Лабораторная работа №3

Программирование. Язык СИ.

Структуры. Объединения. Перечисления.

Комплект 1: Структуры

Задание 1.1

Задача:

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

Список идентификаторов:

Имя	Смысл	Тип
functionExp	Функция в структуре	struct structExp
*expon	Указатель на функцию	double

Код программы:

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

struct structExp
{
    double (*expon) (double);
};

int main()
{
    struct structExp functionExp;
    functionExp.expon = exp;
    printf("%.2lf", functionExp.expon(8));

    return 0;
}
```

Результат выполнения программы:

```
2980.96
Process finished with exit code
```

Задание 1.2

Задача:

1.2: Создать структуру для вектора в 3-х мерном пространстве. Реализовать и испольхзовать 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.

Список идентификаторов:

Имя	Смысл	Тип
v1	Первый вектор	struct vector
v2	Второй вектор	struct vector

Код программы:

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

struct vector
{
    char name;
    int x;
    int y;
    int z;
};

int main()
{
    struct vector v1 = {'i',7, 4, -5};
    struct vector v2 = {'l',-8, -3, 2};
    printf("Scalar multiplication: %c{%d; %d; %d} * %c{%d; %d; %d} = %d\n",
v1.name, v1.x, v1.y, v1.z, v2.name, v2.x, v2.y, v2.z, v1.x * v2.x + v1.y *
v2.y + v1.z * v2.z);
    printf("Vector product: %c{%d; %d; %d} * %c{%d; %d; %d} = {%d; %d;
%d}\n", v1.name, v1.x, v1.y, v1.z, v2.name, v2.x, v2.y, v2.z, (v1.y * v2.z -
v1.z * v2.y), - (v1.z * v2.x - v1.x * v2.z), (v1.x * v2.y - v1.y * v2.x));
    printf("Vector modulus 1: |%c{%d; %d; %d}| = %.3lf\n", v1.name, v1.x,
v1.y, v1.z, sqrt((v1.x * v1.x + v1.y * v1.y + v1.z * v1.z)));
    printf("Vector modulus 2: |%c{%d; %d; %d}| = %.3lf\n", v2.name, v2.x,
v2.y, v2.z, sqrt((v2.x * v2.x + v2.y * v2.y + v2.z * v2.z)));

    return 0;
}
```

Результат выполнения программы:

```
Scalar multiplication: i{7; 4; -5} * l{-8; -3; 2} = -78
Vector product: i{7; 4; -5} * l{-8; -3; 2} = {-7; -26; 11}
Vector modulus 1: |i{7; 4; -5}| = 9.487
Vector modulus 2: |l{-8; -3; 2}| = 8.775

Process finished with exit code 0
```

Задание 1.3

Задача:

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

$$exp(z) = 1 + z + \frac{1}{2!}z^2 + \frac{1}{3!}z^3 + \ldots + \frac{1}{n!}z^n. \quad (1)$$

Список идентификаторов:

Имя	Смысл	Тип
b	Вспомогательная переменная	double
buffer	Вспомогательная переменная	double
n	Точность вычисления	int
i	Параметр цикла	int
z	Комплексное число	struct complex
pow	Подсчёт степени в структуре	struct complex
exp	Вычисление значения в структуре	struct complex

Код программы:

```c
#include <stdio.h>

struct complex
{
    double real;
    double imaginary;
};
int main()
{
    double b = 1;
    double buffer = 1;
    int n = 20;
    struct complex z;
    printf("Enter real part: ");
    scanf("%lf", &z.real);
    printf("Enter imaginary part:");
    scanf("%lf", &z.imaginary);
    struct complex exp = {1 + z.real, z.imaginary};
    struct complex pow = {z.real, z.imaginary};
    for (int i = 2; i <= n; ++i)
    {
        b=b*i;
        buffer = pow.real;
        pow.real = pow.real * z.real - pow.imaginary * z.imaginary;
        pow.imaginary = buffer * z.imaginary + pow.imaginary * z.real;
        exp.real += pow.real/b;
        exp.imaginary += pow.imaginary/b;
    }
    printf("exp(z) = %lf + (%lf)i\n", exp.real, exp.imaginary);

    return 0;
}
```

Результат выполнения программы:

```
Enter real part:8
 Enter imaginary part:-3
exp(z) = -2951.258489 + (-419.583982)i

Process finished with exit code 0
```

Задание 1.4

Задача:

1.4: Используя так называемые "битовые" поля в структуре С, создать экономную структуру в оперативной памяти для заполнения даты некоторого события, например даты рождения человека. Ссылки на описание битовых полей:

- https://en.cppreference.com/w/cpp/language/bit_field;
- https://en.wikipedia.org/wiki/Bit_field.

Список идентификаторов:

Имя	Смысл	Тип
date	Дата рождения	struct birthday

Код программы:

```c
#include <stdio.h>

struct birthday
{
    unsigned int d: 5;
    unsigned int m: 4;
    unsigned int y: 11;
};
int main()
{
    struct birthday date = {3, 3, 2003};
    printf("Birthday: %d.%d.%d", date.d, date.m, date.y);

    return 0;
}
```

Результат выполнения программы:

```
Birthday: 3.3.2003
Process finished with exit code
```

Задание 1.5

Задача:

1.5: 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	Смысл	Тип
n	Массив структур	struct list
i	Параметр цикла	int

Код программы:

```c
#include <stdio.h>

struct list
{
    int number;
    struct list *next;
    struct list *prev;
};

int main()
{
    struct list n[6];
    n[0].prev = NULL;
    n[0].number = 1;
    n[0].next = &n[1];

    n[5].prev = &n[4];
    n[5].number =999;
    n[5].next = NULL;

    for(int i =1; i<5; i++){
        n[i].prev = &n[i-1];
        n[i].number = i*10;
        n[i].next = &n[i+1];
    };
    struct list *pointer;
    pointer = &n[0];
    do
    {
        printf("%d ", pointer->number);
        pointer = pointer->next;
    }
    while(pointer != NULL);
    pointer = &n[5];
    do
    {
        printf("%d ", pointer->number);
        pointer = pointer->prev;
    }
    while(pointer != NULL);

    return 0;
}
```

Результат выполнения программы:

```
1 10 20 30 40 999 999 40 30 20 10 1
Process finished with exit code 0
```

Комплект 2: Объединения и перечисления

Задание 2.1

Задача:

2.1: Напишите программу, которая использует указатель на некоторое объединение `union`.

Список идентификаторов:

Имя	Смысл	Тип
num	Экземпляр объединения unTest, который содержит double	union unTest
pointer	Указатель на структуру	union unTest *

Код программы:

```c
#include <stdio.h>

union unTest
{
    double num1;
    int num2;
};
int main()
{
    union unTest num = {-60785.564125};
    union unTest *pointer = &num;
    printf("%lf %d", pointer->num1, pointer->num2);

    return 0;
}
```

Результат выполнения программы:

```
-60785.564125 223338299
Process finished with exit code 0
```

Задание 2.2

Задача:

2.2: Напишите программу, которая использует **union** для побайтовой распечатки типа **unsigned long**.

Список идентификаторов:

Имя	Смысл	Тип
num	Экземпляр объединения unLong, который содержит unsigned long	union unLong
pointer	Указатель на структуру	union unLong
i	Параметр цикла	int

Код программы:

```c
#include <stdio.h>

union unLong
{
    unsigned long num1;
    char num2[4];
};
int main()
{
    union unLong num = {953036924};
    union unLong *pointer = &num;
    for (int i = 0; i < 4; ++i)
    {
        printf("%d ", *(pointer->num2 + i));
    }

    return 0;
}
```

Результат выполнения программы:

```
124 48 -50 56
Process finished with exit code 0
```

Задание 2.3

Задача:

2.3: Создайте перечислимый тип данных (**enum**) для семи дней недели и распечатайте на экране его значения, как целые числа.

Список идентификаторов:

Код программы:

```c
#include <stdio.h>

enum week
{
    monday = 1,
    tuesday = 2,
    wednesday = 3,
    thursday = 4,
    friday = 5,
    saturday = 6,
    sunday = 7
};
int main()
{
    printf("%d ", monday);
    printf("%d ", tuesday);
    printf("%d ", wednesday);
    printf("%d ", thursday);
    printf("%d ", friday);
    printf("%d ", saturday);
    printf("%d ", sunday);

    return 0;
}
```

Результат выполнения программы:

```
1 2 3 4 5 6 7

Process finished with exit code 0
```