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

Базильський Лев

Перевірив

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

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

КИЇВ 2016

lab1\_part2.h

#if defined \_LAB1\_PART2

#endinput

#endif

#define \_LAB1\_PART2

#include <stdio.h>

#include <string.h> // strlen(), strstr(), ..

#include <stdlib.h> // calloc(), malloc(), realloc(), free()

#undef ENOMEM

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);

lab1\_part2.c

#include "lab1\_part2.h"

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

{

if (sep < 0 || sep > 255)

return ENOMEM;

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

if (\*argz\_len == 1)

return ENOMEM;

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

return ENOMEM;

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

{

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

{

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

}

else

{

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

}

}

return OK;

}

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

{

size\_t num = 0;

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

{

if (\*(argz + i) == '\0'/\* && \*(argz + i + 1) != '\0'\*/)

{

num++;

}

}

return num;

}

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

{

int len = strlen(str);

if (len == 0)

return ENOMEM;

if ((\*argz = (char\*)realloc(\*argz, (\*argz\_len + len + 1) \* sizeof(char))) == NULL)

return ENOMEM;

for (int i = 0; i <= len; i++)

{

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

}

\*argz\_len += len + 1;

return OK;

}

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

{

const int len = strlen(entry);

memmove(entry, entry + (len + 1) \* sizeof(char), (\*argz\_len - (entry - \*argz + len + 1)) \* sizeof(char));

\*argz\_len -= len + 1;

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

}

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

{

const int len = strlen(entry);

if (len == 0)

return ENOMEM;

if (before == NULL)

{

if (realloc(\*argz, (\*argz\_len + len + 1) \* sizeof(char)) == NULL)

return ENOMEM;

for (int i = 0; i <= len; i++)

{

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

}

}

else

{

if (realloc(\*argz, (\*argz\_len + len + 1) \* sizeof(char)) == NULL)

return ENOMEM;

memmove(before + (len + 1) \* sizeof(char), before, (\*argz\_len - (before - \*argz)) \* sizeof(char));

for (int i = 0; i <= len; i++)

{

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

}

}

\*argz\_len += len + 1;

return OK;

}

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

{

if (entry == NULL)

return argz;

for (char \*ptr = entry; ptr <= argz + argz\_len; ptr++)

{

if (ptr == argz + argz\_len - 1)

{

return NULL;

}

else if (\*ptr == '\0')

{

return ptr + 1;

}

}

return NULL;

}

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

{

const int len1 = strlen(str);

const int len2 = strlen(with);

if (len1 == 0 || len2 == 0)

return ENOMEM;

char \*ptr = \*argz, \*begin = NULL;

while (ptr < \*argz + \*argz\_len)

{

begin = strstr(ptr, str);

if (begin == NULL)

{

ptr += strlen(ptr) + 1;

continue;

}

else break;

}

if (begin == NULL)

return ENOMEM;

if (len1 < len2)

{

if (realloc(\*argz, (\*argz\_len - len1 + len2) \* sizeof(char)) == NULL)

return ENOMEM;

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

}

else if (len1 > len2)

{

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

if (realloc(\*argz, (\*argz\_len - len1 + len2) \* sizeof(char)) == NULL)

return ENOMEM;

}

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

{

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

}

\*argz\_len += len2 - len1;

return OK;

}

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

{

char \*entry = NULL;

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

{

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

}

}

lab1\_part2\_test.c

#include "lab1\_part2.h"

int main()

{

char \*argz;

size\_t argz\_len;

// 1

printf("1:\n");

if (argz\_create\_sep("SHELL=/bin/bash:usr=monty:=/bin/monty:LANG=en\_US.UTF-8", 58 /\*ASCII for : \*/, &argz, &argz\_len) == OK)

{

argz\_print(argz, argz\_len);

// 2

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

// 3

printf("\n3:\n");

argz\_add(&argz, &argz\_len, "asdf");

argz\_print(argz, argz\_len);

// 4

printf("\n4:\n");

argz\_delete(&argz, &argz\_len, argz + 16); // в качестве третьего параметра, лучше использовать argz\_next()

argz\_print(argz, argz\_len);

// 5

printf("\n5:\n");

argz\_insert(&argz, &argz\_len, argz, "new string");

argz\_print(argz, argz\_len);

// 6

// argz\_next()

// используется в argz\_print()

// 7

printf("\n7:\n");

argz\_replace(&argz, &argz\_len, "new string", "string2");

argz\_print(argz, argz\_len);

}

else printf("ERROR: Something went wrong! Check it out and try again.");

return 0;

}