Семинар #7: Повторение. Классные задачи.

Основы

- На вход подаются 2 целых числа `a` и `b`. Напечатайте сначала `b`, а потом `a` через пробел.

```c
#include <stdio.h>
int main() {
    int a, b;
    scanf("%i%i", &a, &b);
    printf("%i %i\n", b, a);
}
```

- На вход подаются 2 целых числа `a` и `b`. Напечатайте остаток деления первого числа на второе.

```c
#include <stdio.h>
int main() {
    int a, b;
    scanf("%i%i", &a, &b);
    printf("%i\n", a % b);
}
```

- На вход подаются 3 целых числа. Напечатайте `Yes`, если третье число является суммой двух первых.

```c
#include <stdio.h>
int main() {
    int a, b, c;
    scanf("%i%i%i", &a, &b, &c);
    if (a + b == c) {
        printf("Yes\n");
    }
    else {
        printf("No\n");
    }
}
```

- На вход подаются 2 целых числа `a` и `b`. Напечатайте наибольшее из этих чисел.

```c
#include <stdio.h>
int main() {
    int a, b;
    scanf("%i%i", &a, &b);
    if (a > b) {
        printf("%i\n", a);
    }
    else {
        printf("%i\n", b);
    }
}
```

- На вход подаются 2 целых числа `a` и `b`. Напечатайте все числа от наибольшего из этих чисел до наименьшего.

вход	выход
2 8	8 7 6 5 4 3 2
9 6	9 8 7 6

```c
#include <stdio.h>
int main() {
    int a, b;
    scanf("%i%i", &a, &b);
    int max, min;
    if (a > b) {
        max = a;
        min = b;
    }
    else {
        max = b;
        min = a;
    }

    for (int i = max; i >= min; --i) {
        printf("%i ", i);
    }
}
```

- На вход поступает число `n` и, затем, `n` целых чисел. Напечатайте сумму этих `n` чисел.

вход	выход
3 7 3 1	11

```c
#include <stdio.h>
int main() {
    int n;
    scanf("%i", &n);

    int sum = 0;
    for (int i = 0; i < n; ++i) {
        int num;
        scanf("%i", &num);
        sum += num;
    }

    printf("%i\n", sum);
}
```

- На вход поступает число `n` и, затем, `n` целых чисел. Напечатайте наибольшее из этих `n` чисел.

вход	выход
4 7 3 8 2	8

```c
#include <stdio.h>
#include <limits.h>
int main() {
    int n;
    scanf("%i", &n);

    int max = INT_MIN;
    for (int i = 0; i < n; ++i) {
        int num;
        scanf("%i", &num);
        if (num > max) {
            max = num;
        }
    }

    printf("%i\n", max);
}
```

- На вход поступает число `n` и, затем, `n` целых чисел. Напечатайте сумму первого и последнего элемента последовательности.

вход	выход
3 7 3 5	12
4 5 8 2 4	9

```c
#include <stdio.h>

int main() {
    int n;
    scanf("%i", &n);
    int sum = 0;

    for (int i = 0; i < n; ++i) {
        int num;
        scanf("%i", &num);
        if (i == 0) {
            sum += num;
        }
        if (i == n - 1) {
            sum += num;
        }
    }

    printf("%i\n", sum);
}
```

Переполнение

- На вход подаётся 1 целое число `a` из диапазона от 0 до $2^{64} - 2$. Напечатайте число, которое на 1 больше.

вход	выход
5	6
123456789123	123456789124

```
#include <stdio.h>

int main() {
    unsigned long long a;
    scanf("%llu", &a);
    printf("%llu\n", a + 1);
}
```

- На вход подаются 2 целых числа из диапазона от 0 до $2^{32} - 1$. Напечатайте их произведение.

вход	выход
2 2	4
123456789 1000000	123456789000000
123456789 123456789	15241578750190521

```
#include <stdio.h>

int main() {
    unsigned long long a, b;
    scanf("%llu%llu", &a, &b);
    printf("%llu\n", a * b);
}
```

Вещественные числа

- На вход подаются 2 вещественных числа. Напечатайте их сумму.
  Используя тип `float`:

```
#include <stdio.h>

int main() {
    float a, b;
    scanf("%f%f", &a, &b);
    printf("%f\n", a + b);
}
```

  Или используя более точный тип `double`:

```
#include <stdio.h>

int main() {
    double a, b;
    scanf("%lf%lf", &a, &b);
    printf("%lf\n", a + b);
}
```

- На вход подаются 2 вещественных числа `x` и `y`. Напечатайте `Yes` если точка `(x, y)` попадает внутрь единичной окружности и `No` иначе.

вход	выход
0.5 -0.5	Yes
0.7 0.7	Yes
0.7 0.8	No

```c
#include <stdio.h>

int main() {
    double x, y;
    scanf("%lf%lf", &x, &y);
    if (x * x + y * y < 1) {
        printf("Yes\n");
    }
    else {
        printf("No\n");
    }
}
```

- На вход подаётся 1 вещественное число a – значение угла в градусах. Напечатайте значение выражения $sin(a) \cdot tan(a)$.

вход	выход
45	0.707
10	0.031
80	5.585

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

int main() {
    double a;
    scanf("%lf", &a);
    const double pi = 3.14159265;
    printf("%lf\n", sin(a * pi / 180) * tan(a * pi / 180));
}
```

Массивы

- На вход поступает число `n` и, затем, `n` целых чисел. Напечатайте эту последовательность 2 раза.

вход	выход
3 7 3 1	7 3 1 7 3 1

```
#include <stdio.h>

int main() {
    int n;
    scanf("%i", &n);
    int array[1000];
    for (int i = 0; i < n; ++i) {
        scanf("%i", &array[i]);
    }

    for (int i = 0; i < n; ++i) {
        printf("%i ", array[i]);
    }
    for (int i = 0; i < n; ++i) {
        printf("%i ", array[i]);
    }
    printf("\n");
}
```

- На вход поступает число `n` и, затем, `n` целых чисел. Напечатайте эту последовательность 2 раза. Первый раз в нормальном порядке, второй раз – в обратном.

вход	выход
3 7 3 1	7 3 1 1 3 7

```
#include <stdio.h>

int main() {
    int n;
    scanf("%i", &n);
    int array[1000];
    for (int i = 0; i < n; ++i) {
        scanf("%i", &array[i]);
    }

    for (int i = 0; i < n; ++i) {
        printf("%i ", array[i]);
    }
    for (int i = 0; i < n; ++i) {
        printf("%i ", array[n - 1 - i]);
    }
    printf("\n");
}
```

- На вход поступает число `n` и, затем, `n` целых чисел. Напечатайте эту последовательность в обратном порядке, повторив каждое число дважды.

вход	выход
3 7 3 1	1 1 3 3 7 7

```c
#include <stdio.h>

int main() {
    int n;
    scanf("%i", &n);
    int array[1000];
    for (int i = 0; i < n; ++i) {
        scanf("%i", &array[i]);
    }

    for (int i = 0; i < n; ++i) {
        printf("%i ", array[n - 1 - i]);
        printf("%i ", array[n - 1 - i]);
    }
    printf("\n");
}
```

- На вход поступает число `n` и, затем, `n` целых чисел. Ещё на вход приходит целое число `k`. Напечатайте эту последовательность в обратном порядке, повторив каждое число `k` раз.

вход	выход
3 7 3 1 4	1 1 1 1 3 3 3 3 7 7 7 7

```c
#include <stdio.h>

int main() {
    int n;
    scanf("%i", &n);
    int array[1000];
    for (int i = 0; i < n; ++i) {
        scanf("%i", &array[i]);
    }
    int k;
    scanf("%i", &k);

    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < k; ++j) {
            printf("%i ", array[n - 1 - i]);
        }
    }
    printf("\n");
}
```

- На вход поступает число `n` и, затем, две последовательности по `n` целых чисел каждая. Сложите эти две последовательности поэлементно и напечатайте её.

вход	выход
4 7 3 1 2 5 1 9 2	12 4 10 4

```c
#include <stdio.h>

int main() {
    int n;
    scanf("%i", &n);
    int array1[1000];
    for (int i = 0; i < n; ++i) {
        scanf("%i", &array1[i]);
    }
    int array2[1000];
    for (int i = 0; i < n; ++i) {
        scanf("%i", &array2[i]);
    }

    for (int i = 0; i < n; ++i) {
        array1[i] += array2[i];
    }
    for (int i = 0; i < n; ++i) {
        printf("%i ", array1[i]);
    }
    printf("\n");
}
```

Двумерные массивы

- На вход поступают числа `n` и `m` и, затем, матрица целых чисел размера `n` строк на `m` столбцов. Напечатайте все суммы строк.

вход	выход
3 4 7 3 1 2 5 1 9 2 7 2 5 4	13 17 18

```
#include <stdio.h>

int main() {
    int n, m;
    scanf("%i%i", &n, &m);
    int matrix[100][100];
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m; ++j) {
            scanf("%i", &matrix[i][j]);
        }
    }

    for (int i = 0; i < n; ++i) {
        int row_sum = 0;
        for (int j = 0; j < m; ++j) {
            row_sum += matrix[i][j];
        }
        printf("%i ", row_sum);
    }
    printf("\n");
}
```

Эту задачу можно решить гораздо более эффективно – без использования двумерного массива:

```
#include <stdio.h>

int main() {
    int n, m;
    scanf("%i%i", &n, &m);
    for (int i = 0; i < n; ++i) {
        int row_sum = 0;
        for (int j = 0; j < m; ++j) {
            int num;
            scanf("%i", &num);
            row_sum += num;
        }
        printf("%i ", row_sum);
    }
    printf("\n");
}
```

Если вы запустите эту программу в терминале, то сумма строки будет печататься после ввода каждой строки. Это происходит потому что и ввод и вывод производится в/из одного места. Если же бы мы считывали, к примеру, из одного файла, а записывали в другой, то программа работала бы как надо.

- На вход поступают числа `n` и `m` и, затем, матрица целых чисел размера `n` строк на `m` столбцов. Напечатайте все суммы столбцов.

вход	выход
3 4 7 3 1 2 5 1 9 2 7 2 5 4	19 6 15 8

```
#include <stdio.h>
int main() {
    int n, m;
    scanf("%i%i", &n, &m);
    int matrix[100][100];
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m; ++j) {
            scanf("%i", &matrix[i][j]);
        }
    }

    int column_sums[100] = {};
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m; ++j) {
            column_sums[j] += matrix[i][j];
        }
    }

    for (int j = 0; j < m; ++j) {
        printf("%i ", column_sums[j]);
    }
    printf("\n");
}
```

Эту задачу можно решить гораздо более эффективно – без использования двумерного массива:

```
#include <stdio.h>
int main() {
    int n, m;
    scanf("%i%i", &n, &m);

    int column_sums[100] = {};
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m; ++j) {
            int num;
            scanf("%i", &num);
            column_sums[j] += num;
        }
    }

    for (int j = 0; j < m; ++j) {
        printf("%i ", column_sums[j]);
    }
    printf("\n");
}
```

- На вход поступают числа `n` и `m` и, затем, матрица целых чисел размера `n` строк на `m` столбцов. Напечатайте индексы наибольшего элемента матриц. Нумерация строк и столбцов начинается с `0`.

вход	выход
3 4 7 3 1 2 5 1 9 2 7 2 5 4	1 2

```c
#include <stdio.h>
#include <limits.h>
int main() {
    int n, m;
    scanf("%i%i", &n, &m);

    int max = INT_MIN;
    int imax = 0;
    int jmax = 0;
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m; ++j) {
            int num;
            scanf("%i", &num);
            if (num > max) {
                max = num;
                imax = i;
                jmax = j;
            }
        }
    }

    printf("%i %i %i\n", max, imax, jmax);
}
```

- На вход поступают числа `n` и `m` и, затем, матрица целых чисел размера `n` строк на `m` столбцов. Поменяйте последние 2 столбца местами и напечатайте.

вход	выход
3 4	7 3 2 1
7 3 1 2	5 1 2 9
5 1 9 2	7 2 4 5
7 2 5 4	

```c
#include <stdio.h>

int main() {
    int n, m;
    scanf("%i%i", &n, &m);
    int array[100][100];
    for (int i = 0; i < n; ++i) {
       for (int j = 0; j < m; ++j) {
           scanf("%i", &array[i][j]);
       }
    }
    for (int i = 0; i < n; ++i) {
        int temp = array[i][m - 1];
        array[i][m - 1] = array[i][m - 2];
        array[i][m - 2] = temp;
    }

    for (int i = 0; i < n; ++i) {
       for (int j = 0; j < m; ++j) {
           printf("%i ", array[i][j]);
       }
       printf("\n");
    }
}
```

Функции

- Напишите функцию, которая принимает 2 целых числа и печатает их сумму.

```c
#include <stdio.h>

int func(int a, int b) {
    printf("%i\n", a + b);
}

int main() {
    func(7, 3);
}
```

- Напишите функцию, которая принимает 2 целых числа и возвращает их сумму. Протестируйте эту функцию в `main`.

```c
#include <stdio.h>

int func(int a, int b) {
    return a + b;
}

int main() {
    printf("%i\n", func(7, 3));
}
```

- Напишите функцию, которая принимает 2 вещественных числа a и b и возвращает их среднее геометрическое c.

$$c = \sqrt{a \cdot b}$$

Протестируйте эту функцию в `main`.

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

double func(double a, double b) {
    return sqrt(a * b);
}

int main() {
    printf("%lf\n", func(7, 3));
}
```

Функции и массивы

- Напишите функцию, которая принимает на вход массив целых чисел и печатает сумму этих чисел.

```c
#include <stdio.h>

void func(int array[], int n) {
    int sum = 0;
    for (int i = 0; i < n; ++i) {
        sum += array[i];
    }
    printf("%i\n", sum);
}

int main() {
    int a[5] = {6, 2, 1, 5, 2};
    func(a, 5);
}
```

- Напишите функцию, которая принимает на вход массив целых чисел и возвращает сумму этих чисел. Протестируйте эту функцию в `main`.

```c
#include <stdio.h>

int func(int array[], int n) {
    int sum = 0;
    for (int i = 0; i < n; ++i) {
        sum += array[i];
    }
    return sum;
}

int main() {
    int a[5] = {6, 2, 1, 5, 2};
    printf("%i\n", func(a, 5));
}
```

- Напишите функцию, которая принимает на вход массив вещественных чисел и возвращает среднее значение этих чисел. Протестируйте эту функцию в `main`.

```c
#include <stdio.h>

double average(double array[], int n) {
    double sum = 0;
    for (int i = 0; i < n; ++i) {
        sum += array[i];
    }
    return sum / n;
}

int main() {
    double a[5] = {6, 2, 1, 5, 2};
    printf("%lf\n", average(a, 5));
}
```

- Напишите функцию, которая принимает на вход массив целых чисел и возвращает `1` если все эти числа делятся на `7`. Если хотя бы одно из чисел не делится на `7`, то функция должна вернуть `0`.

```c
#include <stdio.h>

int is_all_div7(int array[], int n) {
    for (int i = 0; i < n; ++i) {
        if (array[i] % 7 != 0) {
            return 0;
        }
    }
    return 1;
}

int main() {
    int a[5] = {7, 147, 7, 14, 21};
    printf("%i\n", is_all_div7(a, 5));
}
```

Функции. Рекурсия

- Напишите рекурсивную функцию, которая будет принимать целое положительное число и возвращать сумму цифр в этом числе.

```c
#include <stdio.h>

int digit_sum(int a) {
    if (a < 10) {
        return a;
    }
    return a % 10 + digit_sum(a / 10);
}

int main() {
    printf("%i\n", digit_sum(54316));
}
```

Символы

- Напишите программу, которая принимает на вход число `n` и 1 символ и печатает этот символ `n` раз.

```c
#include <stdio.h>

int main() {
    int n;
    char x;
    scanf("%i %c", &n, &x);
    for (int i = 0; i < n; ++i) {
        printf("%c", x);
    }
    printf("\n");
}
```

Простые алгоритмы сортировки ($O(N^2)$)

- На вход поступает число `n` и, затем, `n` целых чисел. Отсортируйте эти числа по возрастанию и напечатайте.

```c
#include <stdio.h>

int main() {
    int n;
    scanf("%i", &n);
    int array[1000];
    for (int i = 0; i < n; ++i) {
        scanf("%i", &array[i]);
    }

    for (int i = 0; i < n; ++i) {
        int min_index = i;
        for (int j = i; j < n; ++j) {
            if (array[j] < array[min_index]) {
                min_index = j;
            }
        }
        int temp = array[i];
        array[i] = array[min_index];
        array[min_index] = temp;
    }

    for (int i = 0; i < n; ++i) {
        printf("%i ", array[i]);
    }
    printf("\n");
}
```

- На вход поступает число `n` и, затем, `n` целых чисел. Напишите функцию, которая будет сортировать эти числа. Примените эту функцию в `main` и напечатайте эти числа.

```c
#include <stdio.h>

void sort(int array[], int n) {
    for (int i = 0; i < n; ++i) {
        int min_index = i;
        for (int j = i; j < n; ++j) {
            if (array[j] < array[min_index]) {
                min_index = j;
            }
        }
        int temp = array[i];
        array[i] = array[min_index];
        array[min_index] = temp;
    }
}

int main() {
    int n;
    scanf("%i", &n);
    int array[1000];
    for (int i = 0; i < n; ++i) {
        scanf("%i", &array[i]);
    }

    sort(array, n);

    for (int i = 0; i < n; ++i) {
        printf("%i ", array[i]);
    }
    printf("\n");
}
```

Структуры и функции

- Структура `struct point` задаётся следующим образом.

```
struct point {
    float x, y;
};
```

  - Напишите функцию, которая не будет ничего принимать, а будет возвращать точку с координатами `(7, 5)`.

```
struct point func1() {
    struct point a = {7, 5};
    return a;
}
```

  - Напишите функцию, которая будет принимать точку, и печатать её.

```
void func2(struct point a) {
    printf("%i %i\n", a.x, a.y)
}
```

  - Напишите функцию, которая будет принимать точку, менять местами координаты и возвращать её.

```
struct point func3(struct point a) {
    float temp = a.x;
    a.x = a.y;
    a.y = temp;
    return a;
}
```

  - Напишите функцию, которая будет принимать 2 точки, и возвращать точку, которая лежит посередине между ними.

```
struct point func4(struct point a, struct point b) {
    struct point c = {(a.x + b.x) / 2, (a.y + b.y) / 2};
    return c;
}
```

  - Напишите функцию, которая не будет ничего возвращать, а будет принимать указатель на точку, и менять местами координаты.

```
void func5(struct point* p) {
    float temp = p->x;
    p->x = p->y;
    p->y = temp;
}
```

  - Протестируйте все эти функции в `main`.

- Создайте массив из 10-ти элементов типа `struct point` в функции main. Значения задайте сами.

- Отсортируйте все эти точки по первой координате и напечатайте.

```c
#include <stdio.h>

struct point {
    float x, y;
};
typedef struct point Point;

void sort(Point array[], int n) {
    for (int i = 0; i < n; ++i) {
        int min_index = i;
        for (int j = i; j < n; ++j) {
            if (array[j].x < array[min_index].x) {
                min_index = j;
            }
        }
        Point temp = array[i];
        array[i] = array[min_index];
        array[min_index] = temp;
    }
}

int main() {
    Point array[]  = {{5, 1}, {6, 3}, {1, 2}, {4, 1}, {7, -1}};

    sort(array, 5);

    for (int i = 0; i < 5; ++i) {
        printf("(%g, %g) ", array[i].x, array[i].y);
    }
    printf("\n");
}
```

- Отсортируйте все эти точки по удалению от начала координат и напечатайте.

```c
#include <stdio.h>

struct point {
    float x, y;
};
typedef struct point Point;

float sqdist(Point a) {
    return a.x * a.x + a.y * a.y;
}

void sort(Point array[], int n) {
    for (int i = 0; i < n; ++i) {
        int min_index = i;
        for (int j = i; j < n; ++j) {
            if (sqdist(array[j]) < sqdist(array[min_index])) {
                min_index = j;
            }
        }
        Point temp = array[i];
        array[i] = array[min_index];
        array[min_index] = temp;
    }
}

int main() {
    Point array[]  = {{5, 1}, {6, 3}, {1, 2}, {4, 1}, {7, -1}};

    sort(array, 5);

    for (int i = 0; i < 5; ++i) {
        printf("(%g, %g) ", array[i].x, array[i].y);
    }
    printf("\n");
}
```

- Напишите функцию, которая принимает на вход массив точек и возвращает точку – центр масс этих точек (при условии, что все точки имеют одинаковую массу).

```c
#include <stdio.h>

struct point {
    float x, y;
};
typedef struct point Point;


Point center(Point array[], int n) {
    Point sum = {0, 0};
    for (int i = 0; i < n; ++i) {
        sum.x += array[i].x;
        sum.y += array[i].y;
    }
    sum.x /= n;
    sum.y /= n;
    return sum;
}

int main() {
    Point array[]  = {{5, 1}, {6, 3}, {1, 2}, {4, 1}, {7, -1}};

    Point c = center(array, 5);

    printf("(%g, %g)\n", c.x, c.y);
}
```