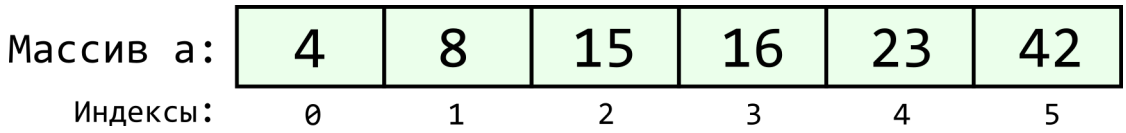
Семинар #3: Массивы.

Часть 1: Основы работы с массивом

Создаём массив из шести элементов и печатаем элемент с индексом 1, то есть число 8.

```
#include <stdio.h>
int main() {
        int a[6] = {4, 8, 15, 16, 23, 42};
        printf("%d\n", a[1]);
}
```



Задачи:

- Напечатайте первый элемент массива (то есть число 4).
- Напечатайте сумму двух послених элементов массива.
- Измените элемент с индексом 1 с 8 на 20 и напечатайте его. Чтобы изменить элемент массива:

```
a[1] = 20;
```

Работа с элементами массива в цикле

Часто к элементам массива необходимо обращаться в цикле. Пример программы, которая прибавляет 1 к каждому элементу массива a и печатает этот массив.

```
#include <stdio.h>
int main() {
    int a[6] = {4, 8, 15, 16, 23, 42};
    for (int i = 0; i < 6; ++i) {
        a[i] += 1;
    }
    for (int i = 0; i < 6; ++i) {
        printf("%d ", a[i]);
    }
}
```

Задачи:

- Напечатайте массив a так, чтобы элементы были разделены запятыми.
- Напечатайте массив a так, чтобы каждый элемент печатался в новой строке.
- Напечатайте массив a два раза (через пробел).
- Напечатайте каждый второй элемент массива a. (Получится `4 15 23`).
- Напечатайте все чётные элементы массива a. (`4 8 16 42`).
- Напечатайте массив a наоборот. (`42 23 16 15 8 4`).

- Напечатайте каждый второй элемент массива `a` наоборот. (`42 16 8`).
- Напечатайте индексы и элементы массива в следующем формате:

```
0:  4
1:  8
2: 15
3: 16
4: 23
5: 42
```

  Для печати используйте `%2d` или `%02d`.

- Напечатайте все элементы массива, умноженные на `2`, не меняя массив.

Во всех следующих задачах нужно сначала менять сам массив, а затем печатать его элементы.

- Измените массив, умножив все его элементы на `2`. Напечатайте его.
- Прибавьте к элементам изначального массива `a` их индексы и напечатайте массив. Должно получиться:

```
4 9 17 19 27 47
```

- Измените изначальный массив `a` следующим образом:
  - Если число чётное, то его нужно разделить на `2`
  - Если число нечётное, то его нужно умножить на `3` и прибавить 1

  Напечатайте этот массив. Должно получиться:

```
2 4 46 8 70 21
```

Считывание элементов массива в цикле

Считывать нужно каждый элемент массив по отдельности в цикле. Но тут возникает проблема: мы не знаем количество элементов, которые придут на вход. Поэтому сначала мы считываем количество элементов `n`. Тут возникает ещё одна проблема: очень нежелательно создавать массив переменной длины. То есть не нужно писать так:

```
int n = 10;
int a[n];
```

Некоторые компиляторы `C/C++` это поддерживают, а некоторые нет. Поэтому лучше создать массив побольше и работать с ним. Пример программы, которая считывает массив и печатает его:

```
#include <stdio.h>
int main() {
    int n;
    scanf("%d", &n);
    int a[1000];
    for (int i = 0; i < n; ++i) {
        scanf("%d", &a[i]);
    }

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

Задачи:

- Считайте массив и напечатайте его 2 раза.

вход	выход
3 1 6 2	1 6 2 1 6 2

- Считайте массив и напечатайте его в обратном порядке.

вход	выход
4 1 7 4 6	6 4 7 1

- Сдвиньте все элементы массива `a` вправо на `1`. Первый элемент должен стать вторым, второй первым третьим, ... последний первым. Сначала нужно изменить массив, а потом напечатать его.

вход	выход
5 4 2 9 6 1	1 4 2 9 6

- Поменяйте местами соседей. То есть первый элемент меняется со вторым, третий с четвёртым. Если количество элементов нечётно, то последний элемент не меняется. Сначала нужно изменить массив, а потом напечатать его.

вход	выход
6 1 2 3 4 5 6	2 1 4 3 6 5
5 4 2 9 6 1	2 4 6 9 1

- Обратите массив. То есть первый элемент должен стать последним, а последний первым. Второй – предпоследним, а предпоследний – вторым и т.д. Сначала нужно изменить массив, а потом напечатать его.

вход	выход
6 1 2 3 4 5 6	6 5 4 3 2 1
3 7 4 2	2 4 7

- На вход подаётся массив, нужно применить к каждому элементу функцию факториала:

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

Сначала нужно напечатать сами элементы в строку, а затем значения функции факториала.

вход	выход
5 3 1 5 0 4	3 1 5 0 4 6 1 120 1 24

Часть 2: Основные алгоритмы

Поиск максимума

Пример программы, которая считывает массив и печатает максимальный элемент. ($n \geq 1$).

```
#include <stdio.h>
int main() {
    int n;
    scanf("%d", &n);
    int a[1000];
    for (int i = 0; i < n; ++i) {
        scanf("%d", &a[i]);
    }
    int max = a[0];
    for (int i = 1; i < n; ++i) {
        if (a[i] > max)
            max = a[i];
    }
    printf("%d\n", max);
}
```

- Измените программу выше так, чтобы программа искала не максимум, а индекс максимального элемента

вход	выход
6 7 1 5 2 9 5	4
3 5 7 4	1
1 1	0

- Измените программу выше так, чтобы программа меняла местами максимальный и первый элементы.

вход	выход
6 7 1 5 2 9 5	9 1 5 2 7 5
3 5 7 4	7 5 4
1 1	1

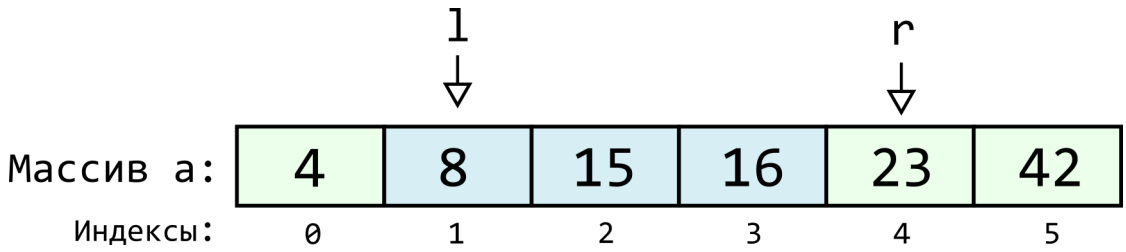
- **Поиск элемента в массиве.** На вход поступает массив и некоторое число. Нужно напечатать индекс этого числа в массиве или `-1` если такого элемента в массиве нет. Если элементов несколько, то нужно напечатать индекс первого из них.

вход	выход
6 7 1 5 2 9 5 5	2
6 7 1 5 2 9 5 4	-1
1 1 1	0

Алгоритмы на подмассивах

Подмассив - это некоторая последовательная часть массива. Будем обозначать подмассив `a[l, r]` такую часть массива, элементы которого имеют индекс `i` в диапазоне `l <= i < r`. Обратите внимание, что мы договорились, что элемент `a[r]` не входит в подмассив `a[l, r]`.

Например, в подмассив `[1, 4]` массива `a` входят элементы `8, 15, 16`, а элемент `23` не входит.



Пример программы, которая считывает массив и границы подмассива в этом массиве и печатает этот подмассив.

```
#include <stdio.h>
int main() {
    int n;
    scanf("%d", &n);
    int a[1000];
    for (int i = 0; i < n; ++i) {
        scanf("%d", &a[i]);
    }
    int l, r;
    scanf("%d%d", &l, &r);

    for (int i = l; i < r; ++i) {
        printf("%d\n", a[i]);
    }
}
```

- Измените программу выше так, чтобы программа печатала сумму подмассива

вход	выход
6 7 1 5 9 2 5 1 4	15

- Измените программу выше так, чтобы программа находила индекс минимального элемента на подмассиве

вход	выход
6 7 1 5 9 2 5 2 6	4

- Измените программу выше так, чтобы программа меняла местами минимальный и первый элемент подмассива

вход	выход
6 7 1 2 9 5 5 2 6	7 1 2 5 9 5

Сортировка

Сортировка – это упорядочение элементов по возрастанию, убыванию или по какому-то другому критерию.

- **Сортировка выбором** – это простейший алгоритм сортировки, который заключается в следующем: Для каждого подмассива `[i, n]` (где `i` последовательно меняется от `0` до `n - 1`) поменять местами первый элемент этого подмассива и минимальный. Напишите эту сортировку. Нужно сначала отсортировать массив, а потом его напечатать.

вход	выход
6 7 1 2 9 5 5	1 2 5 5 7 9

- Сделайте так, чтобы сортировка выбором сортировала по убыванию.

вход	выход
6 7 1 2 9 5 5	9 7 5 5 2 1

- Сделайте так, чтобы сортировка выбором сортировала по возрастанию последней цифры. То есть впереди будут идти числа, у которых последняя цифра – наименьшая.

вход	выход
6 65 41 28 92 153 2	41 92 2 153 65 28

- **Сортировка пузырьком** – это простейший алгоритм сортировки, который заключается в следующем: Для каждого подмассива `[0, n-i]` мы делаем следующую операцию: пробегаем по этому подмассиву и, если соседние элементы стоят неправильно, то меняем их местами. Напишите эту сортировку. Нужно сначала отсортировать массив, а потом его напечатать.

- Сделайте так, чтобы сортировка пузырьком сортировала по возрастанию последней цифры.

Двумерные массивы

Пример программы, которая считывает массив размера `nxm`, прибавляет к каждому элементу `1` и печатает его:

```
#include <stdio.h>
int main() {
    int n, m;
    scanf("%d%d", &n, &m);
    int a[100][100];
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m; ++j) {
            scanf("%d", &a[i][j]);
        }
    }
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m; ++j) {
            a[i][j] += 1;
        }
    }
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m; ++j) {
            printf("%d ", a[i][j]);
        }
        printf("\n");
    }
}
```

- Изменить программу выше так, чтобы каждый элемент массива возводился в квадрат.

вход	выход
3 3 1 2 3 6 5 4 7 8 9	1 4 9 36 25 16 49 64 81

- Напишите программу, которая считывает двумерный массив `nxm` и печатает его сумму

вход	выход
3 3 1 2 3 6 5 4 7 8 9	45

- Напишите программу, которая считывает двумерный массив `nxn` и печатает сумму чисел на двух диагоналях:

вход	выход
3 6 1 2 7 5 4 4 6 9	20 11

  Для вычисления этих сумм использовать 1 цикл (невложенный).

Простое чтение из файла

```
#include <stdio.h>
int main() {
    FILE* f = fopen("input.txt", "r");

    int a;
    fscanf(f, "%d", &a);

    printf("%d\n", a);
    fclose(f);
}
```

- `FILE* f = fopen("input.txt", "r")` - "открывает"файл `input.txt`. Такой файл должен лежать в той же папке, что и исполняемый файл. `"r"` означает, что файл открывается на чтение (read). Для записи в файл нужно использовать `"w"` (write).
- Теперь, когда файл открыт, нужно использовать `fscanf`, чтобы считать из него. Эта функция работает также, как и `scanf`, только первым аргументом нужно пережать `f`.
- Аналогично, есть функция `fprintf`, которая записывает в файл.

Задачи

- С помощью текстового редактора создайте файл `input.txt` и запишите в него число. Считайте это число в программе и напечатайте на экран.
- Считайте числа из файла `numbers.txt`, отсортируйте их и напечатайте на экран.
- Считайте числа из файла `special_numbers.txt`, отсортируйте их и сохраните в файл `sorted.txt`.