Лабораторная работа №4

Введение в функции. Базовая работа со строками (однобайтовыми)

Выполнил:

Студент 1-го курса

Группы ИВТ-1.1

Шардт Максим Александрович

1. Комплект 1: Введение в функции

1.1. Создайте две функции, которые вычисляют факториал числа: • функцию, которая вычисляет факториал, используя цикл; • функцию, которая вычисляет факториал, используя рекурсивный вызов самой себя. Продемонстрируйте работу обеих функций.

Математическая модель

$$n! = 1 \cdot 2 \cdot \ldots \cdot n = \prod_{k=1}^{n} k.$$

Список идентификаторов

Имя переменной	*Тип данных*	*Смысловое значение*
n	Long Integer	Факториал
result	Long Integer	Факториал числа

Код программы

```c
#include <stdio.h>

long int fact(long int n)
{
   if (n == 0)
       return 1;

   long int result = 1;
   for (int i = 1; i <= n; i++)
   {
       result *= i;
   }
```

```c

    return result;
}

long int fact_recursive(long int n)
{
    if (n == 0 || n == 1)
        return 1;
    else
        return n * fact_recursive(n - 1);
}
int main(void)
{

    printf("fact(5): %ld\n", fact(5));
    printf("fact(5) rec: %ld\n", fact_recursive(5));
    return 0;
}
```

Листинг 1: Lab4 - 1-1.c

Результат выполненной работы

```
fact(5): 120
fact(5) rec: 120
```

Выводится значение факториалов от 5.

1.2. 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

- **Чётное число** — целое число, которое *делится* на 2 без остатка: ..., −4, −2, **0**, 2, 4, 6, 8, ...
- **Нечётное число** — целое число, которое *не делится* на 2 без остатка: ..., −3, −1, 1, 3, 5, 7, 9, ...

Если m чётно, то оно представимо в виде $m = 2k$, а если нечётно, то в виде $m = 2k + 1$, где $k \in \mathbb{Z}$.

Список идентификаторов

Имя переменной	*Тип данных*	*Смысловое значение*

array	Integer	Массив значений
temp	Integer	Временная переменная
ARRAY_SIZE		Размер массива

Код программы

```c
#include <stdio.h>
#include <stdlib.h>
#define ARRAY_SIZE 12

void array_swap(int *array, int size)
{
   for (int i = 0; i < size; i += 2)
   {
       int temp = array[i];
       array[i] = array[i + 1];
       array[i + 1] = temp;
   }
}
int main(void)
{
   int *array = (int *)malloc(sizeof(int) * ARRAY_SIZE);

   for (int i = 0; i < ARRAY_SIZE; i++)
       array[i] = i + 1;

   array_swap(array, ARRAY_SIZE);

   for (int i = 0; i < ARRAY_SIZE; i++)
       printf("[%d]: %d\n", i, array[i]);

   free(array);
   return 0;
}

#undef ARRAY_SIZE
```

Листинг 2: Lab4 - 1-2.c

Результат выполненной работы

```
[0]: 2
[1]: 1
[2]: 4
[3]: 3
[4]: 6
[5]: 5
[6]: 8
[7]: 7
[8]: 10
[9]: 9
[10]: 12
[11]: 11
```

1.3. 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

Выделяем память для матрицы n на m, заполняем случайными значениями от 1 до 100 и выводим на экран.

Список идентификаторов

Имя переменной	*Тип данных*	*Смысловое значение*
n_rows	Integer	Число строк
n_columns	Integer	Число столбцов
matrix	Double	Матрица

Код программы

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

```
4
5  double **alloc_matrix(int n_rows, int n_columns)
6  {
7     double **matrix = (double **)malloc(n_rows * sizeof(double));
8
9     for (int i = 0; i < n_rows; i++)
10        matrix[i] = (double *)malloc(n_columns * sizeof(double));
11
12    return matrix;
13 }
14
15 void free_matrix(double **matrix, int n_rows, int n_columns)
16 {
17    for (int i = 0; i < n_rows; i++)
18        free(matrix[i]);
19    free(matrix);
20 }
21
22 void fill_matrix(double **matrix, int n_rows, int n_columns)
23 {
24    srand(time(NULL));
25    for (int i = 0; i < n_rows; i++)
26        for (int j = 0; j < n_columns; j++)
27            matrix[i][j] = random() % 101;
28 }
29
30 void print_matrix(double **matrix, int n_rows, int n_columns)
31 {
32    printf("Matrix:\n");
33    for (int i = 0; i < n_rows; i++)
34    {
35        for (int j = 0; j < n_columns; j++)
36            printf("%3.0lf ", matrix[i][j]);
37        printf("\n");
38    }
39 }
40
41 int main(void)
42 {
43    int n_rows = 10, n_columns = 10;
44    double **matrix = alloc_matrix(n_rows, n_columns);
45
46    fill_matrix(matrix, n_rows, n_columns);
```

```
47
48      print_matrix(matrix, n_rows, n_columns);
49
50      free_matrix(matrix, n_rows, n_columns);
51
52      return 0;
53  }
```

Листинг 3: Lab4 - 1-3.c

Результат выполненной работы

```
Matrix:
 13  75  80  58   9  73  36  29  27  74
 65  89  82   6  90   4  73  16  98  34
 33  37  21  26  22  31  23  91  40  63
 38  19   4  18  77  13  57  12  42  51
 53   6  39  34  79  95   5  18  77   2
 18  10  39  39   2  28  70  92  85   9
 21  23  96  25   7  72  38  30  51  80
 81   3  52  86   3  98  80   8  15  57
 78  33  33  16  73   1  10   8  93  96
 18  13  85  13  38  58  51  42  88  68
```

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

Математическая модель

Координаты вектора: $\vec{a}(x_a;y_a;z_a) \Leftrightarrow \vec{a}=x_a\vec{i}+y_a\vec{j}+z_a\vec{k}$

Список идентификаторов

Имя переменной	*Тип данных*	*Смысловое значение*
a	Integer	Вектор A
b	Integer	Вектор B
vector	Integer	Векторное произведение A и B

Код программы

```c
#include <stdio.h>

void cross_product(int *a, int *b, int *target)
{
   target[0] = a[1] * b[2] - a[2] * b[1];
   target[1] = a[0] * b[2] - a[2] * b[0];
   target[2] = a[0] * b[1] - a[1] * b[0];
}

int main(void)
{

   int a[] = {-1, 2, -3};
   int b[] = {0, -4, 1};
   int vector[2];
   cross_product(a, b, vector);

   printf("cross: %d %d %d \n", vector[0], vector[1], vector[2]);

   return 0;
}
```

Листинг 4: Lab4 - 1-4.c

Результат выполненной работы

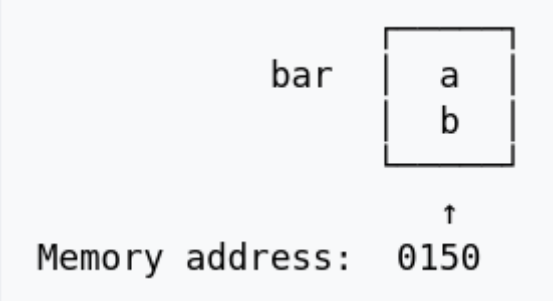
```
cross: -10 -1 4
```

2. Комплект 2: Базовые операции со строками.

2.1. Создайте новую программу, где с клавиатуры вводится строка некоторой длины порядка 10 латинских символов (не используйте кириллицу) в классическую строку языка С, 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 языкеС 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 3 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 литералами, 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),

Математическая модель



Список идентификаторов

Имя переменной	*Тип данных*	*Смысловое значение*
my_string	char[]	Изначальная строка
string_size	char[]	Размер строки
first_str	char[]	Первая строка
second_str	char[]	Вторая строк
third_str	char[]	Соединение первой и второй строки

Код программы

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

#define MY_SIZE 10

int main(void)
{
   setlocale(LC_ALL, "en_US.iso88591");

   char my_string[MY_SIZE];
   printf("Enter string: ");
   fscanf(stdin, "%s", my_string);

   int string_size = 0;
   // Первый способ нахождения длины
   for (int i = 0; my_string[i]; i++)
   {
       string_size = i;
   }
   // Второй
   string_size = 0;
   int i = 0;
   for (char *p = my_string; *p; p++)
   {
       string_size = i;
       i++;
```

```c
    }
    // Третий
    size_t s = strlen(my_string);

    printf("String size: %d\n", string_size);
    // Копия строк
    char my_string_copy[MY_SIZE];
    strcpy(my_string_copy, my_string);
    // Соединение строк
    char first_str[] = "foo";
    char second_str[] = "bar";
    char third_str[6];
    strcat(third_str, first_str);
    strcat(third_str, second_str);
    // Сравнение строк
    printf("Compare result: %d\n", strcmp(first_str, second_str));

    printf("Enter another string: ");
    fscanf(stdin, "%s", my_string);
    // В нижний регистр
    for (int i = 0; my_string[i]; i++)
        my_string[i] = tolower(my_string[i]);
    printf("%s\n", my_string);
    // В верхний регистр
    for (int i = 0; my_string[i]; i++)
        my_string[i] = toupper(my_string[i]);
    printf("%s\n", my_string);

    return 0;
}
```

Листинг 5: Lab4 - 2-1.c

Результат выполненной работы

```
Enter string: string
String size: 5
Compare result: 4
Enter another string: string
string
STRING
```

2.2. 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

Список идентификаторов

Имя переменной	*Тип данных*	*Смысловое значение*
input_str	char[]	Вводимая строка
double_input	Double	Вещественное значение
integer_input	Integer	Целое значение

Код программы

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

int main(void)
{
   printf("Enter integer: ");
   char input_str[20];
   fscanf(stdin, "%s", input_str);

   double double_input = atof(input_str);
   int integer_input = atoi(input_str);

   printf("Numbers are: %.2lf, %d\n", double_input, integer_input);

   return 0;
}
```

Листинг 6: Lab4 - 2-2.c

Результат выполненной работы

```
Enter integer: 12.2
Numbers are: 12.20, 12
```

2.3. 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 4 каждый символ (своими словами, в свободной форме). Постарайтесь использовать только латиницу.

Список идентификаторов

Имя переменной	*Тип данных*	*Смысловое значение*
str	char[]	Изначальная строка

Код программы

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

int main(void)
{
   char str[] = "~`1!@#$^&*()_Qq{}:\",./?";

   for (int i = 0; str[i]; i++)
   {
       printf("%c\n", str[i]);
       printf("isalnum  %d\n", isalnum(str[i]));
       printf("isalpha  %d\n", isalpha(str[i]));
       printf("islower  %d\n", islower(str[i]));
       printf("isupper  %d\n", isupper(str[i]));
       printf("isdigit  %d\n", isdigit(str[i]));
       printf("isxdigit %d\n", isxdigit(str[i]));
       printf("iscntrl  %d\n", iscntrl(str[i]));
       printf("isgraph  %d\n", isgraph(str[i]));
       printf("isspace  %d\n", isspace(str[i]));
       printf("isblank  %d\n", isblank(str[i]));
       printf("isprint  %d\n", isprint(str[i]));
       printf("ispunct  %d\n", ispunct(str[i]));
   }
   return 0;
}
```

Листинг 7: Lab4 - 2-3.c

Результат выполненной работы

```
?
isalnum  0
isalpha  0
islower  0
isupper  0
isdigit  0
isxdigit 0
iscntrl  0
isgraph  32768
isspace  0
isblank  0
isprint  16384
ispunct  4
```

```
~
isalnum  0
isalpha  0
islower  0
isupper  0
isdigit  0
isxdigit 0
iscntrl  0
isgraph  32768
isspace  0
isblank  0
isprint  16384
ispunct  4
```

```
q
isalnum  8
isalpha  1024
islower  512
isupper  0
isdigit  0
isxdigit 0
iscntrl  0
isgraph  32768
isspace  0
isblank  0
isprint  16384
ispunct  0
```