Міністерство освіти та науки України  
Національний технічний університет України  
«Київський політехнічний інститут»  
Факультет прикладної математики  
Кафедра системного програмування і спеціалізованих  
комп’ютерних систем

**Лабораторна робота №1**

**Частина 2**з дисципліни

**«Об’єктно орієнтоване програмування»**Тема: **«**Указатели, строки, массивы**»**

Виконав: Сахнік Іван Сергійович

Студент групи КВ-42

Перевірив(ла)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Київ-2015**

2.3. Завдання на роботу (частина II).

* 1. Написати функцію error\_t argz\_create\_sep (const char \*string, int sep, char \*\*argz, size\_t \*argz\_len). Функція перетворює null-terminated рядок string в argz вектор, розбиваючи рядок на елементи щораз, коли зустрічається символ sep. Вектор argz є null-terminated рядок, у якій елементи розділені символом ’\0’.
  2. Написати функцію size\_t argz\_count (const char \*argz, size\_t arg\_len), що повертає число елементів argz. Для попереднього приклада результатом буде 4.

3. Написати функцію error\_t argz\_add (char \*\*argz, size\_t \*argz\_len, const char \*str), що додає рядок str у кінець argz, модифікуючи належним чином argz й argz\_len.

4. Написати функцію void argz\_delete (char \*\*argz, size\_t \*argz\_len, char \*entry), що видаляє елемент entry з argz, модифікуючи належним чином argz й argz\_len.

5. Написати функцію error\_t argz\_insert (char \*\*argz, size\_t \*argz\_len, char \*before, const char \*entry), що вставляє рядок entry перед елементом before, модифікуючи належним чином argz й argz\_len.

6. Написати функцію char \* argz\_next (char \*argz, size\_t argz\_len, const char \*entry), що забезпечує можливість переміщення по argz. Вона повертає покажчик на елемент argz, що слідує за елементом entry або 0, якщо таких немає. Якщо entry дорівнює 0, повертається покажчик на перший елемент argz. Таким чином, забезпечується одна із двох можливостей доступу до елементів argz

7. Написати функцію error\_t argz\_replace(char \*\*argz, size\_t \*argz\_len, const char \*str, const char \*with), що заміщає рядок str елементом with, модифікуючи належним чином argz й argz\_len.

8. Написати функцію void argz\_print(const char \*argz, size\_t argz\_len) друку argz.

***Текст програми:***

#ifndef MYHEADER\_H

#define MYHEADER\_H

#include <stddef.h> // for size\_t

#include <string.h>

#include <stdio.h>

#include <malloc.h>

#include <stdlib.h>

typedef enum {OK, ENOMEM} error\_t;

error\_t argz\_create\_sep (const char \*string, int sep, char \*\*argz, size\_t \*argz\_len);

size\_t argz\_count (const char \*argz, size\_t arg\_len);

error\_t argz\_add (char \*\*argz, size\_t \*argz\_len, const char \*str);

void argz\_delete (char \*\*argz, size\_t \*argz\_len, char \*entry);

error\_t argz\_insert (char \*\*argz, size\_t \*argz\_len, char \*before, const char \*entry);

char \* argz\_next (char \*argz, size\_t argz\_len, const char \*entry);

error\_t argz\_replace(char \*\*argz, size\_t \*argz\_len, const char \*str, const char \*with);

void argz\_print(const char \*argz, size\_t argz\_len);

#endif //MYHEADER\_H

#include"myheader.h"

int substr(const char \*string1, const char \*string2, size\_t \*argz\_len)

{

int str2 = strlen(string2);

int str1 = \*argz\_len;

int i,j;

int result =- 1;

if (str2 > str1)

{

printf("string too large\n");

return -1;

}

for (i=0; i < str1 - str2 + 1; i++)

for(j = 0; j < str2; j++)

{

if (string2[j] != string1[i+j])

break;

else

if (j == str2 - 1)

result = i;

}

return result;

}

error\_t argz\_create\_sep (const char \*string, int sep, char \*\*argz, size\_t \*argz\_len)

{

int i;

\*argz\_len = strlen(string);

if (NULL == (\*argz=(char\*)malloc((\*argz\_len+1)\*sizeof(char))))

{

printf("Allocation fail");

exit(1);

}

for (i=0; i < \*argz\_len; i++)

if (string[i] == sep)

(\*argz)[i] = '\0';

else

(\*argz)[i] = string[i];

return OK;

}

void argz\_print(const char \*argz, size\_t argz\_len)

{

int i;

if (argz != NULL)

for (i = 0; i < argz\_len; i++)

printf("%c", argz[i]);

printf("\n\n");

}

size\_t argz\_count (const char \*argz, size\_t argz\_len)

{

int i, count = 0;

if (argz != NULL)

for (i = 0; i < argz\_len; i++)

if (argz[i] == '\0' && argz[i+1] != '\0')

++count;

if (argz[argz\_len - 1] != '\0')

++count;

return count;

}

error\_t argz\_add (char \*\*argz, size\_t \*argz\_len, const char \*str)

{

int i,j;

int str\_l = strlen(str);

char \*arg = NULL;

if (NULL == (arg=(char\*)malloc((\*argz\_len + str\_l + 1)\*sizeof(char))))

{

printf("Allocation fail");

exit(1);

}

for (i = 0; i < (\*argz\_len+1); i++)

arg[i] = (\*argz)[i];

j = \*argz\_len;

for(i=0; i<str\_l; i++,j++)

arg[j] = str[i];

arg[\*argz\_len+str\_l] = '\0';

free(\*argz);

(\*argz) = arg;

\*argz\_len = \*argz\_len+str\_l + 1;

return OK;

}

void argz\_delete (char \*\*argz, size\_t \*argz\_len, char \*entry)

{

int poz, j, l\_ent;

char \*arg = NULL, \*pointer = NULL;

pointer = \*argz;

poz = substr(\*argz, entry, argz\_len);

pointer = (\*argz)+poz;

l\_ent = strlen(pointer) + 1;

if (NULL == (arg = (char\*)malloc((\*argz\_len - l\_ent + 1)\*sizeof(char))))

{

printf("Allocation is failed\n");

exit(1);

}

for (j=0; j< poz; j++)

arg[j] = (\*argz)[j];

j = j + l\_ent;

while (j <= \*(argz\_len))

{

arg[poz] = (\*argz)[j];

++j; ++poz;

}

\*(argz\_len) = \*(argz\_len) - l\_ent;

free(\*argz);

(\*argz) = arg;

}

error\_t argz\_insert (char \*\*argz, size\_t \*argz\_len, char \*before, const char \*entry)

{

int poz, j, k, l\_ent;

char \*pointer, \*arg;

if (NULL == before)

{

argz\_add(argz, argz\_len, entry);

return OK;

}

l\_ent = strlen(entry) + 1;

if (NULL == (arg = (char\*)malloc(((\*argz\_len) + l\_ent + 1)\*sizeof(char))))

{

printf("Allocation fail\n");

exit(1);

}

pointer = \*argz;

poz = substr(\*argz, before, argz\_len);

pointer = (\*argz) + poz;

j = k = 0;

if (poz <= \*argz\_len)

{

for (j=0; j< poz; j++)

arg[j] = (\*argz)[j];

}

else

poz = 0;

while (k < l\_ent)

arg[j++] = entry[k++];

while (poz <= \*(argz\_len))

{

arg[j] = (\*argz)[poz];

++j; ++poz;

}

\*(argz\_len) = \*(argz\_len) + l\_ent;

free(\*argz);

(\*argz) = arg;

return OK;

}

error\_t argz\_replace(char \*\*argz, size\_t \*argz\_len, const char \*str, const char \*with)

{

int poz;

poz = substr(\*argz, str, argz\_len);

if (poz != -1)

{

if (argz\_insert (argz, argz\_len, str, with) != ENOMEM)

argz\_delete (argz, argz\_len, str);

return OK;

}

else

return ENOMEM;

}

char \* argz\_next (char \*argz, size\_t argz\_len, const char \*entry)

{

int poz, len;

char \*pointer = NULL;

if (entry == NULL)

return argz;

poz = substr(argz, entry, &argz\_len);

pointer = argz+poz;

len = strlen(pointer);

if (poz + len < argz\_len)

{

pointer = argz + poz + len + 1;

return pointer;

}

return NULL;

}

#include "myheader.h"

int main()

{

size\_t argz\_len;

int i,k;

char \*argz, \*entry;

char \*string="Sakhnik:Ivan:Sergiyovych:";

argz\_create\_sep (string, 58, &argz, &argz\_len);

argz\_print(argz, argz\_len);

printf("%d\n\n",argz\_count (argz,argz\_len));

argz\_add (&argz,&argz\_len, "KV-42");

argz\_print(argz, argz\_len);

argz\_delete (&argz, &argz\_len, "Ivan");

argz\_print(argz, argz\_len);

argz\_insert (&argz, &argz\_len, "Sergiyovych", "KPI");

argz\_print(argz, argz\_len);

argz\_replace(&argz, &argz\_len, "KPI", "with");

argz\_print(argz, argz\_len);

entry = NULL;

k = argz\_count(argz,argz\_len);

i = 0;

while (i < k)

{

printf("Argz[%d]=%s\n", i, entry = argz\_next(argz,argz\_len,entry));

i++;

}

getchar();

return 0;

}

***Тестування програми:***

