Лабораторная работа №4

Тема Лабораторной работы: 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.

Список идентификаторов:

Имя	**Тип**	**Смысл**
func1	int	Функция
func2	int	Функция
x	int	Аргумент func1 и func2
a	int	Промежуточная переменная
i	int	Параметр цикла

Код программы:

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

int func1 (int x)
{
int a=1;
for (int i=1;i<=x;i++)
{
a=a*i;
}
return a;
}

int func2 (int x)
{
if (x==1)
{
return 1;
}
else
{
return x*func2(x-1);
}
}

int main(void)
{
printf("V1=%d\n",func1(16));
printf("V2=%d\n",func2(16));
}
```

Результат выполненной работы:

```
V1=2004189184
V2=2004189184
```

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

Список идентификаторов:

Имя	Тип	Смысл
swap	void	Функция
a	int	Указатель
i	int	Параметр цикла

temp	int	Промежуточная переменная
size	int	Размер массива

Код программы:

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

void swap(int* a, int size) {
    for (int i = 0; i < size - 1; i += 2) {
        int temp = a[i];
        a[i] = a[i + 1];
        a[i + 1] = temp;
    }
}

int main() {
    int* a = (int*)malloc(12 * sizeof(int));
    for (int i = 0; i < 12; i++) {
        a[i] = i + 1;
    }

    printf("Bef swap: ");
    for (int i = 0; i < 12; i++) {
        printf("%d ", a[i]);
    }
    printf("\n");

    swap(a, 12);

    printf("Aft swap: ");
    for (int i = 0; i < 12; i++) {
        printf("%d ", a[i]);
    }
    printf("\n");

    free(a);

    return 0;
}
```

Результат выполненной работы:

```
Bef swap: 1 2 3 4 5 6 7 8 9 10 11 12
Aft swap: 2 1 4 3 6 5 8 7 10 9 12 11
```

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.

Список идентификаторов:

Имя	**Тип**	**Смысл**
C_M	double	Функция, выделяет память под двумерный массив
F_M	void	Функция, освобождает память
W_M	void	Функция, заполняет матрицу значениями
P_M	void	Функция, выводит матрицу на экран
M	double	Указатель
rows	int	Входная переменная
cols	int	Входная переменная
i	int	Параметр цикла
j	int	Параметр цикла

Код программы:

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

double **C_M(int rows, int cols) {
    double **M = (double **) malloc(rows * sizeof(double *));
    for (int i = 0; i < rows; i++) {
        M[i] = (double *) malloc(cols * sizeof(double));
    }
    return M;
}

void F_M(double **M, int rows) {
    for (int i = 0; i < rows; i++) {
        free(M[i]);
    }
    free(M);
}

void W_M(double **M, int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("Enter element [%d][%d]: ", i, j);
            scanf("%lf", &M[i][j]);
        }
    }
}

void P_M(double **M, int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%lf ", M[i][j]);
        }
        printf("\n");
    }
}

int main() {
    int rows, cols;
    printf("Enter number of rows: ");
```

```
    printf("Enter number of rows: ");
    scanf("%d", &rows);
    printf("Enter number of columns: ");
    scanf("%d", &cols);

    double **M = C_M(rows, cols);
    W_M(M, rows, cols);
    printf("Matrix:\n");
    P_M(M, rows, cols);
    F_M(M, rows);

    return 0;
}
```

Результат выполненной работы:

```
Enter element [0][1]: 2
Enter element [0][2]: 3
Enter element [1][0]: 4
Enter element [1][1]: 5
Enter element [1][2]: 6
Enter element [2][0]: 7
Enter element [2][1]: 8
Enter element [2][2]: 9
Matrix:
1.000000 2.000000 3.000000
4.000000 5.000000 6.000000
7.000000 8.000000 9.000000
```

Задание 1.4

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

Список идентификаторов:

Имя	**Тип**	**Смысл**
Vec_Proiz	void	Функция, вычисляет произведение матриц
a	double	Значение элемента первой матрицы
b	double	Значение элемента второй матрицы
result	double	хранит результат векторного произведения

Код программы:

```c
#include <stdio.h>

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

int main() {
    double a[3], b[3], result[3];

    printf("Vector A (x, y, z): ");
    scanf("%lf %lf %lf", &a[0], &a[1], &a[2]);

    printf("Vector B (x, y, z): ");
    scanf("%lf %lf %lf", &b[0], &b[1], &b[2]);

    Vec_Proiz(a, b, result);

    printf("A*B: (%lf, %lf, %lf)\n", result[0], result[1], result[2]);

    return 0;
}
```

Результат выполненной работы:

```
Vector A (x, y, z): 1 2 3
Vector B (x, y, z): 4 5 6
A*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);

Список идентификаторов:

Имя	Тип	Смысл
MY_SIZE	const	Максимальная длина строки
my_string	char	Вводимая строка
fgets	char	Функция

Код программы:

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

#define MY_SIZE 20

int main() {
    char my_string[MY_SIZE];
    printf(":Write ");
    fgets(my_string, MY_SIZE, stdin);
    printf("%s\n", my_string);

    free(my_string);

    return 0;
}
```

Результат выполненной работы:

```
:Write JOJ
JOJ
```

Задание 2.1.1

Постановка задачи: 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).

Список идентификаторов:

Имя	Тип	Смысл
MY_SIZE	const	Максимальная длина строки
my_string	char	Вводимая строка
fgets	char	Функция

count	int	Счётчик
i	int	Параметр цикла

Код программы:

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

#define MY_SIZE 20

int main()
{
    char* my_string = (char*)malloc(MY_SIZE * sizeof(char));
    fgets(my_string, MY_SIZE, stdin);

    int count = 0;
    for(int i = 0; my_string[i] != '\0'; i++)
    {
        count++;
    }

    printf("%d\n", count -1);

    free(my_string);

    return 0;
}
```

Результат выполненной работы:

```
jpji
4
```

Задание 2.1.2

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

Список идентификаторов:

Имя	**Тип**	**Смысл**
MY_SIZE	const	Максимальная длина строки
my_string	char	Вводимая строка

fgets	char	Функция
count	int	Счётчик
ptr	char	Указатель

Код программы:

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

#define MY_SIZE 20

int main()
{
    char* my_string = (char*)malloc(MY_SIZE * sizeof(char));
    fgets(my_string, MY_SIZE, stdin);

    char* ptr = my_string;
    int count = 0;
    while (*ptr != '\0')
    {
        count++;
        ptr++;
    }

    printf("%d\n", count - 1);

    free(my_string);

    return 0;
}
```

Результат выполненной работы:

```
KOlaps
6
```

Задание 2.1.3

Постановка задачи: Используйте функции size_t strlen(const char∗ str); илиsize_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.

Список идентификаторов:

Имя	Тип	Смысл
MY_SIZE	const	Максимальная длина строки
my_string	char	Вводимая строка
fgets	char	Функция
strlen	int	Функция

Код программы:

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

#define MY_SIZE 20

int main()
{
    char* my_string = (char*)malloc(MY_SIZE * sizeof(char));
    fgets(my_string, MY_SIZE, stdin);

    printf("%d", strlen(my_string)-1);

    free(my_string);

    return 0;
}
```

Результат выполненной работы:

```
pops
4
```

Задание 2.1.4

Постановка задачи: 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);.

Список идентификаторов:

Имя	Тип	Смысл
MY_SIZE	const	Максимальная длина строки
my_string	char	Указатель на вводимую строку
fgets	char	Функция
my_string_copy	char	Указатель на строку, куда будет скопирована строка
strcpy	char	Функция, для копирования строки

Код программы:

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

#define MY_SIZE 20

int main()
{
    char* my_string = (char*)malloc(MY_SIZE * sizeof(char));
    fgets(my_string, MY_SIZE, stdin);

    char* my_string_copy = (char*)malloc(MY_SIZE * sizeof(char));
    strcpy(my_string_copy, my_string);

    printf("Original string: %s\n", my_string);
    printf("Copied string: %s\n", my_string_copy);

    free(my_string);
    free(my_string_copy);

    return 0;
}
```

Результат выполненной работы:

```
exers
Original string: exers

Copied string: exers
```

Задание 2.1.5

Постановка задачи: 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.

Список идентификаторов:

Имя	**Тип**	**Смысл**
MY_SIZE	const	Максимальная длина строки
my_string	char	Указатель на вводимую строку
fgets	char	Функция
strcat	char	Функция, для объединения строк
strcpy	char	Функция, для копирования строки
str1	char	указатель на строку "Hello, "
str2	char	указатель на строку "world!"
result	char	указатель на массив символов, в который будет записана объединенная строка

Код программы:

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

#define MY_SIZE 20

int main()
{
    char* my_string = (char*)malloc(MY_SIZE * sizeof(char));
    fgets(my_string, MY_SIZE, stdin);

    char* str1 = "Hello, ";
    char* str2 = "world!";
    char* result = (char*)malloc((strlen(str1) + strlen(str2) + 1) * sizeof(char));
    strcpy(result, str1);
    strcat(result, str2);
    printf("%s\n", result);

    free(my_string);
    free(result);

    return 0;
}
```

Результат выполненной работы:

```
lopser
Hello, world!
```

Задание 2.1.6

Постановка задачи: Сравните две новые строки, заданные в коде строковыми литералами, используя функцию int strcmp(const char *lhs, const char *rhs); или int strncmp (const char *s1, const char *s2, size_t n).

Список идентификаторов:

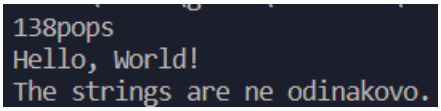
Имя	Тип	Смысл
MY_SIZE	const	Максимальная длина строки
my_string	char	Указатель на вводимую строку
fgets	char	Функция
strcat	char	Функция, для объединения строк
strcpy	char	Функция, для

		копирования строки
str1	char	указатель на строку "Hello, "
str2	char	указатель на строку "world!"
result	char	указатель на массив символов, в который будет записана объединенная строка
strcmp	char	Функция для сравнения строк

Код программы:

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

#define MY_SIZE 20

int main()
{
    char* my_string = (char*)malloc(MY_SIZE * sizeof(char));
    fgets(my_string, MY_SIZE, stdin);

    char* str1 = "Hello, ";
    char* str2 = "World!";
    char* result = (char*)malloc((strlen(str1) + strlen(str2) + 1) * sizeof(char));
    strcpy(result, str1);
    strcat(result, str2);
    printf("%s\n", result);

    if (strcmp(my_string, result) == 0) {
        printf("The strings are odinakovo.\n");
    } else {
        printf("The strings are ne odinakovo.\n");
    }

    free(my_string);
    free(result);

    return 0;
}
```

Результат выполненной работы:



Задание 2.1.7

Постановка задачи: 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

Список идентификаторов:

Имя	**Тип**	**Смысл**
MY_SIZE	const	Максимальная длина строки
my_string	char	Указатель на вводимую строку
fgets	char	Функция
strcat	char	Функция, для объединения строк
strcpy	char	Функция, для копирования строки
str1	char	указатель на строку "Hello, "
str2	char	указатель на строку "world!"
result	char	указатель на массив символов, в который будет записана объединенная строка
strcmp	char	Функция для сравнения строк
lowercase_string	char	Указатель на массив символов, в который будет записана строка в нижнем регистре
uppercase_string	char	Указатель на массив символов, в который будет записана строка в верхнем регистре
toupper	char	Функция для преобразования

		символа в верхний регистр
tolower	char	Функция для преобразования символа в нижний регистр
i	int	Параметр цикла

Код программы:

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

#define MY_SIZE 20

int main()
{
    char* my_string = (char*)malloc(MY_SIZE * sizeof(char));
    fgets(my_string, MY_SIZE, stdin);

    char* lowercase_string = (char*)malloc(strlen(my_string) * sizeof(char));
    strcpy(lowercase_string, my_string);
    for (int i = 0; i < strlen(lowercase_string); i++) {
        if (lowercase_string[i] >= 'A' && lowercase_string[i] <= 'Z') {
            lowercase_string[i] += 32;
        }
    }
    printf("%s\n", lowercase_string);

    char* uppercase_string = (char*)malloc(strlen(my_string) * sizeof(char));
    strcpy(uppercase_string, my_string);
    for (int i = 0; i < strlen(uppercase_string); i++) {
        if (uppercase_string[i] >= 'a' && uppercase_string[i] <= 'z') {
            uppercase_string[i] -= 32;
        }
    }
    printf("%s\n", uppercase_string);

    char* str1 = "Hello, ";
    char* str2 = "World!";
    char* result = (char*)malloc((strlen(str1) + strlen(str2) + 1) * sizeof(char));
    strcpy(result, str1);
    strcat(result, str2);
    printf("%s\n", result);

    if (strcmp(my_string, result) == 0) {
        printf("The strings are equal.\n");
```

```
    } else {
        printf("The strings are ne odinakovo.\n");
    }

    free(my_string);
    free(lowercase_string);
    free(uppercase_string);
    free(result);

    return 0;
}
```

Результат выполненной работы:

```
ROkoPn
rokopn

ROKOPN

Hello, World!
The strings are ne odinakovo.
```

Задание 2.2

Постановка задачи: Конвертируйте введённые заданные как строки: число с плавающей точкой (double) и целое число (int) в значения типаdouble и int, используя функциями atof и atoi

Список идентификаторов:

Имя	Тип	Смысл
str_double	char	Массив символов, содержащий строковое представление числа с плавающей точкой
str_int	char	Массив символов, содержащий строковое представление целого числа
num_double	double	Переменная, в которую будет записано преобразованное число с плавающей точкой
num_int	int	Переменная, в которую будет

		записано преобразованное целое число
atof		Функция для преобразования строки в число с плавающей точкой
atoi		Функция для преобразования строки в целое число

Код программы:

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

int main() {
    char str_double[] = "3.14159";
    char str_int[] = "42";
    double num_double = atof(str_double);
    int num_int = atoi(str_int);
    printf("num_double = %f\n", num_double);
    printf("num_int = %d\n", num_int);
    return 0;
}
```

Результат выполненной работы:

```
num_double = 3.141590
num_int = 42
```

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∗(/∗... ∗/)

Список идентификаторов:

Имя	**Тип**	**Смысл**
string	char	указатель
length	int	Длина строки
isalpha		функция для определения,

		является ли символ буквой
isupper		функция для определения, является ли символ заглавной буквой
islower		функция для определения, является ли символ строчной буквой
isdigit		функция для определения, является ли символ цифрой
isspace		функция для определения, является ли символ пробелом
ispunct		функция для определения, является ли символ знаком препинания

Код программы:

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

int main() {
    char *string = "3Ga- 53F,p";
    int length = strlen(string);

    for (int i = 0; i < length; i++) {
        printf(" Symbol: %c\n", string[i]);
        if (isalpha(string[i])) {
            if (isupper(string[i])) {
                printf("Big letter.");
            } else if (islower(string[i])) {
                printf("Low letter.");
            }
        } else if (isdigit(string[i])) {
            printf("Cifra.");
        } else if (isspace(string[i])) {
            printf("Space.");
        } else if (ispunct(string[i])) {
            printf("Mark.");
        } else {
            printf("This character is not printable.\n");
        }
    }

    return 0;
}
```

Результат выполненной работы:

```
 Symbol: 3

Cifra. Symbol: G

Big letter. Symbol: a

Low letter. Symbol: -

Mark. Symbol:

Space. Symbol: 5

Cifra. Symbol: 3

Cifra. Symbol: F

Big letter. Symbol: ,

Mark. Symbol: p

Low letter.
```