

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  Беліцький Олександр Сергійович  (*прізвище, ім’я, по батькові*)  варіант № 3 |  | 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**

**Частина 1. Символи**

Формат команд для символів: char+{class}

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

all, lower, number, punct

Команди:

char+all

char+lower

char+number

char+punct

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. lower - вивести такий рядок, у якому всі літери оригінального рядка замінені на відповідні літери у нижньому регістрі
5. contains, {char} - перевірити чи оригінальний рядок містить у собі символ ({char}) і вивести True/False

Команди:

string+print

string+set+A new string!

string+substr+startIndex+length

string+lower

string+contains+char

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 **Викладач**.

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 **Викладач**.
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.

Приклади команд:

csv+load

csv+text

csv+table

csv+teachers

csv+get+index

csv+set+index+field+newValue

csv+save

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

**Аналіз вимог і проектування**

**СSV-текст сутності Викладачі**

|  |
| --- |
| id,fullname,subject,age  1,Sayori Sage,english,33  2,Victor Green,literature,54  3,Mary Poppins,ethics,27  4,Morgan Freeman,physics,66  5,Joseph Haydn,music,41 |

**CSV таблиця рядків даних**

|  |  |  |  |
| --- | --- | --- | --- |
| id | fullname | subject | age |
| 1 | Sayori Sage | english | 33 |
| 2 | Victor Green | literature | 54 |
| 3 | Mary Poppins | ethics | 27 |
| 4 | Morgan Freeman | physics | 66 |
| 5 | Joseph Haydn | music | 41 |

**Текстовий опис структури:**

Сутність - Викладач

id - індивідуальний номер

fullname - Ім'я та призвище викладача

subject - предмет викладання

age - вік викладача

struct Teacher

public int id

public string fullname

public string subject

public int age

**Тексти коду програм**

|  |
| --- |
| **Program.cs** |
| using System;  using static System.Console;  using static System.IO.File;  namespace lab\_5  {  struct Teacher  {  public int id;  public string fullname;  public string subject;  public int age;  }  class Program  {  /\*  char+all  char+lower  char+number  char+punct    string+print  string+set+A new string!  string+substr+startIndex+length  string+lower  string+contains+char  csv+load  csv+text  csv+table  csv+teachers  csv+get+index  csv+set+index+field+newValue  csv+save  \*/  static string task2String = "Lord, save and protect.";  static string task3CsvText = "";  static string[,] task3Table = new string[0, 0];  static Teacher[] task3Teachers = new Teacher[0];  static void Main(string[] args)  {  WriteLine("==== Info Table ====");  WriteLine("Char commands: char+all, char+number, char+lower, char+punct");  WriteLine("String commands: string+print, string+set+a new string!, string+substr+startIndex+length, string+lower, string+contains+char");  WriteLine("Csv commands: csv+load, csv+text, csv+table, csv+teachers, csv+get+index, csv+set+index+field+newValue, csv+save");  WriteLine("To end program execution: exit");  while (true)  {  WriteLine();  Write("Enter command: ");  string command = ReadLine();  WriteLine("`{0}`", command);  string[] subcommands = command.Split('+');  if (subcommands[0] == "char")  {  ProcessChar(subcommands);  }  else if (subcommands[0] == "string")  {  ProcessString(subcommands);  }  else if (subcommands[0] == "csv")  {  ProcessCsv(subcommands);  }  else if (subcommands[0] == "exit")  {  break;  }  else  {  WriteLine("Unknown command `{0}`", subcommands[0]);  }  }  WriteLine("Ending of processing");  }  //Char block  static void ProcessChar(string[] subcommands)  {  if (subcommands.Length != 2)  {  WriteLine("Error: should be 2 parts in char commands, but have: {0}", subcommands.Length);  return;  }  if (subcommands[1] == "all")  {  PrintAllChars();  }  else if (subcommands[1] == "lower")  {  PrintLowerLetters();  }  else if (subcommands[1] == "number")  {  PrintCharNumb();  }  else if (subcommands[1] == "punct")  {  PrintCharPunct();  }  else  {  WriteLine("Error: Unknown char command `{0}`", subcommands[1]);  return;  }  }  static void PrintAllChars()  {  for (int code = 0; code < 128; code++)  {  WriteLine("{0} '{1}'", code, (char)code);  }  }  static void PrintLowerLetters()  {  for (int code = 97; code <= 122; code++)  {  WriteLine("{0} '{1}'", code, (char)code);  }  }  static void PrintCharNumb()  {  for (int code = 48; code <= 57; code++)  {  WriteLine("{0} '{1}'", code, (char)code);  }  }  static void PrintCharPunct()  {  for (int code = 33; code <= 47; code++)  {  WriteLine("{0} '{1}'", code, (char)code);  }  for (int code = 58; code <= 64; code++)  {  WriteLine("{0} '{1}'", code, (char)code);  }  for (int code = 91; code <= 96; code++)  {  WriteLine("{0} '{1}'", code, (char)code);  }  for (int code = 123; code <= 126; code++)  {  WriteLine("{0} '{1}'", code, (char)code);  }  }  //String block  static void ProcessString(string[] subcommands)  {  if ((subcommands.Length != 2) && (subcommands.Length != 3) && (subcommands.Length != 4))  {  WriteLine("Error: must be 2-4 parts in string commands, but have: {0}", subcommands.Length);  return;  }  if (subcommands[1] == "substr")  {  if (subcommands.Length != 4)  {  WriteLine("Error: must be 4 parts in substr command, but have: {0}", subcommands.Length);  return;  }  bool ifStartIndex = Int32.TryParse(subcommands[2], out int startIndex);  if (ifStartIndex)  {  bool ifLength = Int32.TryParse(subcommands[3], out int Length);  if (ifLength)  {  PrintSubString(startIndex, Length);  }  else  {  WriteLine("Error: Substr length must be an integer number, but have: {0}", subcommands[3]);  }  }  else  {  WriteLine("Error: Substr startIndex must be an integer number, but have: {0}", subcommands[2]);  }  }  else if (subcommands[1] == "print")  {  if (subcommands.Length != 2)  {  WriteLine("Error: must be 2 parts in string print command, but have: {0}", subcommands.Length);  return;  }  PrintString2Task();  }  else if (subcommands[1] == "set")  {  if (subcommands.Length != 3)  {  WriteLine("Error: must be 3 parts in string set command, but have: {0}", subcommands.Length);  return;  }  ChangeString(subcommands[2]);  }  else if (subcommands[1] == "lower")  {  if (subcommands.Length != 2)  {  WriteLine("Error: must be 2 parts in string lower command, but have: {0}", subcommands.Length);  return;  }  StringToLower();  }  else if (subcommands[1] == "contains")  {  if (subcommands.Length != 3)  {  WriteLine("Error: must be 3 parts in string contains command, but have: {0}", subcommands.Length);  return;  }  if (subcommands[2].Length != 1)  {  WriteLine("Error: must be 1 char in string contains command, but have: {0}", subcommands[2].Length);  return;  }  ContainChar(subcommands[2]);  }  else  {  WriteLine("Error: Unknown string command `{0}`", subcommands[1]);  return;  }  }  static void PrintSubString(int startIndex, int Length)  {  if (Length < 0)  {  WriteLine("Error: Substr length is negative ");  return;  }  if (startIndex < 0)  {  WriteLine("Error: Substr startIndex is negative ");  return;  }  string s = task2String;  if (startIndex > s.Length)  {  WriteLine("Error: Substr startIndex is more than length of string ");  return;  }  if ((s.Length - startIndex) < Length)  {  WriteLine("Error: Substr length is more than length of string");  return;  }  s = task2String.Substring(startIndex, Length);  WriteLine("Substring is: `{0}`", s);  }  static void PrintString2Task()  {  Write("Current string: `{0}`, its length is {1}", task2String, task2String.Length);  }  static void ChangeString(string newString)  {  task2String = newString;  }  static void StringToLower()  {  Write(task2String.ToLower());  }  static void ContainChar(string symbol)  {  if (task2String.Contains(symbol))  {  WriteLine("Current string `{0}` contains char `{1}` ", task2String, symbol);  }  else  {  WriteLine("Current string `{0}` doesn't contain char `{1}` ", task2String, symbol);  }  }  //csv block  static void ProcessCsv(string[] subcommands)  {  if (subcommands[1] == "load")  {  LoadCsv();  }  else if (subcommands[1] == "text")  {  WriteCsvText();  }  else if (subcommands[1] == "table")  {  task3Table = CsvToTable();  PrintTable(task3Table);  }  else if (subcommands[1] == "teachers")  {  PrintTeachers();  }  else if (subcommands[1] == "get")  {  Getprocess(subcommands);  }  else if (subcommands[1] == "set")  {  SetProcess(subcommands);  }  else if (subcommands[1] == "save")  {  SaveCsv();  }  else  {  WriteLine("Error: Unknown csv command");  }  }  static void LoadCsv()  {  task3CsvText = ReadAllText("./data.csv");  CsvToTeacher();  WriteLine("Tip: Csv table was uploaded");  }  static void CsvToTeacher()  {  string[] lines = ReadAllLines("./data.csv");  int i = 0;  Array.Resize(ref task3Teachers, task3Teachers.Length + lines.Length);  foreach (string line in lines)  {  string[] items = line.Split(',');  task3Teachers[i].id = int.Parse(items[0]);  task3Teachers[i].fullname = items[1];  task3Teachers[i].subject = items[2];  task3Teachers[i].age = int.Parse(items[3]);  i++;  }  task3Table = CsvToTable();  }  static void PrintTeachers()  {  string[] lines = ReadAllLines("./data.csv");  WriteLine("{0,3} ║ {1,-15} ║ {2,-12} ║ {3,4}║", "id", "fullname", "subject", "age");  for (int i = 0; i < 44; i++)  {  Write("~");  }  WriteLine();  for (int j = 0; j < lines.Length; j++)  {  WriteLine("{0,3} ║ {1,-15} ║ {2,-12} ║ {3,4}║", task3Teachers[j].id, task3Teachers[j].fullname, task3Teachers[j].subject, task3Teachers[j].age);  }  }  static void WriteCsvText()  {  WriteLine(task3CsvText);  }  static string[,] CsvToTable()  {  string[] lines = task3CsvText.Split("\r\n");  string[] column = lines[0].Split(",");  string[,] Table = new string[lines.Length, column.Length];  for (int i = 0; i < Table.GetLength(0); i++)  {  Table[i, 0] = task3Teachers[i].id.ToString();  Table[i, 1] = task3Teachers[i].fullname;  Table[i, 2] = task3Teachers[i].subject;  Table[i, 3] = task3Teachers[i].age.ToString();  }  return Table;  }  static void PrintTable(string[,] Table)  {  WriteLine("{0,3} | {1,-15} | {2,-12} | {3,4}|", "id", "fullname", "subject", "age");  for (int i = 0; i < 44; i++)  {  Write("-");  }  WriteLine();  for (int i = 0; i < Table.GetLength(0); i++)  {  WriteLine("{0,3} | {1,-15} | {2,-12} | {3,4}|", task3Table[i, 0], task3Table[i, 1], task3Table[i, 2], task3Table[i, 3]);  }  }  static void Getprocess(string[] subcommands)  {  bool ifIndex = Int32.TryParse(subcommands[2], out int index);  if ((ifIndex) && (index >= 0))  {  if (index > task3Teachers.Length)  {  WriteLine("Error: {index} is out of range");  return;  }  WriteLine("{0,3} ║ {1,-15} ║ {2,-12} ║ {3,4}║", "id", "fullname", "subject", "age");  for (int i = 0; i < 44; i++)  {  Write("-");  }  WriteLine();  for (int i = 0; i < task3Teachers.Length; i++)  {  if (i == index)  {  WriteLine("{0,3} ║ {1,-15} ║ {2,-12} ║ {3,4}║", task3Teachers[i].id, task3Teachers[i].fullname, task3Teachers[i].subject, task3Teachers[i].age);  return;  }  }  }  else  {  WriteLine("Error: {index} must be a integer positive (>=0) number");  }  }  static void SetProcess(string[] subcommands)  {  if (subcommands.Length != 5)  {  WriteLine("Error: must be 5 parts in csv+set command, but have: {0}", subcommands.Length);  return;  }  bool ifIndex = Int32.TryParse(subcommands[2], out int index);  if ((ifIndex) && (index >= 0))  {  if (index < task3Table.GetLength(0))  {  if (subcommands[3] == "id")  {  bool ifNum = Int32.TryParse(subcommands[4], out int num);  if ((ifNum) && (num > 0))  {  for (int i = 0; i < task3Teachers.Length; i++)  {  if (num == task3Teachers[i].id)  {  WriteLine("Error: new id must be original");  return;  }  }  for (int i = 0; i < task3Teachers.Length; i++)  if (i == index)  {  task3Teachers[i].id = num;  WriteLine("Tip: Id was changed");  task3Table[i, 0] = subcommands[4];  TableToText();  return;  }  }  else  {  WriteLine("Error: new value of id must be positive integer");  return;  }  }  else if (subcommands[3] == "fullname")  {  foreach (char item in subcommands[4])  {  if ((subcommands[4].Contains('"')) || (subcommands[4].Contains(',')))  {  WriteLine("Error: must not use \" and ,");  return;  }  }  for (int i = 0; i < task3Teachers.Length; i++)  if (i == index)  {  task3Teachers[i].fullname = subcommands[4];  WriteLine("Tip: Fullname was changed");  task3Table[i, 1] = subcommands[4];  TableToText();  return;  }  }  else if (subcommands[3] == "subject")  {  foreach (char item in subcommands[4])  {  if ((subcommands[4].Contains('"')) || (subcommands[4].Contains(',')))  {  WriteLine("Error: must not use \" and ,");  return;  }  }  for (int i = 0; i < task3Teachers.Length; i++)  if (i == index)  {  task3Teachers[i].subject = subcommands[4];  WriteLine("Tip: Subject was changed");  task3Table[i, 2] = subcommands[4];  TableToText();  return;  }  }  else if (subcommands[3] == "age")  {  bool ifNum = Int32.TryParse(subcommands[4], out int num);  if ((ifNum) && (num > 0))  {  for (int i = 0; i < task3Teachers.Length; i++)  if (i == index)  {  task3Teachers[i].age = num;  WriteLine("Tip: Age was changed");  task3Table[i, 3] = subcommands[4];  TableToText();  return;  }  }  else  {  WriteLine("Error: new value of age must be positive integer");  return;  }  }  else  {  WriteLine("Error: can not found such field in table, but have `{0}`", subcommands[2]);  return;  }  }  else  {  WriteLine("Error: Teacher under index {0} doesn't exist", index);  return;  }  }  else  {  WriteLine("Error: index must be integer positive (>=0) number`{0}`", subcommands[2]);  return;  }  }  static void TableToText()  {  string[] lines = new string[task3Table.GetLength(0)];  for (int i = 0; i < task3Table.GetLength(0); i++)  {  string[] item = new string[4];  for (int j = 0; j < 4; j++)  {  item[j] = task3Table[i, j];  }  string line = string.Join(",", item);  lines[i] = line;  }  task3CsvText = string.Join("\r\n", lines);  }  static void SaveCsv()  {  WriteAllText("./data.csv", task3CsvText);  WriteLine("Tip: csv was updated");  }  }  } |

**Приклади результатів**

Результати виклику команд:

|  |
| --- |
| char+all |
| 0 '' 1 '' 2 '' 3 '' 4 '' 5 '' 6 '' 7 '' 8 ' 9 ' ' 10 ' ' 11 ' ' 12 ' ' '3 ' 14 '' 15 '' 16 '' 17 '' 18 '' 19 '' 20 '' 21 '' 22 '' 23 '' 24 '' 25 '' 26 '' 27 ' 8 '' 29 '' 30 '' 31 '' 32 ' ' 33 '!' 34 '"' 35 '#' 36 '$' 37 '%' 38 '&' 39 ''' 40 '(' 41 ')' 42 '\*' 43 '+' 44 ',' 45 '-' 46 '.' 47 '/' 48 '0' 49 '1' 50 '2' 51 '3' 52 '4' 53 '5' 54 '6' 55 '7' 56 '8' 57 '9' 58 ':' 59 ';' 60 '<' 61 '=' 62 '>' 63 '?' 64 '@' 65 'A' 66 'B' 67 'C' 68 'D' 69 'E' 70 'F' 71 'G' 72 'H' 73 'I' 74 'J' 75 'K' 76 'L' 77 'M' 78 'N' 79 'O' 80 'P' 81 'Q' 82 'R' 83 'S' 84 'T' 85 'U' 86 'V' 87 'W' 88 'X' 89 'Y' 90 'Z' 91 '[' 92 '\' 93 ']' 94 '^' 95 '\_' 96 '`' 97 'a' 98 'b' 99 'c' 100 'd' 101 'e' 102 'f' 103 'g' 104 'h' 105 'i' 106 'j' 107 'k' 108 'l' 109 'm' 110 'n' 111 'o' 112 'p' 113 'q' 114 'r' 115 's' 116 't' 117 'u' 118 'v' 119 'w' 120 'x' 121 'y' 122 'z' 123 '{' 124 '|' 125 '}' 126 '~' 127 '' |
| char+lower |
| 97 'a' 98 'b' 99 'c' 100 'd' 101 'e' 102 'f' 103 'g' 104 'h' 105 'i' 106 'j' 107 'k' 108 'l' 109 'm' 110 'n' 111 'o' 112 'p' 113 'q' 114 'r' 115 's' 116 't' 117 'u' 118 'v' 119 'w' 120 'x' 121 'y' 122 'z' |
| char+number |
| 48 '0' 49 '1' 50 '2' 51 '3' 52 '4' 53 '5' 54 '6' 55 '7' 56 '8' 57 '9' |
| char+punct |
| 33 '!' 34 '"' 35 '#' 36 '$' 37 '%' 38 '&' 39 ''' 40 '(' 41 ')' 42 '\*' 43 '+' 44 ',' 45 '-' 46 '.' 47 '/' 58 ':' 59 ';' 60 '<' 61 '=' 62 '>' 63 '?' 64 '@' 91 '[' 92 '\' 93 ']' 94 '^' 95 '\_' 96 '`' 123 '{' 124 '|' 125 '}' 126 '~' |
| string+print |
| Current string: `Lord, save and protect.`, its length is 23 |
| string+set+a new line |
| Enter command: string+set+The work is done  Tip: string was updated |
| string+substr+startIndex+length |
| Enter command: string+substr+0+8  Substring is: The work |
| string+lower |
| Enter command: string+lower  the work is done |
| string+cintains+char |
| Enter command: string+contains+w  Current string `The work is done contains` char `w` |
| csv+load |
| Enter command: csv+load  Tip: Csv table was uploaded |
| csv+text |
| Enter command: csv+text  1,Sayori Sage,english,33  2,Victor Green,literature,54  3,Mary Poppins,ethics,27  4,Morgan Freeman,physics,66  5,Joseph Haydn,music,41 |
| csv+table |
|  |
| csv+teachers |
|  |
| csv+get+index |
|  |
| csv++set+index+field+newValue |
| Enter command: csv+set+1+id+22  Tip: Id was changed  //csv+get+1 для перевірки |
| csv+save |
| Enter command: csv+save  Tip: csv was updated |

**Результат, коли неправильно введені вхідні дані:**

|  |
| --- |
| char+ss, string+ss, csv+ss |
| Enter command: char+sss  Error: Unknown char command  Enter command: string+ss  Error: Unknown string command  Enter command: csv+ss  Error: Unknown csv command |
| char+all+a |
| Enter command: char+all+a  Error: should be 2 parts in char commands, but have: 3 |
| string+sss+ss+ss+s |
| Enter command: string+sss+ss+ss+s  Error: must be 2-4 parts in string commands, but have: 5 |
| string+substr+ss+ss+s |
| Enter command: string+substr+ss+ss+s  Error: must be 2-4 parts in string commands, but have: 5 |
| string+substr+Nan+Nan (startIndex,length<0,startIndex>length,  startIndex,length>string length) |
| Enter command: string+substr+t+4  Error: Substr startIndex must be an integer number, but have: t  Enter command: string+substr+5+t  Error: Substr length must be an integer number, but have: t  Enter command: string+substr+-4+4  Error: Substr startIndex is negative  Enter command: string+substr+4+-3  Error: Substr length is negative  Enter command: string+substr+6+111  Error: Substr length is more than length of string  Enter command: string+substr+1000+2  Error: Substr startIndex is more than length of string |
| string+print+ss |
| Enter command: string+print+ss  Error: must be 2 parts in string print command, but have: 3 |
| string+set+ss+s |
| Enter command: string+set+ss+s  Error: must be 3 parts in string set command, but have: 4 |
| string+lower+s |
| Enter command: string+lower+s  Error: must be 2 parts in string lower command, but have: 3 |
| string+contains+s+s |
| Enter command: string+contains+s+s  Error: must be 3 parts in string contains command, but have: 4 |
| string+contains+ss |
| Enter command: string+contains+ss  Error: must be 1 char in string contains command, but have: 2 |
| csv+get+Nan(or num out of range) |
| Enter command: csv+get+t  Error: {index} must be a integer positive (>=0) number  Enter command: csv+get+333  Error: {index} is out of range |
| csv+set+4+4+4+4 |
| Enter command: csv+set+4+4+4+4  Error: must be 5 parts in csv+set command, but have: 6 |
| csv+set+Nan(or negative num)+id+Nan(or negative num) |
| Enter command: csv+set+ff+id+4  Error: index must be integer positive (>=0) number`ff`  Enter command: csv+set+-1+id+3  Error: index must be integer positive (>=0) number`-1`  Enter command: csv+set+1+id+yy  Error: new value of id must be positive integer  Enter command: csv+set+1+id+-33  Error: new value of id must be positive integer |
| csv+set+1+unknownfield+2 |
| Enter command: csv+set+1+gg+2  Error: can not found such field in table, but have gg |
| csv+set+1+fullname+(contains " or ,) |
| Enter command: csv+set+1+fullname+Ge,"orge  Error: must not use " and , |
| csv+set+1+id+not original id |
| Enter command: csv+set+1+id+2  Error: new id must be original |

**Висновки**

Поряд з класами структури представляють ще один спосіб створення власних типів даних в C #. Більш того багато примітивних типів, наприклад, int, double і т.д., по суті є структурами. Тип структури - це тип значення, який може инкапсулювати дані і пов'язані функції. Для визначення типу структури використовується ключове слово ***struct***.

CSV (від англ. comma-separated values ‘значення, розділені комою’) — файловий формат, котрий є відмежовувальним форматом для представлення табличних даних, у якому поля відокремлюються символом коми та переходу на новий рядок. Поля, що містять коми, декілька рядків, або лапки (позначаються подвійними лапками), мають обмежуватися з обох боків лапками.

Формат CSV використовується для перенесення даних між базами даних та програмами — редакторами електронних таблиць.  
 Функції можна назвати невеликими підпрограммами, куди можна винести повторюваний код і звертатися до нього, коли це буде потрібно. Функції значно полегшують побудову програм, так як нам не треба копіювати однотипний код безліч разів, а можна просто скористатися однією спільною функцією.

Для створення функцій необхідно вказати тип даних, що повертає функція, вказати назву і параметри. У разі коли функція нічого не повертає, то вказується тип даних void.