

Лабораторная работа 10.

Процедуры.

1. Создать некоторую структуру с указателем на некоторую функцию в качестве поля. Вызвать эту функцию через имя переменной этой структуры и поле указателя на функцию.

Имя	Значение	Тип
call	Переменная структуры	myStruct
func	Указатель на функцию в структуре	int

Код программы:

```
#include <stdio.h>

int function() {
    printf("return num: ");
    return 15;
}

typedef struct fStruct {
    int (*func)();
}myStruct;

int main(){
    myStruct call;
    call.func = &function;
    printf("%d", call.func());
    return 0;
}
```

Результат выполненной работы:

```
C:\Users\grish\OneDrive\Рабочий стол\лабы по си\лаба 10>task1
return num: 15
```

2. Создать структуру для вектора в 3-х мерном пространстве. Реализовать и использовать в своей программе следующие операции над векторами:

- скалярное умножение векторов;
- векторное произведение;
- модуль вектора;
- распечатка вектора в консоли.

В структуре вектора указать имя вектора в качестве отдельного поля этой

структуры.

Математическая модель:

$$\vec{a} \cdot \vec{b} = x_1 x_2 + y_1 y_2 + z_1 z_2$$

скалярное произведение

векторное произведение

$$\vec{a} \times \vec{b} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{vmatrix} = \vec{i} \begin{vmatrix} y_1 & z_1 \\ y_2 & z_2 \end{vmatrix} - \vec{j} \begin{vmatrix} x_1 & z_1 \\ x_2 & z_2 \end{vmatrix} + \vec{k} \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} =$$
$$= (y_1 z_2 - y_2 z_1) \vec{i} - (x_1 z_2 - x_2 z_1) \vec{j} + (x_1 y_2 - x_2 y_1) \vec{k}$$

$$|\vec{a}| = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

модуль вектор

Список идентификаторов:

Имя	Значение	Тип
name	Имя вектора, поле структуры	char
x	Координата x, поле структуры	int
y	Координата y, поле структуры	int
z	Координата z, поле структуры	int
vec1	Вектор а	vector
vec2	Вектор б	vector
vec3	Вектор - результат произведения	vector
vec	Вектор принимаемый в функцию	vector

Код программы:

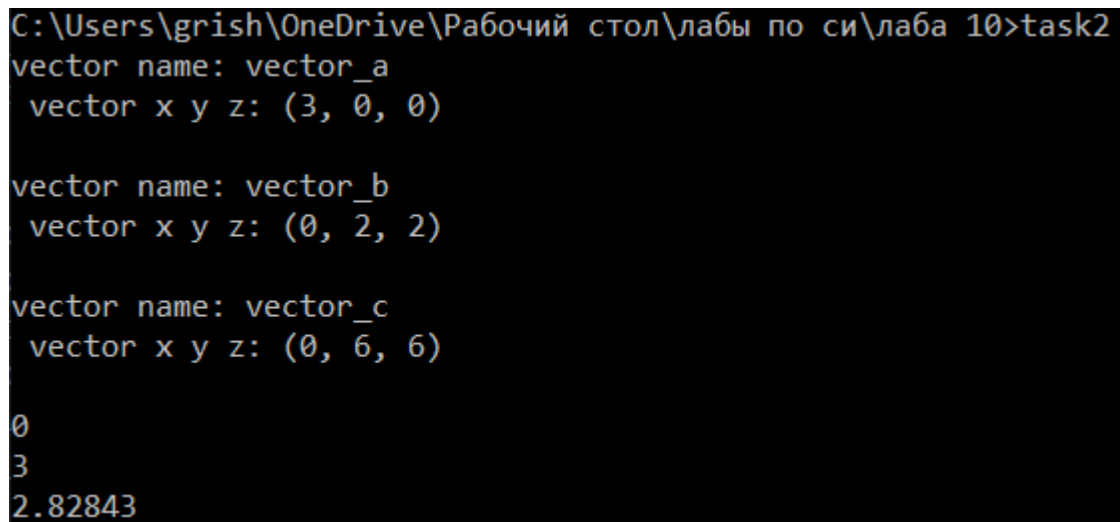
```
#include <stdio.h>
#include <math.h>
typedef struct vectorStruct{
    char name[10];
    float x;
    float y;
    float z;
}vector;
void show_vec(vector *vec) {
    printf("vector name: %s \n vector x y z: (%g, %g, %g)\n\n", vec->name, vec->x,
vec->y, vec->z);
}
```

```

float abs_vec(vector* vec) {
    float absVector;
    absVector = vec->x*vec->x + vec->y * vec->y + vec->z * vec->z;
    absVector = sqrt(absVector);
    return absVector;
}
float scal_vec(vector* vec1, vector* vec2) {
    float scal;
    scal = vec1->x * vec2->x + vec1->y * vec2->y + vec1->z * vec2->z;
    return scal;
}
vector prod_vec(vector* vec1, vector* vec2) {
    vector vec3 = {"vector_c"};
    vec3.x = vec1->y * vec2->z - vec2->y * vec1->z;
    vec3.y = vec1->x * vec2->z - vec2->x * vec1->z;
    vec3.z = vec1->x * vec2->y - vec2->x * vec1->y;
    return vec3;
}
int main(){
    vector vec1 = { "vector_a" , 3, 0, 0 },
        vec2 = { "vector_b" , 0, 2, 2 },
        vec3;
    show_vec(&vec1);
    show_vec(&vec2);
    vec3 = prod_vec(&vec1, &vec2);
    show_vec(&vec3);
    printf("%g\n", scal_vec(&vec1, &vec2));
    printf("%g\n", abs_vec(&vec1));
    printf("%g\n", abs_vec(&vec2));
    return 0;
}

```

Результат выполненной работы:



```

C:\Users\grish\OneDrive\Рабочий стол\лабы по си\лаба 10>task2
vector name: vector_a
vector x y z: (3, 0, 0)

vector name: vector_b
vector x y z: (0, 2, 2)

vector name: vector_c
vector x y z: (0, 6, 6)

0
3
2.82843

```

3. Вычислить, используя структуру комплексного числа, комплексную экспоненту $\exp(z)$ некоторого $z \in \mathbb{C}$.

$$e^z = 1 + \sum_{n=1}^{\infty} \frac{1}{n!} z^n .$$

Математическая модель:
число.

Где z комплексное

Список идентификаторов:

Имя	Значение	Тип
real	Поле структуры complex , действительная часть	float
img	Поле структуры complex, мнимая часть	float
x	Комплексное число	complex
exp	Комплексная экспонента	complex
z	Передаваемое в функцию комплексное число	complex
i	Счетчик цикла	int
zPow	Комплексное число в степени	complex
ar	Действительная часть комплексного числа	float
ai	Мнимая часть комплексного числа	float
expri	Мнимая часть комплексной экспоненты	float
expr	Действительная часть комплексной экспоненты	float
k	Факториал какого числа нужно получить	int
c	Факториал числа k	float
pow	Степень, в которую нужно возвести комплексное число	int

Код программы:

```
#include <stdio.h>
typedef struct complexnum {
    float real;
    float img;
}complex;

complex complex_pow(complex *z, int pow) {
    complex a = *z;
    float ar, ai, i;
    for (i = 1; i < pow; i++) {
        ar = a.real * z->real - a.img * z->img;
        ai = a.real * z->img + a.img * z->real;
        a.img = ai;
        a.real = ar;
    }
    return a;
}

int fact(int k) {
    int i;
    float c = 1;
```

```

        for (i = 1; i <= k; i++)
            c *= i;
        return c;
    }
    complex complex_exp(complex *z) {
        complex zPow;
        complex exp = {1, 0};
        float expr, expi;
        for (int i = 1; i < 12; i++) {
            zPow = complex_pow(z, i);
            exp.real += zPow.real / fact(i);
            exp.img += zPow.img / fact(i);
        }
        return exp;
    }
    int main(){
        complex x = {2, -2};
        printf("x = %g%gi\n", x.real, x.img);
        x = complex_exp(&x);
        printf("exp(x) = %g%gi", x.real, x.img);
        return 0;
    }

```

Результат выполненной работы:

```

C:\Users\grish\OneDrive\Рабочий стол\лабы по си\лаба 10>task3
x = 2-2i
exp(x) = -3.0743-6.71896i

```

+ проверка

Decimal approximation:

```

- 3.0749323206393588671124790547961599481102388620801760611... -
6.7188496974282499712683027713652021236645445949895632970... i

```

4. Реализовать в виде структур двунаправленный связный список и совершить отдельно его обход в прямом и обратном направлениях с распечаткой значений каждого элемента списка.

Список идентификаторов:

Имя	Значение	Тип
value	Поле структуры хранимое значение	int
next	Указатель на следующий экземпляр структуры	List
previous	Указатель на предидущий экземпляр структуры	List
size	Размер списка	int
begin	Указатель на начало списка	list
ptrOnStruct	Указатель на текущий экземпляр структуры	list
ptrOnStructNext	Указатель на следующий экземпляр структуры	list
ptrOnStructLast	Указатель на текущий и крайний экземпляр структуры	list

start	Ссылка на первый элемент списка	list
i	Счетчик цикла	int

Код программы:

```
#include <stdio.h>
#include <stdlib.h>
typedef struct List {
    int value;
    struct List *next;
    struct List *previous;
}list;
list* create_list(int size) {
    list* begin = malloc(sizeof(list));
    begin->value = 0;
    begin->previous = NULL;
    list *ptrOnStruct, *ptrOnStructNext;
    ptrOnStruct = begin;
    for (int i = 1; i < size; i++) {
        ptrOnStructNext = malloc(sizeof(list));
        ptrOnStructNext->value = ptrOnStruct->value + 1;
        ptrOnStruct->next = ptrOnStructNext;
        ptrOnStructNext->previous = ptrOnStruct;
        ptrOnStruct = ptrOnStructNext;
    }
    ptrOnStruct->next = NULL;
    return begin;
}
void show_list(list* begin) {
    list *ptrOnStruct = begin;
    list* ptrOnStructLast;
    while (ptrOnStruct != NULL) {
        printf("%d", ptrOnStruct->value);
        ptrOnStructLast = ptrOnStruct;
        ptrOnStruct = ptrOnStruct->next;
    }
    printf("\n");
    while (ptrOnStructLast != NULL) {
        printf("%d", ptrOnStructLast->value);
        ptrOnStructLast = ptrOnStructLast->previous;
    }
}
void delete_list(list *begin) {
    list *ptrOnStructNext = begin->next;
    if (ptrOnStructNext != NULL) {
        delete_list(ptrOnStructNext);
    }
    free(begin);
}
int main(){
    list* start = create_list(10);
    show_list(start);
    delete_list(start);
    return 0;
}
```

Результат выполненной работы:

```
C:\Users\grish\OneDrive\Рабочий стол\лабы по си\лаба 10>task4  
0123456789  
9876543210
```