НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Лабораторна робота №1.2

З дисципліни: «Об’єктно-орієнтоване програмування»

Виконав:

Студент групи КВ-51

Тимошенко Ігор

Перевірив

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КИЇВ 2016

argz.h

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*file: argz.h

\*purpose: declarations for argz functions, types, constants

\*author: Igor Tymoshenko

\*written: 22/10/2016

\*last modified: 22/10/2016

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#ifndef \_ARGZ\_H

#define \_ARGZ\_H

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

typedef enum {OK, ENOMEM} error\_t;

/\* function prototypes \*/

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(char \*argz, size\_t argz\_len);

#endif

argz.c

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*file: argz.c

\*synopsis: These functions are declared in the include file "argz.h".

\*author: Igor Tymoshenko

\*written: 22/10/2016

\*last modified: 23/10/2016

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "argz.h"

/\*

The argz\_create\_sep function converts the null-terminated string string into an

argz vector (returned in argz and argz len) by splitting it into elements at every

occurrence of the character sep.

\*/

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

int i;

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

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

if ((sep > 255 || sep < 0) && ( \*argz == NULL))

return ENOMEM;

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

if ( \*(string + i) == sep)

\*(\*argz + i) = '\0';

else

\*(\*argz + i) = \*(string + i);

return OK;

}

/\*

Returns the number of elements in the argz vector.

\*/

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

int i;

size\_t count = 0;

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

if ( \*(argz + i) == '\0')

count++;

return count;

}

/\*

The argz\_add function adds the string str to the end of the argz vector // \*argz, and

updates \*argz and \*argz\_len accordingly.

\*/

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

int i;

int length = strlen(str) + 1;

\*argz = (char\*)realloc(\*argz, (\*argz\_len + length)\*sizeof(char));

if ( \*argz == NULL)

return ENOMEM;

for (i = 0; i <= length; i++)

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

\*argz\_len += length;

return OK;

}

/\*

If entry points to the beginning of one of the elements in the argz vector \*argz,

the argz\_delete function will remove this entry and reallocate \*argz, modifying \*argz

and \*argz\_len accordingly.

\*/

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

size\_t del\_len;

if ((entry == \*argz) || (\*(entry - 1) == '\0')){

del\_len = strlen(entry) + 1;

memmove(entry, entry + del\_len, \*argz\_len - (entry - \*argz + del\_len));

\*argz\_len -= del\_len;

\*argz = (char\*)realloc(\*argz, \*argz\_len);

}

}

/\*

The argz\_insert function inserts the string entry into the argz vector \*argz at a point

just before the existing element pointed to by before, reallocating \*argz and updating \*argz

and \*argz\_len.

\*/

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

int length = strlen(entry) + 1;

int i;

if ( before == NULL ){

if ( argz\_add(argz, argz\_len, entry) == ENOMEM )

return ENOMEM;

}

else{

\*argz = (char\*)realloc(\*argz, (\*argz\_len + length)\*sizeof(char));

if ( \*argz == NULL )

return ENOMEM;

memmove(before + length, before, (\*argz\_len - (before - \*argz))\*sizeof(char));

for (i = 0; i < length; i++)

\*(before + i) = \*(entry + i);

\*argz\_len += length;

}

return OK;

}

/\*

The argz\_next function provides a convenient way of iterating over the elements in the argz

vector argz. It returns a pointer to the next element in argz after the element entry, or 0 if

there are no elements following entry. If entry is 0, the first element of argz is returned.

\*/

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

int i = 0;

if (entry == NULL)

return argz;

while ( i < argz\_len){

if (entry == argz + i){

i += strlen(argz + i) + 1;

if ( i < argz\_len)

return argz + i;

}

i += strlen(argz + i) + 1;

}

return NULL;

}

/\*

Replace the string str in argz with string with, reallocating argz as

necessary.

\*/

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

int len1 = strlen(str);

int len2 = strlen(with);

int d\_len = len2 - len1; //delta length

char \*entry = NULL;

char \*begin = NULL;

int i;

if (len1 == 0 )

return ENOMEM;

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

begin = strstr(entry, str);

if ( begin != NULL ){

if ( d\_len < 0){

memmove(begin + len2, begin + len1, (\*argz\_len - (begin - \*argz + len1))\*sizeof(char));

\*argz = (char\*)realloc(\*argz, \*argz\_len + d\_len);

} else{

\*argz = (char\*)realloc(\*argz, \*argz\_len + d\_len);

memmove(begin + len2, begin + len1, (\*argz\_len - (begin - \*argz + len1))\*sizeof(char));

}

\*argz\_len += d\_len;

for (i = 0; i < len2; i++)

\*(begin + i) = \*(with + i);

return OK;

}

}

return ENOMEM;

}

/\*

prints argz vector

\*/

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

char \*entry = NULL;

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

printf("%s\n", entry);

}

argz\_test.c

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*file: argz\_test.c

\*purpose: testing operability of functions from "argz.c"

\*author: Igor Tymoshenko

\*written: 22/10/2016

\*last modified: 23/10/2016

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "argz.h"

int main(){

char \*string = "I=dev/sda1;am=dev/sda2;Bat=dev/sda3";

char \*argz;

size\_t argz\_len;

printf("%s\n1)\n", string);

// 1

if ( argz\_create\_sep(string, ';', &argz, &argz\_len) == OK){

argz\_print(argz, argz\_len);

// 2

printf("\n2) %d\n3)\n", (int)argz\_count(argz, argz\_len));

// 3

if ( argz\_add(&argz, &argz\_len, "man=dev/sda4") == OK )

argz\_print(argz, argz\_len);

else printf(" Error\n");

// 4

printf("\n4)\n");

argz\_delete(&argz, &argz\_len, argz + 23);

argz\_print(argz, argz\_len);

//5

printf("\n5)\n");

if ( argz\_insert(&argz, &argz\_len, argz + 23, "Spider=dev/sda3") == OK)

argz\_print(argz, argz\_len);

else printf(" Error\n");

//6

printf("\n6)\n");

if ( argz\_replace(&argz, &argz\_len, "an", "") == OK){

argz\_print(argz, argz\_len);

} else printf(" Error\n");

}

else printf(" Error\n");

free(argz);

return 0;

}