

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Факультет прикладної математики

Кафедра системного програмування і спеціальних комп’ютерних систем

**Лабораторна робота №1**

з дисципліни “ Об’ектно-орієнтоване програмування (частина I) ”

Тема: «Покажчики, рядки, масиви (Частина II)»

**Варіант № 5**

Виконав:

студент 1 курсу ФПМ

групи КВ-41:

Горпинич-Радуженко

Іван Олександрович

**Перевірено:**

Київ 2015

**Завдання**

* 1. Написать функцию error\_t argz\_create\_sep (const char \*string, int sep, char \*\*argz, size\_t \*argz\_len). Функция преобразует null-terminated строку string в argz вектор разбивая строку на элементы всякий раз, когда встречается символ sep. Вектор argz есть null-terminated строка, в которой элементы разделены символом ’\0’. Элементами векора являются пары

name=value

Например,

char \*const string = "SHELL=/bin/bash:usr=monty:PWD=/bin/monty:LANG=en\_US.UTF-8";

char \*argz;

int argz\_len;

argz\_create\_sep (string, 58 /\*ASCII for : \*/, &argz, &argz\_len);

В результате вызова вектор примет вид

SHELL=/bin/bash’\0’usr=monty’\0’PWD=/bin/monty’\0’LANG=en\_US.UTF-8’\0’

а argz\_len будет равно 58.

2. Написать функцию size\_t argz\_count (const char \*argz, size\_t arg\_len), которая возвращает число элементов argz. Для предыдущего примера результатом будет 4.

3. Написать функцию error\_t argz\_add (char \*\*argz, size\_t \*argz\_len, const char \*str), которая добавляет строку str в конец argz, модифицируя надлежащим образом argz и argz\_len.

4. Написать функцию void argz\_delete (char \*\*argz, size\_t \*argz\_len, char \*entry), которая удаляет элемент entry из argz, модифицируя надлежащим образом argz и argz\_len.

5. Написать функцию error\_t argz\_insert (char \*\*argz, size\_t \*argz\_len, char \*before, const char \*entry), которая вставляет строку entry перед элементом before, модифицируя надлежащим образом argz и argz\_len.

6. Написать функцию char \* argz\_next (char \*argz, size\_t argz\_len, const char \*entry), которая обеспечивает возможность перемещения по argz. Она возвращает указатель на элемент argz, следующий за эементом entry или 0, если таковых нет. Если entry равен 0, возвращается укахзатель на первый элемент argz. Таким образом, обеспечивается одна из двух возможностей доступа к элементам argz

char \*entry = 0;

while ((entry = argz\_next (argz, argz\_len, entry)))

action;

char \*entry;

for (entry = argz; entry; entry = argz\_next (argz, argz\_len, entry))

action;

7. Написать функцию error\_t argz\_replace(char \*\*argz, size\_t \*argz\_len, const char \*str, const char \*with), которая замещает строку str элементом with, модифицируя надлежащим образом argz и argz\_len.

8. Написать функцию void argz\_print(const char \*argz, size\_t argz\_len) печати argz.

Тип error\_t определен следующим образом

typedef enum {OK, ENOMEM} error\_t;

Т.о. в случае успеха возвращается первая константа (или 0) и вторая (1) в пртивном случае.

**Код программи**

argz.h

#ifndef \_ARGZ\_H

#define \_ARGZ\_H

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <malloc.h>

typedef enum { OK, ENOMEMO } error\_t;

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

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

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

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

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

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

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

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

#endif

argz.c

#include "argz.h"

char\* argz\_occur(const char \*argz, size\_t argz\_len, const char \*string);

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

unsigned i;

if (strlen(string) == 0) return ENOMEM;

\*argz\_len = strlen(string) + 1;

\*argz = (char\*)malloc(sizeof(char\*)\*(\*argz\_len));

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

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

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

}

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

return OK;

}

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

unsigned i;

size\_t count = 0;

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

if (argz[i] == '\0') ++count;

return count;

}

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

unsigned i = 0;

\*argz = realloc(\*argz, sizeof(\*argz) + sizeof(char\*)\*(strlen(str) + 2));

do {

(\*argz)[(\*argz\_len) + i] = str[i];

++i;

} while (str[i]);

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

(\*argz\_len) = (\*argz\_len) + i + 1;

return OK;

}

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

unsigned occur\_index, entry\_len; unsigned i;

char \*occur\_ptr = argz\_occur(\*argz, \*argz\_len, entry);

if (occur\_ptr == NULL) return;

occur\_index = occur\_ptr - (\*argz);

entry\_len = strlen(entry) + 1;

for (i = occur\_index; i < (\*argz\_len) - entry\_len; ++i)

(\*argz)[i] = (\*argz)[i + entry\_len];

(\*argz\_len) = (\*argz\_len) - entry\_len;

}

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

unsigned occur\_index, entry\_len; unsigned i;

char \*occur\_ptr = argz\_occur(\*argz, \*argz\_len, before);

if (occur\_ptr == NULL) ENOMEM;

occur\_index = occur\_ptr - (\*argz);

entry\_len = strlen(entry) + 1;

\*argz = realloc(\*argz, sizeof(\*argz) + sizeof(char\*)\*(strlen(entry) + 2));

for (i = (\*argz\_len) - 1; i >= occur\_index; --i)

(\*argz)[i + entry\_len] = (\*argz)[i];

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

(\*argz)[occur\_index + i] = entry[i];

(\*argz\_len) = (\*argz\_len) + entry\_len;

return OK;

}

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

unsigned occur\_index; unsigned i;

char \*occur\_ptr;

if (entry == 0) return argz;

occur\_ptr = argz\_occur(argz, argz\_len, entry);

if (occur\_ptr == NULL) return NULL;

occur\_index = occur\_ptr - argz;

i = occur\_index;

while (argz[i]) ++i;

if (i == argz\_len - 1) return NULL;

++i;

return(&argz[i]);

}

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

unsigned occur\_index, str\_len, with\_len; unsigned i;

char \*occur\_ptr = argz\_occur(\*argz, \*argz\_len, str);

if (occur\_ptr == NULL) return ENOMEM;

occur\_index = occur\_ptr - (\*argz);

i = occur\_index;

while ((\*argz)[i]) ++i;

str\_len = i - occur\_index;

with\_len = strlen(with);

if (with\_len > str\_len) {

\*argz = realloc(\*argz, sizeof(\*argz) + sizeof(char\*)\*(with\_len - str\_len));

for (i = (\*argz\_len) - 1; i >= occur\_index; --i)

(\*argz)[i + with\_len - str\_len] = (\*argz)[i];

}

if (str\_len > with\_len)

for (i = occur\_index + str\_len; i < (\*argz\_len); ++i)

(\*argz)[i - str\_len + with\_len] = (\*argz)[i];

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

(\*argz)[occur\_index + i] = with[i];

(\*argz\_len) = (\*argz\_len) - str\_len + with\_len;

return OK;

}

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

unsigned i;

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

if (argz[i] != '\0') printf("%c", argz[i]);

else printf("'\\0'");

printf("\n");

}

char\* argz\_occur(const char \*argz, size\_t argz\_len, const char \*string) {

unsigned i = 0;

char \*occur\_ptr=NULL;

while (i < argz\_len) {

occur\_ptr = strstr(argz + i, string);

if (occur\_ptr == NULL) {

while (argz[i]) ++i;

++i;

}

else break;

}

return occur\_ptr;

}

Test.c

#include "argz.h"

int main() {

char \*string = "SHELL=/bin/bash:usr=monty:PWD=/bin/monty:LANG=en\_US.UTF-8";

char \*argz; size\_t argz\_len;

char \*entry; /\* for argz\_next \*/

/\* argz\_create\_sep & argz\_print \*/

argz\_create\_sep(string, ':', &argz, &argz\_len);

argz\_print(argz, argz\_len);

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

/\*argz\_count \*/

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

/\*argz\_add \*/

argz\_add(&argz, &argz\_len, "added=element");

argz\_print(argz, argz\_len);

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

/\*argz\_delete \*/

argz\_delete(&argz, &argz\_len, "usr=monty");

argz\_print(argz, argz\_len);

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

/\*argz\_insert \*/

argz\_insert(&argz, &argz\_len, "LANG=en\_US.UTF-8", "one.more=added\_element");

argz\_print(argz, argz\_len);

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

/\*argz\_next \*/

printf("First letters of ARGZ elements: ");

for (entry = argz; entry; entry = argz\_next(argz, argz\_len, entry)) printf("%c ", \*entry);

printf("\n\n");

/\*argz\_replace \*/

argz\_replace(&argz, &argz\_len, "one.more=added\_element", "replaced=element");

argz\_print(argz, argz\_len);

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

}

**Виконання програми**

