Универзитет „Св. Кирил и Методиј“ - Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Structured programming

Exercises 10

Table of Contents

1. Strings

1.1. Remainders from lectures

1.1.1. String functions from `<string.h>`

String mutation functions:

- `strcpy()` - copy one string to another
- `strncpy()` - copy n bytes from one string to another, from `src` are copied or `nulls` are padded
- `strcat()` - concatenates string at the end of another
- `strncat()` - concatenates n bytes of one string to another

Memory changing functions:

- `memset()` - filles array with some byte

Functions for conversion string to numbers:

- `atof()` - converts string to decimal number
- `atoi()` - converts string to integer number

String checking functions:

- `strlen()` - returns the length of the string
- `strcmp()` - compares two strings
- `strncmp()` - compares n bytes from two strings
- `strchr()` - finds the first occurrence of given character in some string
- `strrchr()` - finds the last occurrence of given character not in a set of chars
- `strspn()` - finds the first occurrence of given character in some string in a given set of chars

- `strpbrk()` - finds the first occurrence of given character in some string in a given set of chars
- `strstr()` - finds the first occurrence of string in other string

1.1.2. Functions for characters from the library `<ctype.h>`

Functions for characters:

- `isalnum()` - checks if the character is alphanumeric (letter or number)
- `isalpha()` - checks if the character is letter
- `iscntrl()` - checks if the character is control
- `isdigit()` - checks if the character is digit
- `isxdigit()` - checks if the character is hex digit
- `isprint()` - checks if the character is printable
- `ispunct()` - checks if the character is punctuation
- `isspace()` - checks if the character is printable
- `islower()` - checks if the character is lowercase
- `isupper()` - checks if the character is uppercase
- `tolower()` - converts to lowercase
- `toupper()` - converts to uppercase
- `isgraph()` - checks if the character has local graphic representation

1.2. Problem 1

Write a function that will find how many times a given character occurs in given string.

Example

For the string

```
HELLO FINKI
```

character `L` occurs 2 times.

Solution `p10_1_en.c`

```
#include <stdio.h>
#define MAX 100
int count_char(char *str, char c) {
    int total = 0;
    while (*str != '\0') {
        total += (*str == c);
        str++;
    }
    return total;
}
int main() {
    char s[MAX], c;
    gets(s);
    c = getchar();
    printf("%d\n", count_char(s, c));
    return 0;
}
```

1.3. Problem 2

Write a function that will return the length of a string.

Also write a **recursive** solution.

Example

For string

`zdravo!`

it should return: `7`

Solution `p10_2_en.c`

```
#include <stdio.h>
#define MAX 100

int length(char *s) {
    int i, len = 0;
    for (i = 0; s[i] != '\0'; i++)
        len++;
    return len;
}

int length_r(char *s) {
    if (*s == '\0')
        return 0;
    return 1 + length_r(s + 1);
}

int main() {
    char s[MAX];
    gets(s);
    printf("Length: %d and %d\n", length(s), length_r(s));
    return 0;
}
```

1.4. Problem 3

Write a program that will print substring from given string, determined with the position and the length as parameters read from SI. The substring starts from the character on the position counted from left to right.

Example

For the string:

`banana`

position: `2`

length: `4`

the output is: `nana`

Solution `p10_3_en.c`

```
#include <stdio.h>
#include <string.h>
#define MAX 100

int main() {
    char s[MAX], dest[MAX];
    int position, length;
    gets(s);
    scanf("%d %d", &position, &length);
    if (position <= strlen(s)) {
        strncpy(dest, s + position - 1, length);
        dest[length] = '\0';
        printf("Result: ");
        puts(dest);
    } else
        printf("Invalid input, the read string has only %d characters.\n", strlen(s));
    return 0;
}
```

1.5. Problem 4

Write a function that will check if one string is substring of some other string.

Example

`face` is substring of `Please faceAbook`

Solution `p10_4_en.c`

```
#include <stdio.h>
#include <string.h>
#define MAX 100

int substring(char *s1, char *s2) {
    int i;
    int d1 = strlen(s1);
    int d2 = strlen(s2);
    if (d1 > d2)
        return 0;
    for (i = 0; i <= d2 - d1; i++)
        if (strncmp(s1, s2 + i, d1) == 0)
            return 1;
    return 0;
}
int main() {
    char s1[MAX], s2[MAX];
    gets(s1);
    gets(s2);
    if (substring(s1, s2))
        printf("%s is substring of %s\n", s1, s2);
    else
        printf("%s is NOT a substring of %s\n", s1, s2);
    return 0;
}
```

1.6. Problem 5

Write a function that will check if given string is palindrome. Palindrome is a string that is read same from left to right and from right to left.

Examples for palindromes

```
dovod
ana
kalabalak
```

Solution `p10_5_en.c`

```
#include <stdio.h>
#include <string.h>
#define MAX 100
int is_palindrome(char *str) {
    int i, n = strlen(str);
    for (i = 0; i < n / 2; i++)
        if (*(str + i) != *(str + n - 1 - i))
            return 0;;
    return 1;
}
// Recursive
int is_pal(char *str, int start, int end) {
    if (start >= end) return 1;
    if (str[start] == str[end])
        return is_pal(str, start + 1, end - 1);
    return 0;
}

int main() {
    char s[MAX];
    gets(s);
    printf("%s ", s);
    if (is_pal(s, 0, strlen(s) - 1))
        printf("is a palindrome.");
    else
        printf("is NOT a palindrome.");
    return 0;
}
```

1.6.1. Problem 5-a (homework)

Write a function that will check if given sentence is a palindrome. Ignore the empty spaces, punctuations characters and the case of letters.

Examples for sentences palindromes

```
Jadejne i pienje daj!
A man, a plan, a canal, Panama.
Never odd or even.
Rise to vote sir!
```

1.7. Problem 6

Write a function that for a given string will if it's complex enough to become a password. Every password must have at least one letter, one digit and one special character.

Example

`zdr@v0!` is a valid password.

`zdravo` is not a valid password.

Solution `p10_6_en.c`

```c
#include <stdio.h>
#include <string.h>
#define MAX 100

int is_valid_password(char *str) {
    int letters = 0;
    int digits = 0;
    int spec = 0;
    for (; *str; str++) {
        if (isalpha(*str))
            letters++;
        else if (isdigit(*str))
            digits++;
        else
            spec++;
    }
    return (letters > 0 && digits > 0 && spec > 0);
}

int main() {
    char s[MAX];
    gets(s);
    printf("%s ", s);
    if (is_valid_password(s))
        printf("is a valid password.");
    else
        printf("is NOT a valid password.");
    return 0;
}
```

1.8. Problem 7

Write a function that for will change the case of the letters and will remove all digits and special characters.

Example

For the string:

`0v@ePr1m3R`

the result should be:

`VEpRMr`

Solution `p10_7_en.c`

```
#include <stdio.h>
#include <string.h>
#define MAX 100

void filter(char *str) {
    int i = 0, j = 0;
    while (str[i] != '\0') {
        if (isalpha(str[i])) {
            if (islower(str[i]))
                str[j] = toupper(str[i]);
            else if (isupper(str[i]))
                str[j] = tolower(str[i]);
            j++;
        }
        i++;
    }
    str[j] = '\0';
}
int main() {
    char s[MAX];
    gets(s);
    filter(s);
    printf("%s\n", s);
    return 0;
}
```

1.9. Problem 8

Write a function that will trim a string (remove blanks at front and end of string).

Example

For the string:

```
"   make trim   "
```

the output should be:

```
"make trim"
```

Solution `p10_8_en.c`

```c
#include <stdio.h>
#include <string.h>
#define MAX 100

void trim(char *s) {
    char *d = s;
    while (isspace(*s++))
        ;
    s--;
    while (*d++ = *s++)
        ;
    d--;
    while (isspace(*--d))
        *d = 0;
}
int main() {
    char s[MAX];
    gets(s);
    printf("[%s] -> ", s);
    trim(s);
    printf("[%s]", s);
    return 0;
}
```

2. Source code of the examples and problems

https://github.com/finki-mk/SP/

Source code ZIP