Sample textВеличко Арсений Александрович
ИВТ 1 курс, 2 группа, 3 подгруппа
Предмет: Программирование

# Лабораторная работа №4

## Задание 4.1

### Постановка задачи

4.1: Внутри функции `int main(void) { /*... */}` определите указатель `double **pointer = NULL;`. В оперативной памяти создайте конструкцию, показанную на рисунке 1.

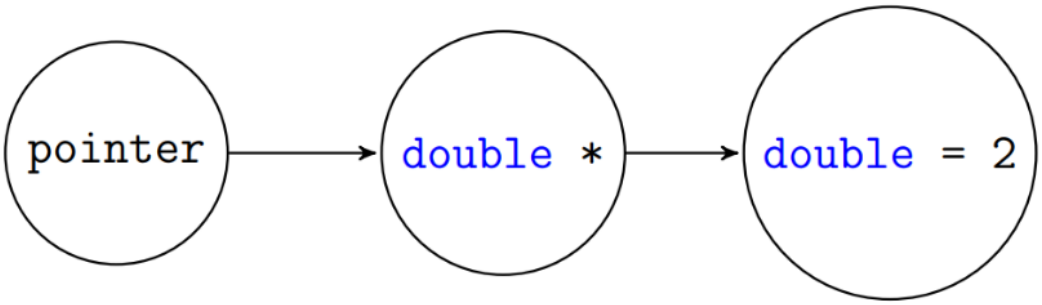


Рис. 1: Последовательность из двух указателей, ведущих к значению.

При этом выполните следующее:

- используйте функции типа `*alloc(...)` для выделения оперативной памяти под динамические объекты;
- выведите число, указанное в крайней правой окружности, на экран, используя указатель `double **pointer = NULL;`;
- используйте функцию `free(...)` для освобождения оперативной памяти, выделенную под динамические объекты.

### Описание переменных

| Переменная | Тип | Суть |
|---|---|---|
| a | double | Переменная A |
| b | double* | Указатель на a |
| ptr | double** | Указатель на указатель на a |

### Код программы

```c
#include <stdio.h>

int main(void) {
    double **ptr = NULL;
    double a = 2.0;
    double *b = &a;
    ptr = &b;
    printf("%f", **ptr);
    return 0;
}
```

### Вывод программы

```
2.000000
```

## Задание 4.2

### Постановка задачи

4.2: Напишите программу, которая складывает два числа с использованием указателей на эти числа.

### Описание переменных

| Переменная | Тип | Суть |
|---|---|---|
| a | int | Слагаемое a |
| b | int | Слагаемое b |
| p1 | int* | Указатель на слагаемое a |
| p2 | int* | Указатель на слагаемое b |
| sum | int | Сумма |

### Код программы

```c
#include <stdio.h>

int main() {
    int a = 2, b = 3;
    printf("a = %d; b = %d \n", a, b);
    int *p1 = &a, *p2 = &b;
    int sum = *p1 + *p2;
    printf("sum = %d", sum);
    return 0;
}
```

### Вывод программы

```
a = 2; b = 3
sum = 5
```

## Задание 4.3

### Постановка задачи

4.3: Напишите программу, которая находит максимальное число из двух чисел, используя указатели на эти числа.

### Описание переменных

| Переменная | Тип | Суть |
|---|---|---|
| a | int | число a |
| b | int | число b |
| p1 | int* | Указатель на число a |
| p2 | int* | Указатель на число b |

## Код программы

```c
#include <stdio.h>

int main() {
    int a, b;
    int *p1 = &a, *p2 = &b;

    printf("A, B: ");
    scanf("%d %d", &a, &b);

    if (*p1 > *p2) {
        printf("A > B");
    }
    else if (*p1 < *p2) {
        printf("A < B");
    }
    else {
        printf("A = B");
    }
    return 0;
}
```

## Вывод программы

```
A, B: 10 20
A < B
```

`Sample text`Величко Арсений Александрович
ИВТ 1 курс, 2 группа, 3 подгруппа
Предмет: Программирование

# Задание 4.4

## Постановка задачи

4.4: Напишите программу, которая создаёт одномерный динамический массив из чисел с плавающей точкой двойной точности, заполняет его значениями с клавиатуры и распечатывает все элементы этого массива, используя арифметику указателей (оператор +), а не обычный оператор доступа к элементу массива — [ ].

## Описание переменных

| Переменная | Тип | Суть |
|---|---|---|
| n | int | Длина массива |
| A | double* | Область памяти на n эл-тов типа double |
| i | double* | Позиция итератора |

## Код программы

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

int main() {
    int n;
    printf("Длина массива: ");
    scanf("%d", &n);
    double *A = calloc(n, sizeof(double));
    double *i = A;

    printf("Значения эл-тов \n");
    while (i < A + n) {
        scanf("%lf", i);
        i++;
    }

    i = A;
    printf("Массив: ");
    while (i < A + n) {
        printf("%lf ", *i);
        i++;
    }

    free(A);
    return 0;
}

```

## Вывод программы

```
Длина массива: 3
Значения эл-тов
1
2
3
Массив: 1.000000 2.000000 3.000000
```

Sample textВеличко Арсений Александрович
ИВТ 1 курс, 2 группа, 3 подгруппа
Предмет: Программирование

# Задание 4.5

## Постановка задачи

4.5: Вывести элементы динамического массива целых чисел в обратном порядке, используя указатель и операцию декремента (--).

## Описание переменных

| Переменная | Тип | Суть |
|---|---|---|
| n | int | Длина массива |
| A | int* | Область памяти на n эл-тов типа int |
| P | int* | Позиция итератора |

## Код программы

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

int main(void) {
    srand(time(NULL));

    int n;
    printf("Длина массива: ");
    scanf("%d", &n);
    int *A = calloc(n, sizeof(int));
    int *p = A;

    printf("Массив: ");
    while (p < A + n) {
        *p = rand() % n;
        printf("%d ", *p);
        p++;
    }
    p--;
    printf("\n");

    printf("Массив в обратном порядке: ");
    while (p >= A) {
        printf("%d ", *p);
        p--;
    }

    return 0;
}

```

## Вывод программы

```
Длина массива: 7
Массив: 6 4 2 3 6 3 0
Массив в обратном порядке: 0 3 6 3 2 4 6
```

Sample textВеличко Арсений Александрович
ИВТ 1 курс, 2 группа, 3 подгруппа
Предмет: Программирование

# Задание 4.6

## Постановка задачи

4.6: Отсортируйте заданный массив целых чисел, используя указатели, а не доступ по индексу ([ ]).

## Описание переменных

| Переменная | Тип | Суть |
|---|---|---|
| n | int | Длина массива |
| t | int | Буфер обмена при перестановках |
| A | Arr of int | Массив целых чисел |
| i | *int | Указатель на эл-т массива A |
| j | *int | Указатель на эл-т массива A |

## Код программы

```c
#include <stdio.h>

int main(void) {
#define n 10
    int t;
    int A[n] = {1, 12, 9, 23, 55, 87, 11, 8, 4, 33};
    int *i = A;
    int *j = A;

    printf("Исходный массив: ");
    while (i < A + n) {
        printf("%d ", *i);
        i++;
    }
    i = A;
    printf("\n");

    while (i < A + n) {
        j = i;
        while (j < A + n) {
            if (*i > *j) {
                t = *i;
                *i = *j;
                *j = t;
            }
            j++;
        }
        i++;
    }
    i = A;

    printf("Отстортированный массив: ");
    while (i < A + n) {
        printf("%d ", *i);
        i++;
    }

    return 0;
}
```

## Вывод программы

```
Исходный массив: 1 12 9 23 55 87 11 8 4 33
Отстортированный массив: 1 4 8 9 11 12 23 33 55 87
```

Sample textВеличко Арсений Александрович
ИВТ 1 курс, 2 группа, 3 подгруппа
Предмет: Программирование

# Задание 4.7

## Постановка задачи

4.7: Определите переменную целого типа `int a = 1234567890;` и выведите побайтово её содержимое на экран, используя указатель `char *`.

## Описание переменных

| Переменная | Тип | Суть |
|---|---|---|
| a | int | Целое число a |
| p | char* | Указатель на часть a |

## Код программы

```c
#include <stdio.h>

int main(void) {
    int a = 1234567890;
    char *p = &a;

    while (p < &a + 1) {
        printf("%d ", *p);
        p++;
    }

    return 0;
}
```

## Вывод программы

```
-46 2 -106 73
```

`Sample text`Величко Арсений Александрович
ИВТ 1 курс, 2 группа, 3 подгруппа
Предмет: Программирование

# Задание 4.8

## Постановка задачи

4.8: Выделите память под двумерный динамический массив — матрицу — таким образом, чтобы данные все строки этой матрицы гарантированно располагались в оперативной памяти друг за другом (C 2D array contiguous memory allocation).

## Описание переменных

| Переменная | Тип | Суть |
|---|---|---|
| n | int | Сторона матрицы |
| i | int | Счетчик цикла |
| j | int | Счетчик цикла |
| A | int** | Двумерная матрица |
| t | int* | Счетчик строк |

## Код программы

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

int main(void) {

    int n, i, j;

    printf("Сторона матрицы: ");
    scanf("%d", &n);

    int **A = malloc(n * sizeof(int));
    int *t = malloc(n * n * sizeof(int));
    for (i = 0; i < n; i++) {
        A[i] = t + i * n;
    }

    printf("Адреса эл-тов в памяти:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            printf("%d ", &A[i][j]);
        }
        printf("\n");
    }

    free(A);
    free(t);
    return 0;
}
```

## Вывод программы

```
Сторона матрицы: 5
Адреса эл-тов в памяти:
616831712 616831716 616831720 616831724 616831728
616831732 616831736 616831740 616831744 616831748
616831752 616831756 616831760 616831764 616831768
616831772 616831776 616831780 616831784 616831788
616831792 616831796 616831800 616831804 616831808
```

`Sample text`Величко Арсений Александрович
ИВТ 1 курс, 2 группа, 3 подгруппа
Предмет: Программирование

# Задание 5.1

## Постановка задачи

5.1: 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.

## Описание переменных

| Переменная | Тип | Суть |
|---|---|---|
| a | int | Основание факториала |
| factorial | int | Значение факториала |

## Код программы

```c
#include <stdio.h>

void iterational_factorial(int a);
int recursive_factorial(int a);

int main() {
    int a;
    printf("Число: ");
    scanf("%d", & a);
    iterational_factorial(a);
    printf("Рекурсивный факториал = %d", recursive_factorial(a));

    return 0;
}

void iterational_factorial(int a) {
    int factorial = 1;

    while (a > 0) {
        factorial *= a;
        --a;
    }

    printf("Циклический факториал = %d \n", factorial);
}

int recursive_factorial(int a) {
    if (a <= 1)
        return 1;
    else
        return a * recursive_factorial(a - 1);
}

```

### Вывод программы

```
Число: 3
Циклический факториал = 6
Рекурсивный факториал = 6
```

## Задание 5.2

### Постановка задачи

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

### Описание переменных

| Переменная | Тип | Суть |
|---|---|---|
| A | Int * | Массив цел. чисел |
| i | int | Итератор |
| n | int | Длина массива A |
| t | int | Буфер обмена |

## Код программы

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

void change_places(int * A, int n);

int main() {
    srand(time(NULL));
    int n = 12, i;
    int * A = calloc(n, sizeof(int));
    printf("Исх. массив: ");

    for (i = 0; i < n; i++) {
        A[i] = rand() % 20 - 10;
        printf("%d ", A[i]);
    }

    printf("\n");
    change_places(A, n);

    printf("Новый массив: ");
    for (i = 0; i < n; i++) {
        printf("%d ", A[i]);
    }
    printf("\n");

    free(A);
    scanf("%s");
    return 0;
}

void change_places(int * A, int n) {
    int t;

    for (int i = 0; i < n - 1; i++) {
        if (i % 2 == 0) {
            t = A[i];
            A[i] = A[i + 1];
            A[i + 1] = t;
        }
    }
}
```

## Вывод программы

```
Исх. массив: -3 1 2 -5 -10 4 -3 -6 -3 4 7 6
Новый массив: 1 -3 -5 2 4 -10 -6 -3 4 -3 6 7
```

Sample textВеличко Арсений Александрович
ИВТ 1 курс, 2 группа, 3 подгруппа
Предмет: Программирование

# Задание 5.3

## Постановка задачи

5.3: 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.

## Описание переменных

| Переменная | Тип | Суть |
|---|---|---|
| A | double** | Двумерный массив чисел double |
| i | int | Размерность массива |
| j | int | |
| m | int | |
| n | int | |

## Код программы

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

double ** generate_an_array(int m, int n);
void free_an_array(double ** A);
void fill_an_array(double ** A, int m, int n);
void output_an_array(double ** A, int m, int n);

int main() {
    int m, n;
    double ** A;

    printf("Размерность массива (MxN): ");
    scanf("%d%d", & m, & n);
    A = generate_an_array(m, n);
    fill_an_array(A, m, n);
    output_an_array(A, m, n);
    free_an_array(A);
    scanf("%s");

    return 0;
}

double ** generate_an_array(int m, int n) {
    double ** A = calloc(m, sizeof(double));
    for (int i = 0; i < m; i++) {
        A[i] = calloc(n, sizeof(double));
    }

    return A;
}

void free_an_array(double ** A) {
    free(A);
}

void fill_an_array(double ** A, int m, int n) {
    srand(time(NULL));
```

```
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            A[i][j] = rand() % 999999;
            A[i][j] /= 10000;
        }
    }
}

void output_an_array(double ** A, int m, int n) {
    printf("Выходной массив:\n");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            printf("%lf ", A[i][j]);
        }
        printf("\n");
    }
}
```

## Вывод программы

```
Размерность массива (MxN): 5 6
Выходной массив:
5.831900 6.075600 0.701000 90.140800 38.573200 64.839100
4.942700 74.612700 41.939600 87.603700 31.943000 6.769600
58.406800 91.325800 16.019800 14.023700 35.891600 3.397100
61.904000 48.855300 39.054300 16.055100 25.119300 85.392900
75.784800 48.721200 80.907100 35.233200 97.833200 96.284700
```

`Sample text`Величко Арсений Александрович
ИВТ 1 курс, 2 группа, 3 подгруппа
Предмет: Программирование

# Задание 5.4

## Постановка задачи

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

## Описание переменных

| Переменная | Тип | Суть |
|---|---|---|
| A | int* | Массив типа int |
| B | int* | |
| C | int* | |
| pointer_a | int* | Хранит указатель соотв. массива |
| pointer_b | int* | |
| pointer_c | int* | |
| address_A | int* | Хранит текущий адрес соотв. массива |
| address_B | int* | |
| address_C | int* | |

## Код программы

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

void output_vector(int * pointer_a, int * pointer_b);
void multiply(int * A, int * B, int * C);

int main() {
    int A[3] = {
        3,
        4,
        5
    };
    int B[3] = {
        4,
        5,
        6
    };
    int C[3];
    int * pointer_a, * pointer_b, * pointer_c;

    pointer_a = A;
    pointer_b = B;
    pointer_c = C;

    output_vector(pointer_a, pointer_b);

    multiply(pointer_a, pointer_b, pointer_c);

    scanf("%d");

    return 0;
}

void output_vector(int * A, int * B) {
    int * adress_A = A;
    int * adress_B = B;

    printf("Вектор A = ");
    while (A < adress_A + 3) {
```

```
        printf("[%d] ", * A);
        A++;
    }
    printf("\n");

    printf("Вектор B = ");
    while (B < adress_B + 3) {
        printf("[%d] ", * B);
        B++;
    }
    printf("\n");
}

void multiply(int * A, int * B, int * C) {
    int * adress_C;
    adress_C = C;

    * C = ( * (A + 1) * * (B + 2) - * (A + 2) * * (B + 1));
    *(C + 1) = ( * (A + 2) * * (B) - * (A) * * (B + 2));
    *(C + 2) = ( * (A) * * (B + 1) - * (A + 1) * * (B));

    printf("Произведение векторов: ");
    while (C < adress_C + 3) {
        printf("[%d] ", * C);
        C++;
    }
}
```

## Вывод программы

```
Вектор A = [3] [4] [5]
Вектор B = [4] [5] [6]
Произведение векторов: [-1] [2] [-1]
```