**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**

з дисципліни

**«Об’єктно-орієнтоване програмування»**

Тема: **«Вказівники, рядки, масиви»**

Виконав: Місік Дмитро Сергійович

Студент групи КВ-31

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

Київ 2014

1. **Файл teststrings.c**

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

name: teststrings.c

description: main file, which call all functions and contains functions

"Menu", where you choose number of task

author: Dima

date of creation: 02.09.2014

written: 04.09.2014

date of last modified: 10.09.2014

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

#include <windows.h>

#include "mystring.h"

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

char **\***EnterString**()** **{** //enter dynamic string

int SLen **=** 0**;** //SLen - length of entered string

char ch**;**

char **\***str **=** **(**char **\*)**malloc**(sizeof(**char**)\*** 256**);** //string

**while** **((**ch **=** getchar**())** **!=** '\n'**)** **if** **(**SLen **!=** 255**)** str**[**SLen**++]** **=** ch**;** //enter string by chars

str**[**SLen**]** **=** '\0'**;** //add in end null symbol

str **=** **(**char **\*)**realloc**(**str**,** **(**1 **+** SLen**)** **\*** **sizeof(**char**));** //reallocation

**return** str**;**

**}**

void gotoxy**(**const int x**,** const int y**)** **{** //set cursor in console on {X,Y}

COORD c **=** **{** x**,** y **};**

SetConsoleCursorPosition**(**GetStdHandle**(**STD\_OUTPUT\_HANDLE**),** c**);**

**}**

int Menu**()** **{** //Choosing menu

int KEY**;**

**do** **{**

gotoxy**(**35**,** 7**);**

printf**(**"Part 1"**);**

gotoxy**(**35**,** 9**);**

printf**(**"1. task 1.1"**);**

gotoxy**(**35**,** 10**);**

printf**(**"2. task 1.2"**);**

gotoxy**(**35**,** 11**);**

printf**(**"3. task 1.3"**);**

gotoxy**(**35**,** 12**);**

printf**(**"4. task 1.4"**);**

gotoxy**(**35**,** 13**);**

printf**(**"5. task 1.5"**);**

KEY **=** \_getch**();**

**if** **(**KEY **<=** '5' **&&** KEY **>=** '1'**)** **break;**

**}** **while** **(**true**);**

system**(**"cls"**);**

**return** KEY **-** 49**;**

**}**

int main**()** **{** //Main program

char **\***string1**,** **\***string2**,** **\***pal**;**

int **\***size**,** i**;**

double **\***arr**;**

**switch** **(**Menu**())** **{**

**case** 0**:**

printf**(**"Print first string\n"**);**

string1 **=** EnterString**();**

system**(**"cls"**);**

printf**(**"Print second string\n"**);**

string2 **=** EnterString**();**

system**(**"cls"**);**

printf**(**"String 2 starts in String 1 from position %d"**,** substr**(**string1**,** string2**));**

free**(**string1**);**

free**(**string2**);**

**break;**

**case** 1**:**

printf**(**"Print first string\n"**);**

string1 **=** EnterString**();**

system**(**"cls"**);**

printf**(**"Print second string\n"**);**

string2 **=** EnterString**();**

system**(**"cls"**);**

printf**(**"Max subsequence - %d"**,** subseq**(**string1**,** string2**));**

free**(**string1**);**

free**(**string2**);**

**break;**

**case** 2**:**

printf**(**"Print string\n"**);**

string1 **=** EnterString**();**

system**(**"cls"**);**

**if** **(**ispal**(**string1**)** **==** 1**)** printf**(**"It is palindrome"**);**

**else** printf**(**"It isn't palindrome"**);**

free**(**string1**);**

**break;**

**case** 3**:**

printf**(**"Print string\n"**);**

string1 **=** EnterString**();**

system**(**"cls"**);**

printf**(**pal **=** makepal**(**string1**));**

free**(**string1**);**

free**(**pal**);**

**break;**

**case** 4**:**

printf**(**"Print string\n"**);**

string1 **=** EnterString**();**

size **=** malloc**(sizeof(**int**));**

system**(**"cls"**);**

arr **=** txt2double**(**string1**,** size**);**

**for** **(**i **=** 0**;** i **<** **\***size**;** i**++)** printf**(**"%d. - %f\n"**,** i**,** arr**[**i**]);**

**if** **(\***size **==** 0**)** printf**(**"Error"**);**

free**(**string1**);**

free**(**size**);**

free**(**arr**);**

**break;**

**}**

\_getch**();**

**return** 0**;**

**}**

1. **Файл mystring.c**

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

name: mystring.c

description: function "substr", that search in string1

subsequence, that equals to string2 and return -1 if

subsequence didn't founded and position of first entry

of string2 in string1

function "subseq" search maximal subsequence that contains

string1 and string2. function returns 0, if string1 and

string2 hasn't mutual subsequence and

return length of subsequence, if they have

function "ispal" that returns 0 if

string isn't palindrome and returns 1 if string

is palindrome

function "makepal" makes with minimal adding characters

in end from string palindrome. returns this palindrome

function "IsCorrect" returns true if string can be

translated into array of double and false in another case.

function "FindSemicolon" search first entering of semicolon

and returns position of it. if semicolon doesn't exist,

function returns -1.

function "FindDot" search entering of dot and returns position

of it. if dot doesn't exist, function returns length of string.

function "Converter" converts string into float

function "Converting" calls function "Converter" twice

and returns full float number.

function "txt2double" translate string into array of float

numbers if it is possible. If it is impossible, function

returns pointer into NULL, if it is possible - pointer into

array of float numbers.

Format of numeric string - [integer].[mantissa]

author: Dima

date of creation: 02.09.2014

written: 04.09.2014

date of last change: 10.09.2014

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

#include "mystring.h"

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int substr**(**const char **\***string1**,** const char **\***string2**)** **{**

int start **=** 0**;** //position of subsequence

bool flag **=** false**;** //flag == true - subsequence founded, false - didn't founded

**while** **(**flag **==** false **&&** string1**[**start **+** strlen**(**string2**)** **-** 1**]** **!=** '\0'**)** **{**

int pos **=** 0**;**

flag **=** true**;**

**while** **(**string1**[**start **+** pos**]** **==** string2**[**pos**]** **&&** string2**[**pos**]** **!=** '\0'**)** pos**++;** //check symbols of strings

**if** **(**string2**[**pos**]** **!=** '\0'**)** **{** //subsequence didin't founded from position #start

flag **=** false**;**

start**++;**

**}**

**}**

**if** **(**flag **==** true**)** **return** start**;** //return position of first subsequence

**else** **return** **-**1**;** //subsequence didn't founded

**}**

int subseq**(**const char **\***string1**,** const char **\***string2**)** **{**

unsigned CUR\_LEN **=** 1**,** MAX\_LEN **=** 0**,** CUR\_POS **=** 0**,** i**;** //CUR\_LEN - current length, MAX\_LEN - maximal length, CUR\_POS - current position

char **\***STR **=** malloc**(sizeof(**char**)\*** 2**);** //STR - subsequence

STR**[**1**]** **=** '\0'**;**

**while** **(**CUR\_LEN **<** strlen**(**string2**)** **&&** CUR\_LEN **<** strlen**(**string1**))** **{**

**if** **(**string2**[**CUR\_POS **+** CUR\_LEN **-** 1**]** **==** '\0'**)** **{**

CUR\_POS **=** 0**;**

STR **=** **(**char **\*)**realloc**(**STR**,** **sizeof(**char**)\*** **((++**CUR\_LEN**)** **+** 1**));**

STR**[**CUR\_LEN**]** **=** '\0'**;**

**}**

**for** **(**i **=** 0**;** i **<** CUR\_LEN**;** i**++)** STR**[**i**]** **=** string2**[**CUR\_POS **+** i**];**

**if** **(**substr**(**string1**,** STR**)** **!=** **-**1**)** MAX\_LEN **=** CUR\_LEN**;**

CUR\_POS**++;**

**}**

free**(**STR**);**

**return** MAX\_LEN**;**

**}**

char ispal**(**const char **\***string**)** **{**

char PAL **=** 1**;**

int LEN **=** strlen**(**string**),** POS **=** 0**;**

**while** **(**POS **<** LEN **/** 2**)** **{**

**if** **(**string**[**POS**]** **!=** string**[**LEN **-** POS **-** 1**])** **{**

PAL **=** 0**;**

**break;**

**}**

**else** POS**++;**

**}**

**return** PAL**;**

**}**

char **\***makepal**(**const char **\***string**)** **{**

unsigned ADD **=** 0**,** i**,** STR\_LEN **=** strlen**(**string**);**

char **\***STR **=** malloc**(**strlen**(**string**)** **+** 1**);**

**for** **(**i **=** 0**;** i **<=** strlen**(**string**);** i**++)** STR**[**i**]** **=** string**[**i**];**

**while** **(**ispal**(**STR**)** **==** 0**)** **{**

STR **=** **(**char **\*)**realloc**(**STR**,** **((++**ADD**)** **+** STR\_LEN **+** 1**)** **\*** **sizeof(**char**));**

**for** **(**i **=** 0**;** i **<** ADD**;** i**++)** STR**[**ADD **+** STR\_LEN **-** i **-** 1**]** **=** STR**[**i**];**

STR**[**ADD **+** STR\_LEN**]** **=** '\0'**;**

**}**

**return** STR**;**

**}**

bool IsCorrect**(**const char **\***string**)** **{** //check string for errors in numbers

bool flag **=** true**;**

unsigned i**;**

**for** **(**i **=** 0**;** i **<** strlen**(**string**)** **-** 1**;** i**++)**

**if** **((**string**[**i**]** **<** '0' **||** string**[**i**]** **>** '9'**)** **&&** string**[**i**]** **!=** ';' **&&** string**[**i**]** **!=** '.'**)** **{**

flag **=** false**;**

**break;**

**}**

**else** **if** **((**string**[**i**]** **==** ';' **||** string**[**i**]** **==** '.'**)** **&&** i **!=** 0 **&&** i **!=** string**[**strlen**(**string**)** **-** 1**])** **{**

**if** **(**string**[**i **-** 1**]** **<** '0' **||** string**[**i **-** 1**]** **>** '9'**)** flag **=** false**;**

**else** **if** **(**string**[**i **+** 1**]** **<** '0' **||** string**[**i **+** 1**]** **>** '9'**)** flag **=** false**;**

**}**

**else** **if** **((**string**[**i**]** **==** ';' **||** string**[**i**]** **==** '.'**)** **&&** **(**i **==** 0 **||** i **==** string**[**strlen**(**string**)** **-** 1**]))** flag **=** false**;**

**return** flag**;**

**}**

int FindSemicolon**(**char **\***string**)** **{** //function returns position of semicolon

unsigned i**;** //and change semicolon into comma

**for** **(**i **=** 0**;** i **<** strlen**(**string**);** i**++)** **if** **(**string**[**i**]** **==** ';'**)** **{**

string**[**i**]** **=** ','**;**

**return** i**;**

**}**

**return** **-**1**;**

**}**

int FindDot**(**char**\*** string**)** **{** //find position of point in number

unsigned i**;**

**for** **(**i **=** 0**;** i **<** strlen**(**string**);** i**++)** **if** **(**string**[**i**]** **==** '.'**)** **return** i**;**

**return** strlen**(**string**);**

**}**

double Converter**(**char**\*** string**,** int pos**,** double m**,** double **\***res**,** bool flag**)** **{** //convert string mantissa into float

**switch** **(**string**[**pos**])** **{**

**case** '1'**:** **\***res **+=** m**;** **break;**

**case** '2'**:** **\***res **+=** 2 **\*** m**;** **break;**

**case** '3'**:** **\***res **+=** 3 **\*** m**;** **break;**

**case** '4'**:** **\***res **+=** 4 **\*** m**;** **break;**

**case** '5'**:** **\***res **+=** 5 **\*** m**;** **break;**

**case** '6'**:** **\***res **+=** 6 **\*** m**;** **break;**

**case** '7'**:** **\***res **+=** 7 **\*** m**;** **break;**

**case** '8'**:** **\***res **+=** 8 **\*** m**;** **break;**

**case** '9'**:** **\***res **+=** 9 **\*** m**;** **break;**

**}**

**if** **(**flag **==** false **&&** string**[**pos**]** **!=** '\0'**)** Converter**(**string**,** **++**pos**,** m **\*** 0.1**,** res**,** false**);**

**else** **if** **(**flag **==** true **&&** pos **-** 1 **!=** **-**1**)** Converter**(**string**,** **--**pos**,** m **\*** 10**,** res**,** true**);**

**return** **\***res**;**

**}**

double Converting**(**char**\*** string**)** **{**

double **\***res **=** malloc**(sizeof(**double**));**

int pos **=** FindDot**(**string**);**

**\***res **=** 0**;**

Converter**(**string**,** pos **-** 1**,** 1**,** res**,** true**);**

Converter**(**string**,** pos **+** 1**,** 0.1**,** res**,** false**);**

**return** **\***res**;**

**}**

double **\***txt2double**(**const char **\***string**,** int **\***size**)** **{**

char **\***str **=** malloc**(sizeof(**char**)\***strlen**(**string**));**

unsigned i**;**

**for** **(**i **=** 0**;** i **<=** strlen**(**string**);** i**++)** str**[**i**]** **=** string**[**i**];**

int LEN **=** strlen**(**str**);**

str **=** **(**char **\*)**realloc**(**str**,** LEN **+** 2**);**

str**[**LEN**]** **=** ';'**;**

str**[**LEN **+** 1**]** **=** '\0'**;**

double **\***arr **=** malloc**(sizeof(**double**));**

int SP **=** 0**;**

**\***size **=** 0**;**

**while** **(**IsCorrect**(**string**)** **&&** str**[**SP**]** **!=** '\0'**)** **{**

int SMCLN **=** FindSemicolon**(**str**);**

**if** **(**SMCLN **!=** **-**1**)** **{**

char **\***num **=** malloc**(sizeof(**char**)\*** **(**SMCLN **-** SP **+** 1**));**

**for** **(**i **=** 0**;** i **<** SMCLN **-** SP**;** i**++)** num**[**i**]** **=** str**[**SP **+** i**];**

num**[**SMCLN **-** SP**]** **=** '\0'**;**

arr**[\***size**]** **=** **(**double**)**Converting**(**num**);**

arr **=** **(**double **\*)**realloc**(**arr**,** **sizeof(**double**)\*** **((++(\***size**))** **+** 1**));**

SP **=** SMCLN **+** 1**;**

free**(**num**);**

**}**

**}**

**if** **(\***size **==** 0**)** **{**

free**(**arr**);**

arr **=** **NULL;**

**}**

**return** arr**;**

**}**

1. **Файл mystring.h**

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

name: lab1\_mpal.h

description: this file contains prototype of functions, that

described in "mystring.c"

author: Dima

date of creation: 02.09.2014

written: 04.09.2014

date of last change: 04.09.2014

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

**typedef** enum **{** **false,** **true** **}** bool**;**

double **\***txt2double**(**const char **\***string**,** int **\***size**);**

int substr**(**const char **\***string1**,** const char **\***string2**);**

int subseq**(**const char **\***string1**,** const char **\***string2**);**

char ispal**(**const char **\***string1**);**

char **\***makepal**(**const char **\***string**);**

1. **Файл argztest.c**

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

name: argztest.c

description: this file contains function "EnterString" that return pointer

to string you entered and "main", that calls functions from

file "argz.h" (functions in "argz.h" described in this file)

author: Dima

date of creation: 04.09.2014

written: 06.09.2014

date of last modified: 10.09.2014

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

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#include "argz.h"

char **\***EnterString**(**char**\*** string**)** **{** //enter dynamic string

int SLen **=** 0**;** //SLen - length of entered string

char ch**;**

char **\***str **=** **(**char **\*)**malloc**(sizeof(**char**)\*** 256**);** //string

printf**(**"Enter %s string\n"**,** string**);**

**while** **((**ch **=** getchar**())** **!=** '\n' **&&** SLen **!=** 255**)** str**[**SLen**++]** **=** ch**;** //enter string by chars

str**[**SLen**]** **=** '\0'**;** //add in end null symbol

str **=** **(**char **\*)**realloc**(**str**,** **(**1 **+** SLen**)** **\*** **sizeof(**char**));** //reallocation

system**(**"cls"**);**

**return** str**;**

**}**

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

ARGZ - string which is formed argz

ADD - string which would be added after call function "argz\_add"

DELETE - string which would be deleted after call function "argz\_delete"

BFR - string which would be sended in function "argz\_insert"

INSERT - string which would be inserted after call function "argz\_insert"

argz - array of strings

START - pointer to ellements of array

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

int main**()** **{**

char **\***ARGZ **=** EnterString**(**"first"**),** **\***ADD **=** EnterString**(**"added"**),** **\***DELETE **=** EnterString**(**"delete"**),**

**\***BFR **=** EnterString**(**"before"**),** **\***INSERT **=** EnterString**(**"insert"**),** **\***argz **=** 0**,** **\***START **=** 0**;**

size\_t **\***size **=** malloc**(sizeof(**int**));**

int sep**,** i**;**

char **\***c **=** malloc**(sizeof(**char**));**

printf**(**"Enter SEP\n"**);**

scanf**(**"%c"**,** c**);**

system**(**"cls"**);**

sep **=** **\***c**;**

free**(**c**);**

**\***size **=** 0**;**

**if** **(**argz\_create\_sep**(**ARGZ**,** sep**,** **&**argz**,** size**)** **==** OK**)** **{**

argz\_print**(**argz**,** **\***size**);**

**if** **(**argz\_add**(&**argz**,** size**,** ADD**)** **==** OK**)** argz\_print**(**argz**,** **\***size**);**

**else** printf**(**"Error of adding\n\n"**);**

argz\_delete**(&**argz**,** size**,** DELETE**);**

argz\_print**(**argz**,** **\***size**);**

**if** **(**argz\_insert**(&**argz**,** size**,** BFR**,** INSERT**)** **==** OK**)** argz\_print**(**argz**,** **\***size**);**

**else** printf**(**"Error of insert\n\n"**);**

**do** **{**

START **=** argz\_next**(**argz**,** **\***size**,** START**);**

**if** **(**START **!=** 0**)** **for** **(**i **=** 0**;** START**[**i**]** **!=** '\0'**;** i**++)** printf**(**"%c"**,** START**[**i**]);**

printf**(**"\n"**);**

**}** **while** **(**START **!=** 0**);**

**}** **else** printf**(**"Error of input\n\n"**);**

free**(**ARGZ**);**

free**(**ADD**);**

free**(**DELETE**);**

free**(**BFR**);**

free**(**INSERT**);**

free**(**START**);**

free**(**argz**);**

free**(**size**);**

\_getch**();**

**return** 0**;**

**}**

1. **Файл argz.c**

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

name: argz.c

description: this file contains function "argz\_create\_sep", that makes from

string array of string, where ellements of array are ellements

of string which separated by sep. every ellement of array

contains symbol '='. argz\_len get value of full argz length.

function type is error\_t - if we get argz, that contains at

least one ellement returns 'OK', else - 'ENOMEM'. argz\_len

got value of index of last null symbol + 1.

function "PosChr" returns index of first entering

in string parameter sep. if string doesn't contain parameter

sep then function returns constant INT\_MAX (see <limits.h>).

function "argz\_count" count number of ellements in array argz.

function "argz\_delete" delete first entering of ellement

entry in argz. it also change value of argz\_len.

function "argz\_insert" insert in argz string entry if

this string contains symbol '=' before element before. it

also change value of argz\_len. function return 'OK', if

string entry was inserted in argz, else return 'ENOMEM'

function "argz\_next" returns pointer to ellement of argz

next to entry, if argz contains ellement entry. If entry = 0, then

function returns pointer to argz. if argz doesn't contains

ellement after entry function returns 0

function "argz\_print" prints argz, argz\_len and number of

ellements in argz.

author: Dima

date of creation: 04.09.2014

written: 06.09.2014

date of last modified: 10.09.2014

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

#include <stdio.h>

#include <stdlib.h>

#include <limits.h>

#include <string.h>

#include "argz.h"

**typedef** enum **{** false**,** true **}** bool**;**

int PosChr**(**const char **\***string**,** int sep**)** **{** //i - index of char sep.

unsigned i**;**

**for** **(**i **=** 0**;** i **<** strlen**(**string**);** i**++)** **if** **(**string**[**i**]** **==** sep**)** **return** i**;**

**return** INT\_MAX**;**

**}**

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

char **\***STR **=** **(**char **\*)**malloc**(sizeof(**char**)\*** **(**strlen**(**string**)** **+** 2**));**

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

unsigned i**,** POS**,** LEN **=** strlen**(**string**);**

**for** **(**i **=** 0**;** i **<** strlen**(**string**);** i**++)** STR**[**i**]** **=** string**[**i**];**

STR**[**strlen**(**string**)]** **=** sep**;**

STR**[**strlen**(**string**)** **+** 1**]** **=** '\0'**;**

**while** **((**POS **=** PosChr**(**STR**,** sep**))** **!=** INT\_MAX**)** **if** **(**PosChr**(**STR**,** '='**)** **<** POS**)** **{**

**\***argz **=** **(**char **\*)**realloc**(\***argz**,** **sizeof(**char**)\*** **(**POS **+** 2 **+** **(\***argz\_len**)));**

**for** **(**i **=** 0**;** i **<** POS**;** i**++)** **(\***argz**)[(\***argz\_len**)++]** **=** STR**[**i**];**

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

**for** **(**i **=** 0**;** i **<** LEN **-** POS**;** i**++)** STR**[**i**]** **=** STR**[**i **+** POS **+** 1**];**

STR**[**LEN **-** POS**]** **=** '\0'**;**

LEN **=** strlen**(**STR**);**

**}** **else** **{**

**for** **(**i **=** 0**;** i **<** LEN **-** POS**;** i**++)** STR**[**i**]** **=** STR**[**i **+** POS **+** 1**];**

STR**[**LEN **-** POS**]** **=** '\0'**;**

LEN **=** strlen**(**STR**);**

**}**

free**(**STR**);**

**if** **(\***argz\_len **==** 0**)** **return** ENOMEM**;**

**else** **return** OK**;**

**}**

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

int i **=** 0**;**

size\_t COUNT **=** 0**;**

**for** **(**i **=** 0**;** i **<** arg\_len**;** i**++)** **if** **(**argz**[**i**]** **==** '\0'**)** COUNT**++;**

**return** COUNT**;**

**}**

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

int i**;**

**\***argz **=** **(**char **\*)**realloc**(\***argz**,** **sizeof(**char**)\*** **((\***argz\_len**)** **+** strlen**(**str**)** **+** 1**));**

**if** **(**PosChr**(**str**,** '='**)** **!=** INT\_MAX**)** **for** **(**i **=** 0**;** i **<=** strlen**(**str**);** i**++)** **(\***argz**)[(\***argz\_len**)++]** **=** str**[**i**];**

**else** **return** ENOMEM**;**

**return** OK**;**

**}**

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

int NEW\_LEN **=** 0**;**

int i**,** SP**;**

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

SP **=** i **=** 0**;**

**do** **{**

int j**,** NEW\_SP **=** SP**;**

bool flag **=** true**;**

**if** **((\***argz**)[**i**]** **==** '\0'**)** **{**

**for** **(**j **=** 0**;** j **<=** i **-** SP**;** j**++)** **if** **((\***argz**)[**j **+** SP**]** **!=** entry**[**j**])** **{**

flag **=** false**;**

**break;**

**}**

NEW\_SP **=** i **+** 1**;**

**}**

**if** **(**flag **==** false**)** **for** **(**j **=** SP**;** j **<** i **+** 1**;** j**++)** STR**[**NEW\_LEN**++]** **=** **(\***argz**)[**j**];**

SP **=** NEW\_SP**;**

**}** **while** **(++**i **<** **\***argz\_len**);**

free**(\***argz**);**

**\***argz **=** STR **=** **(**char **\*)**realloc**(**STR**,** **sizeof(**char**)\*** **(**NEW\_LEN **+** 1**));**

**\***argz\_len **=** NEW\_LEN**;**

**}**

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

int SP**,** i**,** SP\_WRD**;**

i **=** SP\_WRD **=** 0**;**

SP **=** **-**1**;**

**do** **if** **((\***argz**)[**i**]** **==** '\0'**)** **{**

int j**;**

bool flag **=** true**;**

**for** **(**j **=** 0**;** j **<=** i **-** SP\_WRD**;** j**++)** **if** **((\***argz**)[**j **+** SP\_WRD**]** **!=** before**[**j**])** flag **=** false**;**

**if** **(**flag **==** true**)** SP **=** SP\_WRD**;**

**else** SP\_WRD **=** i **+** 1**;**

**}**

**while** **(++**i **<** **\***argz\_len **&&** SP **==** **-**1**);**

**if** **(**SP **!=** **-**1 **&&** PosChr**(**entry**,** '='**)** **!=** INT\_MAX**)** **{**

int i**;**

**\***argz **=** **(**char **\*)**realloc**(\***argz**,** **sizeof(**char**)\*** **((\***argz\_len**)** **+** strlen**(**entry**)** **+** 1**));**

**for** **(**i **=** **\***argz\_len **-** 1**;** i **>** SP **-** 1**;** i**--)** **(\***argz**)[**i **+** strlen**(**entry**)** **+** 1**]** **=** **(\***argz**)[**i**];**

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

**for** **(**i **=** SP**;** i **<** SP **+** strlen**(**entry**)** **+** 1**;** i**++)** **(\***argz**)[**i**]** **=** entry**[**i **-** SP**];**

**}** **else** **return** ENOMEM**;**

**return** OK**;**

**}**

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

**if** **(**entry **==** 0**)** **return** argz**;**

**else** **{**

int i**,** SP**,** SP\_WRD**;**

i **=** SP\_WRD **=** 0**;**

SP **=** **-**1**;**

**while** **(**entry**[**i**++]** **!=** '\0'**);**

**if** **(&(**entry**[**i **-** 1**])** **!=** **&(**argz**[**argz\_len **-** 1**]))** **return** **&(**entry**[**i**]);**

**else** **return** 0**;**

**}**

**}**

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

unsigned i**;**

**for** **(**i **=** 0**;** i **<** argz\_len**;** i**++)** printf**(**"%c"**,** argz**[**i**]);**

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

printf**(**"Count of argz are %d\n\n"**,** argz\_count**(**argz**,** argz\_len**));**

**}**

1. **Файл argz.h**

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

name: argz.h

description: this file contains prototype of functions, that

described in "argz.c"

author: Dima

date of creation: 04.09.2014

written: 06.09.2014

date of last modified: 06.09.2014

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

**typedef** int size\_t**;**

**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**);**

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