з дисципліни “Основи програмування”

тема “Рядки символів та файли”

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | Виконав  студент I курсу | | групи КП-01 | | Пецеля Артем Володимирович | | (*прізвище, ім’я, по батькові*) |   варіант №12 |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Гадиняк Руслан Анатолійович  (*прізвище, ім’я, по батькові*) |

Київ 2020

**Мета роботи**

Навчитися працювати з рядками символів.

Вміти визначати класи символів та використовувати функції для роботи з символами.

Навчитись використовувати стандартні функції для роботи з рядками символів.

Навчитися працювати з файловими потоками даних для зчитування і зберігання даних.

Використати формат даних CSV для зберігання даних програми на файловій системі.

**Постановка завдання**

*Загальні вимоги до завдання*

Створити консольну програму:

* Користувач вводить рядок із командою, що може починатись на підрядок: char, string або csv.
* Зациклити обробку команд так, щоби після виконання команди користувач міг ввести наступну команду. Спеціальна команда quit (або exit) перериває даний цикл.
* Якщо введена команда невідома - вивести користувачу помилку, у якій повторити введену ним команду і вказати, що вона для програми невідома.

*Частина 1. Символи*

Формат команд для символів: char:{class}

Замість {class} користувач задає один із перечислених класів символів:

all, upper, alpha, alnum

Якщо назва класу символів задана вірно - вивести всі ASCII символи відповідного класу у консоль (у довільному форматі).

Якщо користувач задав невірну назву класу - вивести відповідне повідомлення про помилку у консоль і вивести йому невірно введену назву.

*Частина 2. Рядки символів*

Всі команди для рядків символів повинні починатись зі слова string.

Всі команди даного підзавдання працюють зі значенням типу string (поточний рядок), яке за замовчуванням буде пустим рядком символів.

Команди (частини команди розділяти символом :):

1. print - вивести поточний рядок і довжину рядка символів.
2. set, {newString} - змінити значення поточного рядка на нововведене ({newString})
3. substr, {startIndex}, {length} - вивести підрядок поточного рядка із заданої позиції ({startIndex}) і заданої довжини ({length}).
4. upper - вивести такий рядок, у якому всі літери оригінального рядка замінені на відповідні літери у верхньому регістрі
5. contains, {otherString} - перевірити чи оригінальний рядок містить у собі підрядок ({otherString}) і вивести True/False

Якщо команду виконати неможливо (невірний ввід) - вивести користувачу відповідне повідомлення з описом причини.

*Частина 3. CSV*

Всі команди для CSV повинні починатись зі слова csv.

Всі команди даного підзавдання працюють зі значенням типу string, у який записується CSV текст та з масивом структур типу **Столиця країни**.

Команди:

1. load - зчитати CSV дані з файлу data.csv у таблицю рядків (дані збережені у файлі без екранування) і сформувати на основі таблиці *масив сутностей* типу **Столиця країни**.
2. text - вивести зчитаний з файлу CSV текст у консоль.
3. table - вивести отриману таблицю у консоль (форматування при виводі довільне).
4. entities - вивести масив сутностей у консоль (в довільному форматі).
5. get, {index} - отримати структуру з масиву за індексом {index} і вивести у консоль
6. set, {index}, {field}, {newValue} - отримати структуру з масиву за індексом {index} і змінити значення заданого поля {field} та нове {newValue}. Перетворити оновлений масив сутностей на таблицю, а таблицю на CSV текст і оновити значення таблиці і CSV тексту. Не дозволяти у {newValue} ввід символів , та ".
7. save - зберегти CSV текст в файл data.csv.

Якщо команду виконати неможливо (невірний ввід) - вивести користувачу відповідне повідомлення з описом причини.

**Аналіз вимог і проектування**

CSV-текст з даними столиць:

|  |
| --- |
| 1,Kyiv,Ukraine,2.884,826  2,Washington,USA,0.705,177  3,London,UK,8.892,1572  4,Paris,France,2.148,105  5,Berlin,Germany,3.769,892  6,Amsterdam,Holland,0.821,219  7,Minsk,Belarus,1.975,349  8,Moscow,Russia,11.92,2511  9,Sophia,Bulgaria,1.243,492 |

Таблиця рядків даних:



Структура даних “Столиці” була розроблена з такими полями:

1. id - ідентифікатор поля
2. name - назва столиці
3. country - країна, в якій це місто є столицею
4. population - кількість населення в мільйонах
5. square - площа міста в кілометрах квадратних.

**Тексти коду програм**

|  |
| --- |
| **Program.cs** |
| using System;  using static System.Console;  using static System.IO.File;  namespace lab5  {  class Program  {  struct Capital  {  public int id;  public string name;  public string country;  public double population;  public int square;  }  static string task2String = "";  static string task3CsvText = "";  static string[,] task3Table = new string[0,0];  static Capital[] task3Capitals = new Capital[0];  static void Main(string[] args)  {  bool exit = false;  do  {  Write("Enter command: ");  string command = ReadLine();  string[] subcommand = command.Split(':');  if (subcommand[0] == "char")  ProcessChar(command);  else if (subcommand[0] == "string")  ProcessString(command);  else if (subcommand[0] == "csv")  ProcessCsv(command);  else if (subcommand[0] == "exit")  exit = true;  else  WriteLine("Entered command '{0}' is invalid. Can't find this command.", command);  } while (exit == false);  WriteLine("Bye-bye!");  }  static void ProcessChar (string command)  {  string[] subcommand = command.Split(":");  if (subcommand.Length == 2)  {  if (subcommand[1] == "all")  ProcessCharAll();  else if (subcommand[1] == "upper")  ProcessCharUpper();  else if (subcommand[1] == "alpha")  ProcessCharAlpha();  else if (subcommand[1] == "alnum")  ProcessCharAlnum();  else  WriteLine("Entered command '{0}' is invalid. Can't find argument.", command);  }  else  {  WriteLine("Entered command '{0}' is invalid. Invalid arguments.", command);  }  }  static void ProcessCharAll()  {  WriteLine("Full ASCII table:");  for (int i = 0; i < 128; i++)  {  WriteLine("Code: {0} - Char: '{1}'", i, (char)i);  }  }  static void ProcessCharUpper()  {  WriteLine("ASCII upper letters:");  for (int i = 65; i < 91; i++)  {  WriteLine("Code: {0} - Char: '{1}'", i, (char)i);  }  }  static void ProcessCharAlpha()  {  WriteLine("ASCII alphabet:");  for (int i = 65; i < 91; i++)  {  WriteLine("Code: {0} - Char: '{1}'", i, (char)i);  WriteLine("Code: {0} - Char: '{1}'", i+32, (char)(i+32));  }  }  static void ProcessCharAlnum()  {  WriteLine("ASCII letters and nums:");  for (int i = 48; i < 58; i++)  {  WriteLine("Code: {0} - Char: '{1}'", i, (char)i);  }  for (int i = 65; i < 91; i++)  {  WriteLine("Code: {0} - Char: '{1}'", i, (char)i);  WriteLine("Code: {0} - Char: '{1}'", i+32, (char)(i+32));  }  }  static void ProcessString (string command)  {  string[] subcommand = command.Split(':');  if (subcommand.Length == 2)  {  if (subcommand[1] == "print")  ProcessStringPrint();  else if (subcommand[1] == "upper")  ProcessStringUpper();  else  WriteLine("Entered command '{0}' is invalid. Can't find argument.", command);  }  else if (subcommand.Length == 3)  {  if (subcommand[1] == "set")  ProcessStringSet(subcommand[2]);  else if (subcommand[1] == "contains")  ProcessStringContains(subcommand[2]);  else  WriteLine("Entered command '{0}' is invalid. Can't find argument.", command);  }  else if (subcommand.Length == 4)  {  if (subcommand[1] == "substr")  {  int start\_index;  int lenth;  if(int.TryParse(subcommand[2], out start\_index) && int.TryParse(subcommand[3], out lenth))  ProcessStringSubstr(start\_index, lenth);  else  WriteLine("Entered command '{0}' is invalid. Arguments must be integers", command);  }  else  WriteLine("Entered command '{0}' is invalid. Can't find argument.", command);  }  else  WriteLine("Entered command '{0}' is invalid. Invalid arguments.", command);  }  static void ProcessStringPrint()  {  WriteLine("String: '{0}'", task2String);  }  static void ProcessStringSet(string NewString)  {  task2String = NewString;  }  static void ProcessStringSubstr(int start\_index, int lenth)  {  if (task2String.Length < lenth)  WriteLine("String: '{0}'", task2String.Substring(start\_index));  else  WriteLine("Part of string: '{0}'", task2String.Substring(start\_index, lenth));  }  static void ProcessStringUpper()  {  WriteLine("String: '{0}'", task2String.ToUpper());  }  static void ProcessStringContains(string CheckStr)  {  if (task2String.Contains(CheckStr))  WriteLine("True");  else  WriteLine("False");  }  static void ProcessCsv (string command)  {  string[] subcommand = command.Split(':');  if (subcommand.Length == 2)  {  if (subcommand[1] == "load")  ProcessCsvLoad();  else if (subcommand[1] == "text")  ProcessCsvText();  else if (subcommand[1] == "table")  ProcessCsvTable();  else if (subcommand[1] == "capitals")  ProcessCsvCapitals();  else if (subcommand[1] == "save")  ProcessCsvSave();  else  WriteLine("Entered command '{0}' is invalid. Can't find argument.", command);  }  else if (subcommand.Length == 3)  if (subcommand[1] == "get")  {  int index;  if (int.TryParse(subcommand[2], out index) && index > 0)  ProcessCsvGet(index);  else  WriteLine("Entered command '{0}' is invalid. Invalid index", command);  }  else  WriteLine("Entered command '{0}' is invalid. Invalid arguments.", command);  else if (subcommand.Length == 5)  {  if (subcommand[1] == "set")  {  int index;  if (int.TryParse(subcommand[2], out index) && index > 0)  ProcessCsvSet(index, subcommand[3], subcommand[4]);  else  WriteLine("Entered command '{0}' is invalid. Invalid index", command);  }  else  WriteLine("Entered command '{0}' is invalid. Invalid arguments.", command);  }  else  WriteLine("Entered command '{0}' is invalid. Invalid arguments.", command);  }  static string[,] CsvToTable(string csvText)  {  string[] rows = csvText.Split("\r\n");  string[,] capitals = new string[rows.Length, 5];  for (int i = 0; i < rows.Length; i++)  {  string[] cols = rows[i].Split(',');  for (int j = 0; j < cols.Length; j++)  {  capitals[i,j] = cols[j];  }  }  return capitals;  }  static Capital[] TableToCapitals(string[,] csvTable)  {  int length = csvTable.GetLength(0);  Capital[] capitals = new Capital[length];  for (int i = 0; i < length; i++)  {  capitals[i] = new Capital {id = int.Parse(csvTable[i, 0]), name = csvTable[i, 1], country = csvTable[i, 2], population = double.Parse(csvTable[i, 3]), square = int.Parse(csvTable[i,4])};  }  return capitals;  }  static string[,] CapitalsToTable(Capital[] capitals)  {  string[,] csvTable = new string[capitals.Length, 5];  for (int i = 0; i < capitals.Length; i++)  {  csvTable[i, 0] = Convert.ToString(capitals[i].id);  csvTable[i, 1] = capitals[i].name;  csvTable[i, 2] = capitals[i].country;  csvTable[i, 3] = Convert.ToString(capitals[i].population);  csvTable[i, 4] = Convert.ToString(capitals[i].square);  }  return csvTable;  }  static string TableToCsv(string[,] csvTable)  {  string csvText = "";  int num\_cols = csvTable.GetLength(1);  for (int i = 0; i < csvTable.GetLength(0); i++)  {  for (int j = 0; j < num\_cols - 1; j++)  {  csvText = csvText + csvTable[i,j] + ",";  }  if (i != csvTable.GetLength(0) - 1)  csvText = csvText + csvTable[i,num\_cols - 1] + "\r\n";  else  csvText = csvText + csvTable[i,num\_cols - 1];  }  return csvText;  }  static void ProcessCsvLoad()  {  task3CsvText = ReadAllText("./data.csv");  task3Table = CsvToTable(task3CsvText);  task3Capitals = TableToCapitals(task3Table);  }  static void ProcessCsvText()  {  WriteLine("CSV text:");  WriteLine(task3CsvText);  }  static void ProcessCsvTable()  {  WriteLine("CSV Table: ");  WriteLine("{0,3} | {1,20} | {2,20} | {3,20} | {4,20}", "id", "name", "country", "population(mln)", "square(km^2)");  for (int i = 0; i <= 94; i++)  Write("-");  WriteLine();  for (int i = 0; i < task3Table.GetLength(0); i++)  {  WriteLine("{0,3} | {1,20} | {2,20} | {3,20} | {4,20}", task3Table[i,0], task3Table[i,1], task3Table[i,2], task3Table[i,3], task3Table[i,4]);  }  }  static void ProcessCsvCapitals()  {  WriteLine("Capitals: ");  int length = task3Capitals.Length;  for (int i = 0; i < length - 1; i++)  {  Write("{0}, ", task3Capitals[i].name);  }  WriteLine(task3Capitals[length - 1].name + ".");  }  static void ProcessCsvGet(int index)  {  foreach (Capital capital in task3Capitals)  if (capital.id == index)  {  WriteLine("Data for capital:");  WriteLine("{0}. {1} is the capital of {2}. The population is {3} million people. Square is {4} km^2.", capital.id, capital.name, capital.country, capital.population, capital.square);  break;  }  }  static void ProcessCsvSet(int index, string field, string value)  {  for (int i = 0; i < task3Capitals.Length; i++)  {  Capital capital = task3Capitals[i];  if (capital.id == index)  {  if (field == "id")  capital.id = int.Parse(value);  else if (field == "name")  capital.name = value;  else if (field == "country")  capital.country = value;  else if (field == "population")  capital.population = double.Parse(value);  else if (field == "square")  capital.square = int.Parse(value);  task3Capitals[i] = capital;  task3Table = CapitalsToTable(task3Capitals);  task3CsvText = TableToCsv(task3Table);  break;  }  }  }  static void ProcessCsvSave()  {  WriteAllText("./data.csv", task3CsvText);  task3CsvText = "";  task3Table = new string[0,0];  task3Capitals = new Capital[0];  }  }  } |

**Приклади результатів**

Правильне виконання програми:

|  |  |
| --- | --- |
| Приклад команди | Реакція програми |
| char:upper |  |
| char:alpha |  |
| string:print |  |
| string:contains:3:5 |  |
| csv:table |  |
| csv:get:2 |  |

Неправильне виконання програми:

|  |  |
| --- | --- |
| Приклад команди | Реакція програми |
| string:something |  |
| chor:alnum |  |
| csv:get:10 |  |

**Висновки**

Виконавши дану лабораторну роботу було створено консольну утиліту, яка обробляє команди користувача та реагує на них відповідним чином. Під час виконання була відпрацьована робота з рядками символів, типами даних string та char, і відповідно їх масивами. Також була застосована на практиці робота з файлами, а саме з csv - файлами.

Компіляція всього коду відбувалася за допомогою утиліти dotnet.