

МIНIСТЕРСТВО ОСВIТИ 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 “Символи та рядки символів”

|  |  |  |
| --- | --- | --- |
| Виконав(ла)  студент(ка) I курсу  групи КП-92  Мовчан Максим  (*прізвище, ім’я, по батькові*)  варіант №9 |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 2019 р.  викладач  Гадиняк Руслан Анатолійович  (*прізвище, ім’я, по батькові*) |

Київ 2019

**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 обробляти ввід користувача у вигляді консольних команд.

**Постановка завдання**

* Користувач вводить рядок із командою. Якщо команда починається на підрядок:
  + char - вона відноситься до першого підзавдання,
  + string - до другого,
  + strings - до третього.
* 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.

Підзавдання 1. Класи символів

Всі команди даного підзавдання повинні починатись зі слова char.  
Формат команди: char {class} Замість {class} користувач задає один із перечислених класів символів:

upper, digit, alnum, graph, print

Якщо користувач задав невірну назву класу - вивести відповідне повідомлення про помилку у консоль і вивести йому невірно введену назву. Якщо назва класу задана вірно - вивести всі ASCII символи відповідного класу у консоль (у довільному форматі).

Підзавдання 2. Рядок символів

Всі команди даного підзавдання повинні починатись зі слова string. Перед циклом створити масив символів розміру 32 і заповнити його (хардкодно) довільним рядком символів (що завершується на \0). Команда string (без додаткового тексту) має виводити рядок у консоль.  
Після вводу будь-якої команди даного підзавдання вивести оригінальний рядок у консоль, застосувати команду і вивести змінений рядок у консоль.

Команди (частини команди розділяти символом ):

1. length - вивести довжину **рядка** символів.
2. clear - очистити рядок (зробити його пустим, довжина **рядка** має стати 0).
3. concat, {other\_string} - додати вміст іншого рядка ({other\_string}) у кінець рядка
4. lower - замінити всі літери рядка на відповідні літери у нижньому регістрі
5. substr, {from}, {length} - вивести підрядок із заданої позиції ({from}) і заданої довжини ({length}).
6. contains, {char} - перевірити чи оригінальний рядок містить у собі символ ({char})

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

Підзавдання 3. Масив рядків символів

Всі команди даного підзавдання повинні починатись зі слова strings. Перед циклом створити двовимірний масив символів розміру 10х32 і заповнити (хардкодно) кожен рядок цієї матриці довільним рядком символів (що закінчується на \0). Команда strings (без додаткового тексту) має виводити всі рядки у консоль (перед виводом рядка виводити його індекс у матриці, наприклад, [0] 'The first string').

Команди (частини команди розділяти символом ):

1. get, {index} - вивести рядок за індексом {index} та його довжину у консоль (або помилку, якщо індекс невірний).
2. set, {index}, {new\_string} - змінити вміст рядка за індексом ({index}) на нововведений ({new\_string}) (або вивести помилку, якщо індекс невірний).
3. count, non\_empty - вивести кількість не пустих рядків.
4. contain, {other\_string} - перевірити чи серед рядків є рядок, що аналогічний введеному рядку ({other\_string})

**Лістинг програми**

|  |
| --- |
| main.c |
| #include <stdio.h>  #include <stdlib.h>  #include <ctype.h>  #include <string.h>  #include <stdbool.h>  #include <time.h>  #include <unistd.h>  unsigned int sleep(unsigned int seconds);  int conReadLine(char str[], int maxBufLen)  {      fgets(str, maxBufLen, stdin);      int bufLength = strlen(str);      if (str[bufLength - 1] == '\n')      {          str[bufLength - 1] = '\0';          bufLength -= 1;      }      else      {          for (char ch; (ch = getchar()) != '\n';)          {          }      }      return bufLength;  }  char substring(char s[], char sub[], int p, int l) {     int c = 0;       while (c < l) {        sub[c] = s[p+c-1];        c++;     }     sub[c] = '\0';  }  void help(void) //help  {      printf("\nchar.{class} - prints symbols of class:\n  char.upper - prints uppercase letters\n  char.digit - prints digits\n  char.alnum - prints digits, upper and lowercase letters\n  char.graph - prints graphic characters\n  char.print - prints graphic characters and space\n");      printf("\nstring or string.{class} - creates random string\n  string - prints string\n  string.length - length of string\n  string.clear - clear string\n  string.concat.{other\_string} - adds other string\n  string.lower - transform UPPERCASE to lowercase\n  substr.{from}.{length} - prints string from first argument with length of second argument\n  string.contains.{char} - is string contain character\n");      printf("\nstrings - prints random array of strings:\n  strings.get.{index} - prints string with index\n  strings.set.{index}.{new\_str} - change string with index to new\_str\n  strings.count.non\_empty - counts not empty strings\n  strings.contain.{other\_str} - is other string in string\n");  }  int isNum(char \*argum)  {      for (int i = 0; i < strlen(argum); i++)      {          char k = argum[i];          if ((isdigit(k) == 0) && (k != '\n'))          {              return 0;          }      }      return 1;  }  void chart(char \*arg2)  {      int Ch = 6;      char argschr[5][6] = { "upper", "digit", "alnum", "graph", "print" };      if (arg2 == 0) Ch = 0;      else      for (int i = 0; i<5; i++)          if (strncmp(arg2, argschr[i], 6) == 0) Ch = i + 1;      printf("\n  => ");      for (int i = 0; i < 127; i++)      switch (Ch)      {      case 0:          puts("  🚫 ERR: Wrong argument. Try again");          return;      case 1:          if (isupper(i)) printf("%c", i); break;          break;      case 2:          if (isdigit(i)) printf("%c", i); break;      case 3:          if (isalnum(i)) printf("%c", i); break;      case 4:          if (isgraph(i)) printf("%c", i); break;      case 5:          if (isprint(i)) printf("%c", i); break;      default:          puts("  🚫 ERR: Wrong argument. Try again");          return;      }      printf(" <=\n");  }  int strt(char \*arg1, char \*arg2, char \*arg3, char \*arg4)  {      char \*args1 = {"string"};      size\_t len = 2048;      int w = 32; char Str[len];      srand(time(NULL));      printf("\n  => ");      for (int i = 0; i < w; i++)      {          do          {              Str[i] = rand() % 127;          } while (isgraph(Str[i]) == 0);          Str[31] = '\0';      }      printf("%s", Str);  printf(" <=\n");      while(1)      {          size\_t len = 2;          int k = 0;          int num2;          char sub[1000];          char \*\*trash;          int St = 7, l, l1, i, num1, g, z;          char argsstr[8][9] = { "length", "clear", "concat", "lower", "substr", "contains"};          if (arg2 == 0) St = 0;          else              for (int i = 0; i<6; i++)                  if (strncmp(arg2, argsstr[i], 9) == 0) St = i + 1;          switch (St)          {          case 0:              printf("\n  String is: %s\n", Str);              break;          case 1:              if (arg3 !=0) puts("\n  🚫 ERR: Argument 3 must be empty. Try again");              else printf("\n Length: %li\n", strlen(Str));              break;          case 2:              if (arg3 !=0) puts("\n  🚫 ERR: Argument 3 must be empty. Try again");              else Str[0] = '\0'; printf("\n  Length: %li\n", strlen(Str));              break;          case 3:              if (arg4 == 0)              {                  char \*i = arg3;                  if (i == 0) {printf("\n String has not changed: %s\n", Str);}                  else strcat(Str, i);    printf("\n  Now string is: %s\n", Str);              }              else puts("\n   🚫 ERR: Argument 4 must be empty. Try again");              break;          case 4:              l = strlen(Str);              if (l != 0) for (i = 0; i<l; i++) if (isupper(Str[i]) != 0) Str[i] = tolower(Str[i]);              printf("\n  Now string is: %s\n", Str);              break;          case 5:              l = strlen(Str);              if ((arg3 == 0) || (arg4 == 0))  puts ("\n  🚫 ERR: Wrong argument(s). Try again");              else if ((isNum(arg3) == 0) || (isNum(arg3) == 0)) puts ("\n    🚫 ERR: Argument 1 or 2 is not number. Try again");              else              {                  num1 = strtol(arg3, trash, 10);                  num2 = strtol(arg4, trash, 10);                  if (num1+num2>(l-1))                      puts ("\n   🚫 ERR: Too high values. Try again");                  else                  {                      substring(Str, sub, num1, num2);                      printf("\n  Substring is \"%s\"\n", sub);                  }              }              break;          case 6:              if (strlen(arg3) == 1)              {                  arg3[len] = '\0';                  char arg3\_1 = arg3[0];                  printf("\n  Char: \'%c\'\n", arg3\_1);                  for (int i = 0; i < 127 && Str[i] != 0; i++)                  {                      k += (Str[i] == arg3\_1) ? 1 : 0;                  }                  printf("\n  Character \'%c\' on \"%s\" repeated %i times\n", arg3\_1, Str, k);                  break;              }              else              {                  puts ("\n   🚫 ERR: Entered argument is not character. Try again");                  break;              }          case 7:              puts("\n    🚫 ERR: Wrong argument. Try again");              break;          }          size\_t strLen = 1000;          char str[strLen];          char \*saveptr;          do          {              printf("\n   Enter command, main or exit (Now you in STRING menu)\n");              printf("⋙  ");              conReadLine(str, strLen);              arg1 = strtok\_r(str, ".", &saveptr);              arg2 = strtok\_r(NULL, ".", &saveptr);              arg3 = strtok\_r(NULL, ".", &saveptr);              arg4 = strtok\_r(NULL, "\0", &saveptr);              if (strncmp(arg1, "main", 5) == 0) {return 0;}              else if (strncmp(arg1, "exit", 5) == 0) {exit(0);}              else if (strncmp(arg1, args1, 8) != 0) puts("\n 🚫 ERR: Wrong command. Try again");          } while (strncmp(arg1, args1, 8) != 0);      }  }  int strst(char arg1[], char arg2[], char arg3[], char arg4[])  {      char \*args2 = {"strings"};      int h = 10, w = 32, k=0;      srand(time(NULL));      char Arr[h][1000];      for (int i = 0; i < h; i++)      {          for (int j = 0; j < w; j++)          {              do              {                  Arr[i][j] = rand() % 127;              } while (isgraph(Arr[i][j]) == 0);              Arr[i][31] = '\0';          }      }      while(1)      {          size\_t len = 1000;          int Sts = 5, num1;          char argsstrs[4][8] = { "get", "set", "count", "contain"}, \*\*trash;          if (arg2 == 0) Sts = 0;          else              for (int i = 0; i<6; i++)                  if (strncmp(arg2, argsstrs[i], 8) == 0) Sts = i + 1;          switch (Sts)          {          case 0:              for (int i = 0; i < h; i++)              {                  printf("[%i] - %s\n", i, Arr[i]);              }              break;          case 1:              if (arg3 != 0)              {                  if (isNum(arg3) == 1){ num1 = strtol(arg3, trash, 10);                      if (num1 >= 10)                      {                          puts ("\n   🚫 ERR: Argument is too high. Try again");                      }                      else printf("\n String with index %i is \'%s\'\n", num1, Arr[num1]);                  }                  else                  {                      puts ("\n   🚫 ERR: Argument is not number. Try again");                  }              }              else              {                  puts ("\n   🚫 ERR: Argument is empty. Try again");              }              break;          case 2:              if (isNum(arg3) != 0)              {                  if (arg3 == 0)                  {                      puts("  🚫 ERR: Wrong index");                  }                  else                  {                      num1 = strtol(arg3, 0, 10);                      if (arg4 == 0) Arr[num1][0] = '\0';                      else                      {                          memmove(Arr[num1], arg4, strlen(arg4));                          Arr[num1][strlen(arg4)] = '\0';                          printf("\nNow string with index %i is \"%s\"\n", num1, Arr[num1]);                      }                  }              }              else              {                  puts("  🚫 ERR: Index is not number\n");              }              break;          case 3:              if (arg4 != 0) {puts("  🚫 ERR: Unknown index\n");}              else              {                  if (strncmp(arg3, "non\_empty", 10) == 0)                  {                      len = 10;                      int z = len;                      for (int i = 0; i < len; i++)                      {                          if (strlen(Arr[i]) == 0) z--;                      }                      printf("%i not empty strings in array", z);                  }                  else {puts("    🚫 ERR: Wrong argument\n");}              }              break;          case 4:              if ((arg3 == 0) || (arg4 != 0))              {                  puts("  🚫 ERR: Unknown index");              }              else              {                  for(int i = 0; i<10; i++)                  {                      if (strcmp(arg3, Arr[i]) == 0) { k++;}                      if (k==0) {puts("String not found"); break;}                      else printf("Found %i strings", k);                  }              }              break;          case 5:              puts("  🚫 ERR: Wrong command");              break;          }          size\_t strLen = 1000;          char str[strLen];          char \*saveptr;          do          {              printf("\n   Enter command, main or exit (Now you in STRINGS menu)\n");              printf("⋙  ");              conReadLine(str, strLen);              arg1 = strtok\_r(str, ".", &saveptr);              arg2 = strtok\_r(NULL, ".", &saveptr);              arg3 = strtok\_r(NULL, ".", &saveptr);              arg4 = strtok\_r(NULL, ".", &saveptr);              if (strncmp(arg1, "main", 5) == 0) {return 0;}              else if (strncmp(arg1, "exit", 5) == 0) {exit(0);}              else if (strncmp(arg1, args2, 8) != 0) puts("\n 🚫 ERR: Wrong command. Try again");          } while (strncmp(arg1, args2, 8) != 0);      }  }  int main()  {      system("clear");      srand(time(0));      size\_t strLen = 1000;      char str[strLen];      system("clear");      while (1)      {          int St1 = 4, k = 0;          char \*saveptr;          printf("\n   Enter command, help or quit\n");          printf("⋙  ");          conReadLine(str, strLen);          char \*arg1, \*arg2, \*arg3, \*arg4;          arg1 = strtok\_r(str, ".", &saveptr);          arg2 = strtok\_r(NULL, ".", &saveptr);          arg3 = strtok\_r(NULL, ".", &saveptr);          arg4 = strtok\_r(NULL, ".", &saveptr);          printf("  Your input: %s", arg1);          if (arg2 != 0) printf(".%s", arg2);          if (arg3 != 0) printf(".%s", arg3);          if (arg4 != 0) printf(".%s", arg4);          putchar('\n');          char args1[5][8] = { "exit", "char", "string", "strings", "help" };          for (int i = 0; i<4; i++)              if (strncmp(arg1, args1[i], 8) == 0) St1 = i;          switch (St1)          {              case 1:                  chart(arg2);                  break;              case 2:                  strt(arg1, arg2, arg3, arg4);                  break;              case 3:                  strst(arg1, arg2, arg3, arg4);                  break;              case 4:                  help();                  break;              case 0:                  exit(0);                  break;              default:                  puts("\n    🚫 ERR: Wrong command. Try again");                  break;          }      }  } |

**Приклади результатів**

|  |
| --- |
| char upper |
| ABCDEFGHIJKLMNOPQRSTUVWXYZ |
| char digit |
| 123456789 |
| char alnum |
| 0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz |
| char graph |
| !"#$%&'()\*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^\_`abcdefghijklmnopqrstuvwxyz{|}~ |
| char print |
| !"#$%&'()\*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^\_`abcdefghijklmnopqrstuvwxyz{|}~ |
| nbdsvbndbsv |
| Your input: nbdsvbndbsv  🚫 ERR: Wrong command. Try again |
| char.fagbnhgaf |
| Your input: char.fagbnhgaf  => 🚫 ERR: Wrong argument. Try again |

|  |
| --- |
| string length |
| => yU'GgKd\_W{<Qf@F5Jc`DWq'ecVR=:zB <=  Length: 31 |
| string.clear |
| Length: 0 |
| string concat dvsbhnjfthd |
| Now string is: dvsbhnjfthd |
| string |
| String is: dvsbhnjfthd |
| String.lower |
| Now string is: dvsbhnjfthd |
| String.substr.2.4 |
| Substring is "vsbh" |
| String.contain.v |
| Char: 'v'  Character 'v' on "dvsbhnjfthd" repeated 1 times |
| String.concat |
| String has not changed: dvsbhnjfthd  Now string is: dvsbhnjfthd |

|  |
| --- |
| strings |
| Your input: strings  [0] - kECTo-^]\\*}Vgd<g2|C{u>Iw23!x]k`  [1] - ahg8ZOC@N(7Lc3Bt2M(Kd)Hz4U#Y}[5  [2] - uIfx+q%eR;[/UwH:P(pNVz"Glk;gmQ!  [3] - i8Y15o=A{<TR\*AVD9b0lJDu1P@R`)<z  [4] - Gj,f8=|'G^kAfY\5">=9#TOUzwMqO+.  [5] - j]!^W?k&9"EX\*]C/%n@S$Vu)wT\*PhAZ  [6] - x(^)&9mP1w<10U^vWj"Scrsk\_w>tobf  [7] - ;$CLV#bMrMo!K}La%YYoBtOI'}(Aj~\  [8] - LG5R-H3F+Nk|!}ecr5%kT#|]k{22@z>  [9] - Eko62f'k:f-oT6~A1=/v!}9U0!gX)we |
| Strings.get.9 |
| String with index 9 is 'rj94hNP=ktem9A>]Le=Q|nO#C:=k; |
| Strings.set.11.dfghjk |
| 🚫 ERR: Wrong index |
| Strings.get.34 |
| 🚫 ERR: Argument is too high. Try again |

**Висновки**

Виконавши дану лабораторну роботу, було застосовані стандартні функції для роботи з рядками символів. Було створено і застосовано користувацькі функції. Було застосовано циклічні та умовні конструкції для перевірки правильного вводу рядку команди користувача. Булі створенні рядки символів та двовимірний масив.

В роботі був використано цикл while для того, щоб програма працювала до тих пір поки користувач не ввиде “exit”. В випадку, якщо користувач вводить не вірну команду, то у консоль виводить одну з помилок :"Command after char is entered incorrectly ", "Enter two digits after the space after substr", "invalid first number entered ", "invalid second number entered", "It is too long symbol", "Command after string is entered incorrectly " і т.д. Також в даній лабораторній роботі були використані вказівники для того, щоб визначити номер пробіла. Це було зроблено для того, щоб зчитувати елементи які ввів користувач після пробіла.

Компіляція всього коду відбувалася за допомогою компілятора gcc.