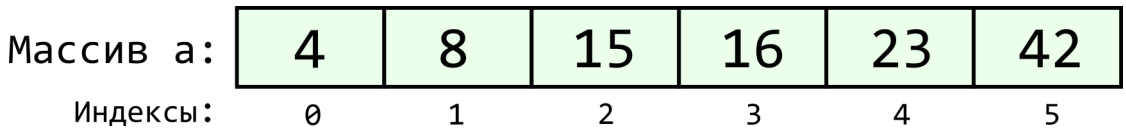
Семинар #3: Массивы.

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;
```

```
#include <stdio.h>
int main() {
        int a[6] = {4, 8, 15, 16, 23, 42};
        printf("%d\n", a[0]);
        printf("%d\n", a[4] + a[5]);
        a[1] = 20;
        printf("%d\n", a[1]);
}
```

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.

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

- Напечатайте массив `a` так, чтобы каждый элемент печатался в новой строке.

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

- Напечатайте массив `a` два раза (через пробел).

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

Или так:

```
#include <stdio.h>
int main() {
    int a[6] = {4, 8, 15, 16, 23, 42};
    for (int j = 0; j < 2; ++j) {
        for (int i = 0; i < 6; ++i) {
            printf("%d ", a[i]);
```

```
        }
    }
}
```

- Напечатайте каждый второй элемент массива `a`. (Получится `4 15 23`).

```
#include <stdio.h>
int main() {
    int a[6] = {4, 8, 15, 16, 23, 42};
    for (int i = 0; i < 6; i += 2) {
        printf("%d ", a[i]);
    }
}
```

- Напечатайте все чётные элементы массива `a`. (`4 8 16 42`).

```
#include <stdio.h>
int main() {
    int a[6] = {4, 8, 15, 16, 23, 42};
    for (int i = 0; i < 6; i++) {
        if (a[i] % 2 == 0)
            printf("%d ", a[i]);
    }
}
```

- Напечатайте массив `a` наоборот. (`42 23 16 15 8 4`).
  Вариант 1:

```
#include <stdio.h>
int main() {
    int a[6] = {4, 8, 15, 16, 23, 42};
    for (int i = 5; i >= 0; i--) {
        printf("%d ", a[i]);
    }
}
```

  Вариант 2:

```
#include <stdio.h>
int main() {
    int a[6] = {4, 8, 15, 16, 23, 42};
    for (int i = 0; i < 6; i++) {
        printf("%d ", a[5 - i]);
    }
}
```

- Напечатайте каждый второй элемент массива `a` наоборот. (`42 16 8`).

```c
#include <stdio.h>
int main() {
    int a[6] = {4, 8, 15, 16, 23, 42};
    for (int i = 5; i >= 0; i -= 2) {
        printf("%d ", a[i]);
    }
}
```

- 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`.

```c
#include <stdio.h>
int main() {
    int a[6] = {4, 8, 15, 16, 23, 42};
    for (int i = 0; i < 6; i++) {
        printf("%d: %2d\n", i, a[i]);
    }
}
```

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

```c
#include <stdio.h>
int main() {
    int a[6] = {4, 8, 15, 16, 23, 42};
    for (int i = 0; i < 6; i++) {
        printf("%d ", 2 * a[i]);
    }
}
```

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

- Измените массив, умножив все его элементы на 2. Напечатайте его.

```c
#include <stdio.h>
int main() {
    int a[6] = {4, 8, 15, 16, 23, 42};
    for (int i = 0; i < 6; i++) {
        a[i] *= 2;
    }
    for (int i = 0; i < 6; i++) {
        printf("%d ", a[i]);
    }
}
```

- Прибавьте к элементам изначального массива `a` их индексы и напечатайте массив. Должно получиться:

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

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

- 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
```

```c
#include <stdio.h>
int main() {
    int a[6] = {4, 8, 15, 16, 23, 42};
    for (int i = 0; i < 6; i++) {
        if (a[i] % 2)
            a[i] = 3 * a[i] + 1;
        else
            a[i] /= 2;
    }
    for (int i = 0; i < 6; i++) {
        printf("%d ", a[i]);
    }
}
```

Вариант 2 (с использованием тернарного оператора):

```
#include <stdio.h>
int main() {
    int a[6] = {4, 8, 15, 16, 23, 42};
    for (int i = 0; i < 6; i++) {
        a[i] = (a[i] % 2) ? 3 * a[i] + 1 : a[i] / 2;
    }
    for (int i = 0; i < 6; i++) {
        printf("%d ", a[i]);
    }
}
```

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

```
#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 j = 0; j < 2; ++j) {
        for (int i = 0; i < n; ++i) {
            printf("%d ", a[i]);
        }
    }
}
```

- Считайте массив и напечатайте его в обратном порядке.

вход	выход
4 1 7 4 6	6 4 7 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]);
    }
    for (int i = n - 1; i >= 0; --i) {
        printf("%d ", a[i]);
    }
}
```

- 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

```c
#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 temp = a[n - 1];
    for (int i = n - 1; i > 0; --i) {
        a[i] = a[i - 1];
    }
    a[0] = temp;
    for (int i = 0; i < n; ++i) {
        printf("%d ", a[i]);
    }
}
```

- 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

```c
#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 - 1; i += 2) {
        int temp = a[i];
        a[i] = a[i + 1];
        a[i + 1] = temp;
    }
    for (int i = 0; i < n; ++i) {
        printf("%d ", a[i]);
    }
}
```

- 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

```c
#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 / 2; i++) {
        int temp = a[i];
        a[i] = a[n - 1 - i];
        a[n - 1 - i] = temp;
    }
    for (int i = 0; i < n; ++i) {
        printf("%d ", a[i]);
    }
}
```

- 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

```
#include <stdio.h>
int fact(int n) {
    int result = 1;
    for (int i = 1; i <= n; ++i) {
        result *= i;
    }
    return result;
}

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("%-3d ", a[i]);
    }
    printf("\n");
    for (int i = 0; i < n; ++i) {
        printf("%-3d ", fact(a[i]));
    }
}
```

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

```
#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_i = 0;
    for (int i = 1; i < n; ++i) {
        if (a[i] > a[max_i])
            max_i = i;
    }
    printf("%d\n", max_i);
}
```

- 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

```c
#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_i = 0;
    for (int i = 1; i < n; ++i) {
        if (a[i] > a[max_i])
            max_i = i;
    }
    int temp = a[max_i];
    a[max_i] = a[0];
    a[0] = temp;
    for (int i = 0; i < n; ++i) {
        printf("%d ", a[i]);
    }
}
```

- **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.

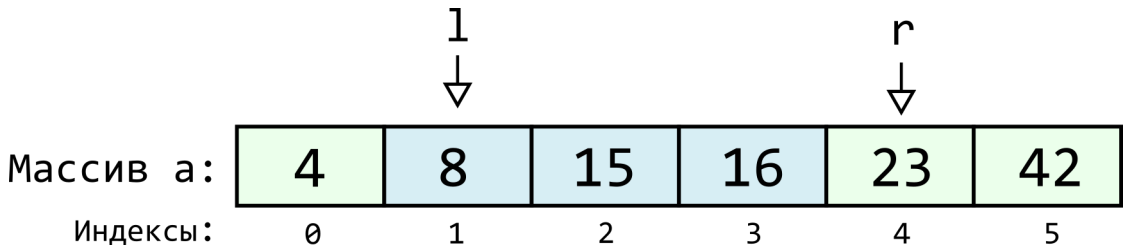
вход	выход
6 7 1 5 2 9 5 5	2
6 7 1 5 2 9 5 4	-1
1 1 1	0

```c
#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 x;
    scanf("%d", &x);

    int result = -1;
    for (int i = 0; i < n; ++i) {
        if (a[i] == x) {
            result = i;
            break;
        }
    }

    printf("%d ", result);
}
```

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

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

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

вход	выход
6 7 1 5 9 2 5 2 6	4

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

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

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

```c
#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);
    int min_i = l;
    for (int i = l; i < r; ++i) {
        if (a[i] < a[min_i])
            min_i = i;
    }
    int temp = a[min_i];
    a[min_i] = a[l];
    a[l] = temp;
    for (int i = 0; i < n; ++i) {
        printf("%d ", a[i]);
    }
}
```

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

```
#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) {
        int min_j = i;
        for (int j = i + 1; j < n; ++j) {
            if (a[j] < a[min_j])
                min_j = j;
        }
        int temp = a[min_j];
        a[min_j] = a[i];
        a[i] = temp;
    }

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

- Сделайте так, чтобы сортировка выбором сортировала по убыванию.

вход	выход
6 7 1 2 9 5 5	9 7 5 5 2 1

В этой строке нужно изменить знак:

```
if (a[j] < a[min_j])     --->    if (a[j] > a[min_j])
```

- 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

```
if (a[j] < a[min_j])     --->    if (a[j] % 10 < a[min_j] % 10)
```

- **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.

```
#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) {
        for (int j = 0; j < n - i - 1; ++j) {
            if (a[j] > a[j + 1]) {
                int temp = a[j];
                a[j] = a[j + 1];
                a[j + 1] = temp;
            }
        }
    }

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

- Сделайте так, чтобы сортировка пузырьком сортировала по возрастанию последней цифры. В этой строке нужно изменить:

```
if (a[j] > a[j + 1]) {     --->    if (a[j] % 10 > a[j + 1] % 10) {
```

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

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

- 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).

```c
#include <stdio.h>
int main() {
    int n;
    scanf("%d", &n);
    int a[100][100];
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            scanf("%d", &a[i][j]);
        }
    }
    int sum1 = 0, sum2 = 0;
    for (int i = 0; i < n; ++i) {
        sum1 += a[i][i];
        sum2 += a[i][n - 1 - i];
    }
    printf("%d %d\n", sum1, sum2);
}
```

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.

```
#include <stdio.h>
int main() {
    FILE* f = fopen("numbers.txt", "r");
    int n;
    fscanf(f, "%d", &n);
    int a[1000];
    for (int i = 0; i < n; ++i) {
        fscanf(f, "%d", &a[i]);
    }
    fclose(f);
    for (int i = 0; i < n; ++i) {
        int min_j = i;
        for (int j = i + 1; j < n; ++j) {
            if (a[j] < a[min_j])
                min_j = j;
        }
        int temp = a[min_j];
        a[min_j] = a[i];
        a[i] = temp;
    }
    for (int i = 0; i < n; ++i) {
        printf("%d ", a[i]);
    }
}
```

- Считайте числа из файла `special_numbers.txt`, отсортируйте их и сохраните в файл `sorted.txt`.

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

    int n;
    fscanf(infile, "%d", &n);
    int a[21000];
    for (int i = 0; i < n; ++i) {
        fscanf(infile, "%d", &a[i]);
    }
    fclose(infile);

    for (int i = 0; i < n; ++i) {
        int min_j = i;
        for (int j = i + 1; j < n; ++j) {
            if (a[j] < a[min_j])
                min_j = j;
        }
        int temp = a[min_j];
        a[min_j] = a[i];
        a[i] = temp;
    }
    FILE* outfile = fopen("sorted.txt", "w");
    for (int i = 0; i < n; ++i) {
        fprintf(outfile, "%d ", a[i]);
    }
    fclose(outfile);
}
```