Лабораторная работа № 4

Введение в функции.

Базовая работа со строками (однобайтовыми).

Задание 1.1

Постановка задачи:

Создайте две функции, которые вычисляют факториал числа:

• функцию, которая вычисляет факториал, используя цикл;

• функцию, которая вычисляет факториал, используя рекурсивный вызов самой себя.

Математическая модель:

$n!$

Список идентификаторов:

Имя	**Тип**	**Назначение**
a	int	Число для нахождения его факториала двумя функциями.

Код программы:

```c
#include <stdio.h>

unsigned loop_fact(int n);
unsigned recusion_fact(int n);


int
main()
{
    int a = 10;
    printf("recursion: %u\n",recusion_fact(a));
    printf("loop: %u\n",loop_fact(a));
    return 0;
}

unsigned
recusion_fact(int n)
{
    unsigned fact = 1;
    if (n == 1)
        return fact;
    else
        return n*recusion_fact(n-1);
}

unsigned
loop_fact(int n)
{
    unsigned fact = 1;
    for (int i = 1; i < n; i++)
        fact*=i;
    return fact;
}
```

Результаты выполненной работы:

```
recursion: 3628800
loop: 3628800
```

Задание 1.2

Постановка задачи:

Объявите указатель на массив типа int и динамически выделите память для 12-ти элементов. Напишите функцию, которая поменяет значения чётных и нечётных ячеек массива.

Математическая модель:

-

Список идентификаторов:

Имя	**Тип**	**Назначение**
LEN	int (macros)	Длинна массива.
ptr	int*	Указатель на динамический выделенный массив.
i	int	Аргумент циклов.
temp	int	Переменная для временного хранинеия элемента во время замены.

Код программы:

```c
#include <stdio.h>
#include <stdlib.h>

#define LEN 12

void swap_even_odd(int *, int);

int
main()
{
    int *ptr = NULL;
    ptr = (int *)(malloc(LEN*sizeof(int)));
    for(int i = 0; i<LEN; i++)
        ptr[i] = i;

    for(int i = 0; i<LEN; i++)
        printf("%d ", ptr[i]);
    putchar('\n');

    swap_even_odd(ptr, LEN);

    for(int i = 0; i<LEN; i++)
        printf("%d ", ptr[i]);
    putchar('\n');

    free(ptr);
    return 0;
}


void
swap_even_odd(int *array, int len)
{
    int temp;
    for(int i = 1; i<len; i+=2)
    {
        temp = array[i-1];
        array[i-1] = array[i];
        array[i] = temp;
    }
}
```

Результаты выполненной работы:

```
0 1 2 3 4 5 6 7 8 9 10 11
1 0 3 2 5 4 7 6 9 8 11 10
```

Задание 1.3

Постановка задачи:

Создать две основные функции:

• функцию для динамического выделения памяти под двумерный динамический массив типа double — матрицу;

• функцию для динамического освобождения памяти под двумерный динамический массив типа double — матрицу.

Создать две вспомогательные функции:

• функцию для заполнения матрицы типа double;

• функцию для распечатки этой матрицы на экране.

Продемонстрировать работу всех этих функций в своей программе.

Математическая модель:

-

Список идентификаторов:

Имя	**Тип**	**Назначение**
LEN	int (macros)	Длинна двуменого массива.
WID	int (macros)	Ширина двуменого массива.
matrix	double**	Указатель на динамический выделенный двумерный массив.
i	int	Аргумент циклов.
j	int	Аргумент циклов.
temp	int	Переменная для временного хранинеия элемента во время замены.

Код программы:

```c
#include <stdio.h>
#include <stdlib.h>

double **matrix_alloc(unsigned, unsigned);
void matrix_free(double **, unsigned, unsigned);

void matrix_print(double **, unsigned, unsigned);
void matrix_fill(double **, unsigned, unsigned);

#define LEN 3
#define WID 3

int
main()
{
    double **matrix = matrix_alloc(LEN, WID);
    matrix_fill(matrix, LEN, WID);
    matrix_print(matrix, LEN, WID);
    matrix_free(matrix, LEN, WID);
    return 0;
}

void matrix_fill(double **matrix, unsigned len, unsigned wid)
{
    for (int i = 0; i < len; i++){
        for (int j = 0; j < wid; j++){
            if (i == j)
                matrix[i][j] = 1;
            else
                matrix[i][j] = 0;
        }
    }
}

double **matrix_alloc(unsigned len, unsigned wid)
{
    double **matrix = NULL;
    matrix = (double **)(malloc(wid*sizeof(double *)));
    for(int i = 0; i < wid; i++){
            matrix[i] = malloc(sizeof(double));
    }
    return matrix;
}

```

```c
void matrix_free(double ** matrix, unsigned len, unsigned wid)
{
    for(int i = 0; i < wid; i++)
            free(matrix[i]);
    free(matrix);
}


void
matrix_print(double **matrix, unsigned len, unsigned wid)
{
    for (int i = 0; i < len; i++){
        for (int j = 0; j < wid; j++){
            printf("%lf ", matrix[i][j]);
        }
        putchar('\n');
    }
}
```

Результаты выполненной работы:

```
1.000000 0.000000 0.000000
0.000000 1.000000 0.000000
0.000000 0.000000 1.000000
```

Задание 1.4

Постановка задачи:

Создать функцию, которая вычисляет векторное произведение двух векторов в декартовых координатах, используя указатели на соответствующие массивы.

Математическая модель:

-

Список идентификаторов:

Имя	**Тип**	**Назначение**
LEN	int (macros)	Длинна двуменого массива.
WID	int (macros)	Ширина двуменого массива.
matrix	double**	Указатель на динамический выделенный двумерный массив.
i	int	Аргумент циклов.
j	int	Аргумент циклов.
temp	int	Переменная для временного хранинеия элемента во время замены.

Код программы:

```c
#include <stdio.h>

void
vecProduct(double *a, double *b, double *rez)
{
    rez[0] = a[1] * b[2] - a[2] * b[1];
    rez[1] = a[2] * b[0] - a[0] * b[2];
    rez[2] = a[0] * b[1] - a[1] * b[0];
}

int
main()
{
    double a[3] = {10, 11, 12};
    double b[3] = {13, 14, 15};
    double c[3];

    vecProduct(a, b, c);
    printf("Product of a and b = (%lf, %lf, %lf)\n", c[0], c[1], c[2]);
    return 0;
}
```

Результаты выполненной работы:

```
Product of a and b = (-3.000000, 6.000000, -3.000000)
```

Задание 2.1

Постановка задачи:

Создайте новую программу, где с клавиатуры вводится строка некоторой длины порядка 10 латинских символов (не используйте кириллицу) в классическую строку языка C, которая имеет вид массива char my_string[MY_SIZE]. MY_SIZE определите с помощью директивы #define. Значение MY_SIZE должно превышать длину вводимой строки с некоторым разумным запасом. Другие строки в этой задаче можете создавать либо также как статические массивы, либо как динамические массивы, но не забывайте освобождать от динамически выделенную память с помощью функции void free(void∗ ptr); .

Выполните следующие действия и распечатайте результаты:

1. Вычислите длину строки my_string, используя цикл for и тот факт, что в языке C такие строки имеют в конце специальный нулевой символ конца строки, представленный escape-последовательностью '\0' ('...' — это тип char).

2. Сделайте тоже самое, что в пункте 1, но создайте указатель на начало вашей строки и используйте операцию инкремента ++.

3. Используйте функции

size_t strlen(const char∗ str); или size_t strnlen (const char ∗string, size_t maxlen); или size_t strnlen_s(const char ∗str, size_t strsz); для получения размера строки в виде значения size_t (псевдоним unsigned int, спецификатор форматирования — "%zu"). Убедитесь, что ваш компилятор явно работает с опцией-std=c11 или с опцией для более позднего стандарта языка для поддержки функции strnlen_s.

4. Создайте вторую строку (второй массив) и скопируйте в неё строку my_string, используя функцию char ∗strcpy(char ∗dest, const char ∗src); или char ∗strncpy (char ∗dest, const char ∗src, size_t n);.

5. Создайте ещё две строки какого-либо размера и задайте их прямо в коде без клавиатуры. Сделайте конкатенацию этих двух строк, используя

char ∗strcat(char ∗dest, const char ∗src); или char ∗strncat(char ∗dest, const char ∗src, size_t n);. Первую строку трактуйте как dest (destination) и подберите размер этого массива с запасом.

6. Сравните две новые строки, заданные в коде строковыми лите- ралами, используя функцию int strcmp(const char ∗lhs, const char ∗rhs); или int strncmp (const char ∗s1, const char ∗s2, size_t n).

7. Задайте прямо в коде строку, в которой есть только латинские символы в верхнем и нижнем регистре. Переведите строку полностью в нижний регистр и отдельно полностью в верхний регистр. Распечатайте каждый результат отдельно. Найдите сигнатуры подходящих функций (tolower и toupper), изучив базовые однобайтовые строковые функции по ссылке https://en.cppreference.com/w/c/string/byte.

Математическая модель:

-

Список идентификаторов:

Имя	**Тип**	**Назначение**
MY_SIZE	int(macros)	Длинна строк.
my_string	char [MY_SIZE]	Первая строка.
len	int	Длинна первой строки полученная циклом for.
iter		Указатель на элемент строки my_string для поиска ее длинны
len2	size_t	Длинна строки полученная при помощи функции strlen.
my_string2	char [MY_SIZE]	Вторая строка в которую будет скопирована первая.
my_string3	char [MY_SIZE]	Третья строка которая будет конкатенирована с 4.
my_string4	char [MY_SIZE*2]	Четвёртая строка которая будет конкатенирована с 3.
rez	int	Результат сравнения строк «ab» и «ba».
my_string5	char [MY_SIZE]	Строка с символами разных регистров для приведения к нижнему и верхнему регистрам.

Код программы:

```c
#include <ctype.h>
#include <string.h>
#include <stdio.h>
#include <locale.h>

#define MY_SIZE 15

int
main()
{
    setlocale(LC_ALL,"en_US.iso88591");
    char my_string[MY_SIZE];
    fscanf(stdin, "%s", my_string);
    int len;
    for(len = 0; my_string[len] != '\0'; len++)
        continue;

    char *iter;
    for(len = 0, iter = &my_string; *iter != '\0'; iter++, len++)
        continue;

    size_t len2 = strlen(my_string);

     char my_string2[MY_SIZE] = "ABCDEFG";
     strcpy(my_string2, my_string);

     char my_string3[MY_SIZE] = "abcd";
     char my_string4[MY_SIZE*2] = "efgn";
     strcat(my_string4,my_string3);

    printf("string1:%s\n\tlen: %d\n\tsize_t len2: %zu\nstring2:%s\nstring3:%s\n",my_s
tring, len, len2, my_string2, my_string3);

     int rez = strcmp("ab", "ba");

    char my_string5[MY_SIZE] = "QwErTyUiOpAsDf";

    for(len = 0; len < MY_SIZE; len++)
        my_string5[len] = tolower(my_string5[len]);
    printf("tolower: %s\n", my_string5);

    for(len = 0; len < MY_SIZE; len++)
        my_string5[len] = toupper(my_string5[len]);
    printf("toupper: %s\n", my_string5);
    return 0;
}
```

Результат выполненной работы:

```
abcdefg
string1:abcdefg
        len: 7
        size_t len2: 7
string2:abcdefg
string3:abcd
tolower: qwertyuiopasdf
toupper: QWERTYUIOPASDF
```

Задание 2.2

Постановка задачи:

Конвертируйте введённые заданные как строки: число с плавающей точкой (double) и целое число (int) в значения типа double и int, используя функциями atof и atoi. См. Документацию по ссылке https://en.cppreference.com/w/c/string/byte.

Математическая модель:

-

Список идентификаторов:

Имя	**Тип**	**Назначение**
string1	string1	Строка с вещественным числом.
string1	string1	Строка с целым числом.
double1	double	Конвертированная в число первая строка.
decimal2	decimal	Конвертированная в число вторая строка.

Код программы:

```c
#include <stdio.h>
#include <stdlib.h>

int
main()
{
    char string1[10] = "1.23456";
    char string2[10] = "1234567";
    double double1 = atof(string1);
    int decimal2 = atoi(string2);
    printf("%lf\n%d\n", double1, decimal2);
    return 0;
}
```

Результат выполненной работы:

```
1.234560
1234567
```

Задание 2.3

Постановка задачи:

Создайте строку от 10 до 20 символов, используя только цифры, латинский буквы в разных регистрах пробельные символы и символы пунктуации. Организуйте цикл, где каждый символ подробно тестируется функциями типа int is∗(/∗... ∗/) (например — isdigit, ispunct). См. документацию по ссылке https://en.cppreference.com/w/c/string/byte. Оформите распечатку информации по каждому символу в виде списка на экране, чтобы можно было прочесть информацию о том что представляет из себя каждый символ (своими словами, в свободной форме). Постарайтесь использовать только латиницу.

Математическая модель:

-

Список идентификаторов:

Имя	**Тип**	**Назначение**
SIZE	int(macors)	Длинна строки string.
string	string[SIZE]	Вводимая пользователем строка, информация о символах которой будет выводится в терминал.
i	int	Аргумент цилка.

Код программы:

```c
#include <stdio.h>
#include <ctype.h>
#include <string.h>

#define SIZE 25

int
main()
{
    char string[SIZE];
    fgets(string, SIZE, stdin );

    for(int i = 0; i < strlen(string); i++)
    {
        printf("%c)[%d]", string[i], i);
        if(isascii(string[i]))
            printf("is ascii ");
        else{
            printf("is non-ascii ");
            continue;
        }
        if(isprint(string[i])){
            printf("printable ");
        }else
            printf("non-printable");
        if(isblank(string[i]))
            printf("blank");
        else if(isdigit(string[i]))
            printf("digit");
        else if(islower(string[i]))
            printf("lower case letter");
        else if(isupper(string[i]))
            printf("upper case letter");
        printf(" character №: %d in ascii table", string[i]);
        putchar('\n');
    }
    return 0;
}
```

Результат выполненной работы:

```
12345 AbCdEfG ^L
1)[0]is ascii printable digit character №: 49 in ascii table
2)[1]is ascii printable digit character №: 50 in ascii table
3)[2]is ascii printable digit character №: 51 in ascii table
4)[3]is ascii printable digit character №: 52 in ascii table
5)[4]is ascii printable digit character №: 53 in ascii table
 )[5]is ascii printable blank character №: 32 in ascii table
A)[6]is ascii printable upper case letter character №: 65 in ascii table
b)[7]is ascii printable lower case letter character №: 98 in ascii table
C)[8]is ascii printable upper case letter character №: 67 in ascii table
d)[9]is ascii printable lower case letter character №: 100 in ascii table
E)[10]is ascii printable upper case letter character №: 69 in ascii table
f)[11]is ascii printable lower case letter character №: 102 in ascii table
G)[12]is ascii printable upper case letter character №: 71 in ascii table
 )[13]is ascii printable blank character №: 32 in ascii table

)[14]is ascii non-printable character №: 12 in ascii table

)[15]is ascii non-printable character №: 10 in ascii table
```