

**ТЕМА: ВВЕДЕНИЕ В ЯЗЫК
ПРОГРАММИРОВАНИЯ «C++»****Домашнее задание 2****ЗАДАНИЕ 1**

В одномерном массиве, заполненном случайными числами, определить минимальный и максимальный элементы.

Подсказка 1

Если у Вас возникли сложности с выполнением данного задания, нажмите кнопку «Подсказка». С полным решением задания вы сможете ознакомиться, нажав на кнопку «Решение»

Решение 1**ЗАДАНИЕ 2**

В одномерном массиве, заполненном случайными числами в заданном пользователем диапазоне, найти сумму элементов, значения которых меньше указанного пользователем.

Подсказка 2

Если у Вас возникли сложности с выполнением данного задания, нажмите кнопку «Подсказка». С полным решением задания вы сможете ознакомиться, нажав на кнопку «Решение»

Решение 2**ЗАДАНИЕ 3**

Пользователь вводит прибыль фирмы за год (12 месяцев). Затем пользователь вводит диапазон (например, 3 и 6 — поиск между 3-м и 6-м месяцем). Необходимо определить месяц, в котором прибыль была максимальна и месяц, в котором прибыль была минимальна с учетом выбранного диапазона.

Подсказка 3

Если у Вас возникли сложности с выполнением данного задания, нажмите кнопку «Подсказка». С полным решением задания вы сможете ознакомиться, нажав на кнопку «Решение»

Решение 3

ЗАДАНИЕ 4

В одномерном массиве, состоящем из N вещественных чисел вычислить:

- Сумму отрицательных элементов.
- Произведение элементов, находящихся между `min` и `max` элементами.
- Произведение элементов с четными номерами.
- Сумму элементов, находящихся между первым и последним отрицательными элементами.

Финальное задание рассчитано на самостоятельное решение. Здесь не будет подсказок и готового алгоритма. Вам необходимо применить все практические навыки, полученные из предыдущих заданий.

ПОДСКАЗКА К ЗАДАНИЮ 1

1. При объявлении массива необходимо указать типа и его размерность.
2. С помощью какой функции в C++ можно сгенерировать случайное число? Необходимо ли подключать дополнительную библиотеку для этого?
3. Для нахождения максимального (минимального) из нескольких чисел (элементов массива) нужно последовательно сравнить их парами, т. е. найденный максимум (минимум) из двух первых чисел сравнивается с третьим числом и т. д. до конца массива
4. В случае, если следующее сравниваемое число больше максимума (меньше минимума), необходимо переопределить максимум (минимум).
5. Какой из операторов условного ветвления нужно выбрать для сравнения двух элементов массива `if- else if` или последовательность из нескольких `if`?

ПОДСКАЗКА К ЗАДАНИЮ 2

1. Какой тип данных используется для работы с отдельными символами?
2. Каждый символ имеет свой код. Чтобы узнать коды нужных символов необходима таблица ASCII-символов (американский стандартный код для обмена информацией, который определяет способ представления символов английского языка в виде чисел от 0 до 127).
3. Каким образом можно реализовать преобразование символьного типа в целочисленный (для получения ASCII-кода символа)?
4. Каким оператором условного ветвления реализуется ситуация, когда нужно последовательно проверить несколько условий (выполниться может только одно из них: символ либо буква, либо цифра, либо знак препинания) и предусмотреть альтернативный шаг в случае, когда они не выполняются?
5. Если необходимо проверить, находится ли число в определенном диапазоне (от — до), то какой логический оператор необходимо использовать в условии?
6. Диапазон кодов у строчных символов один, а у заглавных — другой. Какой логический оператор необходимо использовать в условии, если символ является буквой, когда он принадлежит любому из двух диапазонов кодов?

ПОДСКАЗКА К ЗАДАНИЮ 3

1. Какой будет размерность массива, если необходимо хранить значения прибыли за каждый из 12 месяцев?
2. Если элемент массива соответствует прибыли за определенный месяц, то индекс элемента массива — это номер месяца года, т. е. нужно найти индексы минимального и максимального элемента.
3. Так как нужно найти не значения максимума и минимума, а их индексы, то нужно хранить индексы текущего максимума и текущего минимума.
4. Для нахождения максимального (минимального) из нескольких чисел (элементов массива) нужно последовательно сравнить их парами, т.е. элемент с найденным индексом максимума (минимума) сравнивается со следующим элементом и т. д. до конца массива
5. В случае, если сравниваемое число (элемент массива с текущим индексом) больше, чем элемент с индексом максимума (меньше, чем элемент с индексом минимума), необходимо переопределить индекс максимума (минимума).
6. Так как поиск выполняется в цикле, в котором последовательно анализируются элементы массива, то какие будут начальные и конечные значения переменной — счетчика цикла при условии, что поиск идет не во всем массиве, а в его фрагменте?

РЕШЕНИЕ ЗАДАНИЯ 1

Описание решения

Вначале необходимо определиться с размерностью массива (максимальным количеством элементов, которое может в нем храниться). Пусть в нашем примере длина массива — десять.

Так как в массив будут заноситься случайные числа, то необходимо использовать генератор случайных чисел. Язык C++ имеет свои собственные встроенные генераторы случайных чисел, которые реализованы в двух отдельных функциях:

- функция `srand()` устанавливает передаваемое пользователем значение в качестве стартового (данную функцию следует вызывать только один раз — в начале программы);
- функция `rand()` генерирует следующее случайное число в последовательности. Оно будет находиться в диапазоне от 0 до `RANDMAX` (константа, значением которой является 32767).

Для того, чтобы при каждом новом запуске программы генерировалось новое число в функцию `srand()` в качестве параметра не нужно передавать фиксированное значение.

Общепринятым решением является использование системных часов, т.к. каждый раз, при запуске программы, системное время будет другое. Если мы будем использовать значение времени в качестве стартового числа, то наша программа всегда будет генерировать разную последовательность чисел при каждом новом запуске.

Для получения текущего системного времени будем использовать функцию `time()`, которая возвращает в качестве

времени общее количество секунд, прошедшее от полуночи 1 января 1970 года. Чтобы использовать эту функцию, нам просто нужно подключить заголовочный файл `ctime` (или `#include "time.h"`), а затем инициализировать функцию `srand()` вызовом функции `time(0)` или `time(NULL)`.

Т.к. в каждый элемент массива нужно заносить новое случайное число, то процесс генерации следует повторять в цикле, длина которого будет равна длине массива.

После заполнения массива случайными числами необходимо определить начальное значение максимума и минимума.

Устанавливаем в качестве начального значения как для максимума, так и минимума, значение первого элемента массива.

Далее последовательно двигаясь в цикле по массиву, сравниваем каждый следующий элемент массива с максимумом и минимумом. Если текущий элемент массива больше максимума, то переопределяем максимум (теперь максимум — текущий элемент массива). Соответствующая проверка проводится для этого же элемента массива и текущего минимума.

Например, в массиве находятся числа: 4 5 6 3 7. Начальное значение максимума будет 4. Начальное значение минимума будет 4.

Анализ элементов массива начинаем со второго элемента, т.к. первый уже использован.

Шаг 1:

- число 5 больше максимума (4), переопределяем максимум (максимум равен 5);
- число 5 не меньше минимума (4), минимум не изменяется (4).

Шаг 2:

- число 6 больше максимума (5), переопределяем максимум (максимум равен 6);
- число 6 не меньше минимума (4), минимум не изменяется (4).

Шаг 3:

- число 3 не больше максимума (6), максимум не изменяется (6);
- число 3 меньше минимума (4), минимум переопределяется (3).

Шаг 4:

- число 7 больше максимума (6), максимум переопределяется (7);
- число 7 не меньше минимума (3), минимум не изменяется (3).

Итого: максимум — число 7, минимум — число 3.

Решение

1. Подключаем библиотеку (`#include "time.h"`), которая позволит в дальнейшем использовать встроенную функцию `time()` для получения текущего системного времени, которое необходимо для старта генератора псевдослучайных чисел. Выводим строку в консоль с описанием названия программы (по желанию, данная строка кода может быть пропущена), создаем необходимое число переменных и констант. Три целочисленные константы необходимы для хранения длины массива, начального и конечного значения диапа-

зона генерируемых чисел. В нашем примере пусть длина массива будет 10, а числа генерируются в диапазоне от 10 до 20.

Две целочисленные переменные будут использоваться для хранения максимального и минимального элемента массива.

Далее объявляем целочисленный массив, используя созданную ранее целочисленную константу для задания размера массива.

```
#include<iostream>
#include"time.h"
using namespace std;

int main()
{
    cout << "Home task #8.2.1\n\n";

    int const n = 10, a = 10, b = 20;
    int mass[n];
    int min, max;

    return 0;
}
```

2. Для того, чтобы при каждом новом запуске программы генерировалось новое число используем функцию `srand()`, а в качестве ее параметра — вызов функции `time(NULL)`. Далее организуем цикл (длительность которого равна длине массива), внутри которого выполняются операция генерации случайного числа в заданном диапазоне.

Так как нам не нужны случайные числа между 0 и `RAND_MAX` — нам нужны числа между двумя другими значениями, то следует выполнить преобразование полученного от функции `rand()` числа в число из указанного диапазона. Это выполняется с помощью следующей формулы:

$$a + \text{rand}() \% (b - a),$$

где `a` и `b` — границы нужного диапазона.

После генерации выводим полученное число (элемент массива) в консоль.

```
#include<iostream>
#include"time.h"
using namespace std;

int main()
{
    cout << "Home task #8.2.1\n\n";

    int const n = 10, a = 10, b = 20;
    int mass[n];
    int min, max;
    srand(time(NULL));

    for (int i = 0; i < n; i++)
    {
        mass[i] = a + rand() % (b - a);
        cout << mass[i] << " ";
    }

    return 0;
}
```

3. Устанавливаем в качестве начального значения как для максимума, так и минимума, значение первого элемента массива.

Далее последовательно двигаясь в цикле по массиву, сравниваем каждый следующий элемент массива с максимумом и минимумом.

Если текущий элемент массива больше максимума, то переопределяем максимум (теперь максимум — текущий элемент массива).

Такую проверку выполняем для этого же элемента массива и текущего минимума.

```
#include<iostream>
#include"time.h"
using namespace std;

int main()
{
    cout << "Home task #8.2.1\n\n";

    int const n = 10, a = 10, b = 20;
    int mass[n];
    int min, max;
    srand(time(NULL));

    for (int i = 0; i < n; i++)
    {
        mass[i] = a + rand() % (b - a);
        cout << mass[i] << " ";
    }
    min = mass[0];
    max = mass[0];
```

```
for (int i = 1; i < n; i++)
{
    if (mass[i] > max)
    {
        max = mass[i];
    }

    if (mass[i] < min)
    {
        min = mass[i];
    }
}

return 0;
}
```

4. После окончания цикла, в котором мы проанализировали все элементы массива, выводим значения максимума и минимума в консоль.

```
#include<iostream>
#include"time.h"
using namespace std;

int main()
{
    cout << "Home task #8.2.1\n\n";

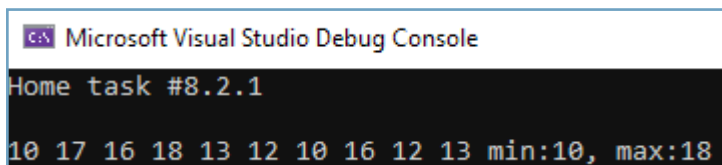
    int const n = 10, a = 10, b = 20;
    int mass[n];
    int min, max;
    srand(time(NULL));
}
```

```
for (int i = 0; i < n; i++)
{
    mass[i] = a + rand() % (b - a);
    cout << mass[i] << " ";
}
min = mass[0];
max= mass[0];
for (int i = 1; i < n; i++)
{
    if (mass[i] > max)
    {
        max = mass[i];
    }

    if (mass[i] < min)
    {
        min = mass[i];
    }
}
cout << "min:" << min << ", max:" << max;

return 0;
}
```

Результаты работы программы (в консоли):



Microsoft Visual Studio Debug Console

Home task #8.2.1

10 17 16 18 13 12 10 16 12 13 min:10, max:18

Рисунок 1

РЕШЕНИЕ ЗАДАНИЯ 2

Описание решения

Вначале необходимо определиться с размерность массива (максимальным количеством элементов, которое может в нем храниться). Пусть в нашем примере длина массива — десять.

Так как в массив будут заноситься случайные числа, то необходимо использовать генератор случайных чисел:

- вызвать один раз (в начале программы) функция `srand()` для установки стартового значения генератора;
- использовать функцию `rand()` для генерации каждого случайного числа.

Для того, чтобы при каждом новом запуске программы генерировалось новое число в функцию `srand()` в качестве параметра нужно передавать новое значение.

Общепринятым решением является использование системного времени. Для получения текущего системного времени будем использовать функцию `time()`. Чтобы использовать эту функцию, нам просто нужно подключить заголовочный файл `ctime` (или `#include "time.h"`), а затем инициализировать функцию `srand()` вызовом функции `time(0)` или `time(NULL)`.

Т.к. в каждый элемент массива нужно заносить новое случайное число, то процесс генерации следует повторять в цикле, длина которого будет равна длине массива.

Так как функция `rand()` генерирует следующее случайное число в от 0 до `RANDMAX` (константа, значением которой является 32767), а нам нужно число из диапазона `[a;b]`, где `a` и `b` — это границы диапазона, которые вводятся пользовате-

лем, то следует выполнить преобразование полученного от функции `rand()` числа в число из указанного диапазона.

Это выполняется с помощью следующей формулы:

$$a + \text{rand}() \% (b - a),$$

где a и b — границы нужного диапазона.

Процесс вычисления суммы элементов можно представить в виде общей формулы:

$$S = S + a_i, \text{ если } a_i < \text{level},$$

где a_i — слагаемое на i — том шаге (элемент массива с i -тым индексом), `level` — число, задаваемое пользователем.

Таким образом, вычисление суммы сводится к ее накоплению в переменной S на каждом шаге цикла.

Например, в массиве находятся числа: 12 8 34 11 16.

Пользователь ввел порог (`level`) равный 15.

Перед началом вычисления (до цикла) суммы еще нет, т.е. ее значение равно нулю.

$$S_0 = 0.$$

Шаг 1:

$$a_1 = 12, 12 < 15, S_1 = S_0 + a_1 = 0 + 12 = 12.$$

Шаг 2:

$$a_2 = 8, 8 < 15, S_2 = S_1 + a_2 = 12 + 8 = 20.$$

Шаг 3:

$$a_3 = 34, 35 > 15, S_3 = S_2.$$

Шаг 4:

$$a_4 = 11, 11 < 15, S_4 = S_3 + a_4 = 20 + 11 = 31.$$

Шаг 5:

$$a_5 = 16, 16 > 15, S_5 = S_4 = 31.$$

Решение

1. Подключаем библиотеку (`#include "time.h"`), которая позволит в дальнейшем использовать встроенную функцию `time()` для получения текущего системного времени, которое необходимо для старта генератора псевдослучайных чисел.

Выводим строку в консоль с описанием названия программы (по желанию, данная строка кода может быть пропущена), создаем необходимое число переменных и констант.

Одна целочисленная константа будет использоваться для хранения длины массива (`10` в нашем примере).

Четыре целочисленные переменные понадобятся для хранения начального и конечного значения диапазона генерируемых чисел, суммы (результата) и уровня, задаваемого пользователем.

```
#include <iostream>
#include "time.h"
using namespace std;

int main()
{
    cout << "Home task #8.2.2\n\n";

    int const n = 10;
    int mass[n];
    int a, b, sum, level;

    return 0;
}
```


2. Для того, чтобы при каждом новом запуске программы генерировалось новое число используем функцию `srand()`, а в качестве ее параметра — вызов функции `time(NULL)`. Спрашиваем у пользователя начальное и конечное значение диапазона для генерируемых чисел, а также уровень для суммирования. Заносим введенные значения в соответствующие переменные.

```
#include <iostream>
#include "time.h"
using namespace std;

int main()
{
    cout << "Home task #8.2.2\n\n";
    int const n = 10;
    int mass[n];
    int a, b, sum, level;

    cout << "Enter the start of the range to generate "
           "numbers:\n";
    cin >> a;
    cout << "Enter the end of the range:\n";
    cin >> b;
    cout << "Enter the level value:\n";
    cin >> level;

    return 0;
}
```

3. Далее организуем цикл (длительность которого равна длине массива), внутри которого выполняются операция

генерации случайного числа и преобразование его в число из заданного пользователем диапазона.

```
#include <iostream>
#include "time.h"
using namespace std;

int main()
{
    cout << "Home task #8.2.2\n\n";
    int const n = 10;
    int mass[n];
    int a, b, sum, level;

    cout << "Enter the start of the range to generate "
           "numbers:\n";
    cin >> a;
    cout << "Enter the end of the range:\n";
    cin >> b;
    cout << "Enter the level value:\n";
    cin >> level;

    for (int i = 0; i < n; i++)
    {
        mass[i] = a + rand() % (b - a);
        cout << mass[i] << " ";
    }
    return 0;
}
```

4. Вычисление суммы производится через ее накопление на каждом шаге цикла. Перед началом цикла устанавливаем значение переменной суммы в ноль. Длина цикла соответ-

ствуем длине массива, так как нам необходимо перебрать все его элементы.

Однако нам необходимо добавлять к сумме только те элементы массива, которые соответствуют условию «значение числа меньше указанного пользователем». Поэтому накопление происходит в блоке условного оператора.

После окончания цикла процесс вычисления суммы завершен, так как проанализированы все элементы массива. Выводим полученный результат (сумму) в консоль.

```
#include <iostream>
#include "time.h"
using namespace std;

int main()
{
    cout << "Home task #8.2.2\n\n";

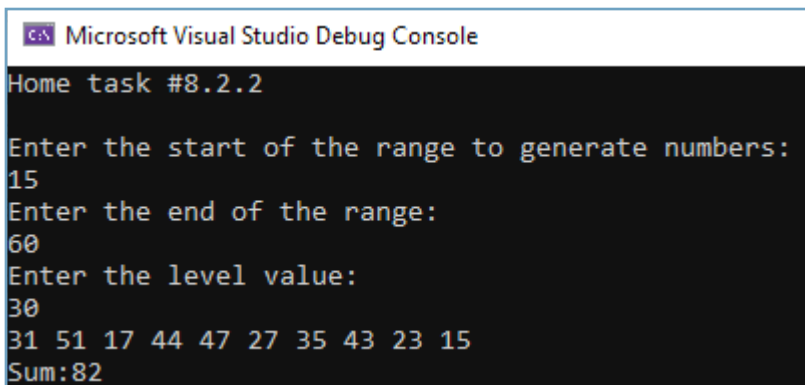
    int const n = 10;
    int mass[n];
    int a, b, sum, level;
    srand(time(NULL));

    cout << "Enter the start of the range to generate "
           "numbers:\n";
    cin >> a;
    cout << "Enter the end of the range:\n";
    cin >> b;
    cout << "Enter the level value:\n";
    cin >> level;

    for (int i = 0; i < n; i++)
    {
```

```
        mass[i] = a + rand() % (b - a);  
        cout << mass[i] << " ";  
    }  
    sum = 0;  
    for (int i = 0; i < n; i++)  
    {  
        if (mass[i] < level)  
        {  
            sum = sum + mass[i];  
        }  
    }  
  
    cout << "\nSum:" << sum ;  
  
    return 0;  
}
```

Результаты работы программы (в консоли):



The screenshot shows the Microsoft Visual Studio Debug Console with the following text:

```
Microsoft Visual Studio Debug Console  
Home task #8.2.2  
Enter the start of the range to generate numbers:  
15  
Enter the end of the range:  
60  
Enter the level value:  
30  
31 51 17 44 47 27 35 43 23 15  
Sum:82
```

Рисунок 2

РЕШЕНИЕ ЗАДАНИЯ 3

Описание решения

Вначале необходимо определиться с размерность массива (максимальным количеством элементов, которое может в нем храниться). В данной задаче необходимо хранить значения прибыли за каждый из 12 месяцев, значит длина массива — двенадцать.

Для заполнения массива данными организуем цикл, в котором будем спрашивать у пользователя величину прибыли для определенного месяца и сохранять введенное значение в соответствующий элемент массива (например, значение, введенное на первом шаге цикла — это величина прибыли в январе, сохраняем в первый элемент массива, т.е. в элемент с нулевым индексом.).

После заполнения массива случайными числами необходимо определить начальное значение индексов максимума и минимума.

Устанавливаем в качестве начальных значений индексов как для максимума, так и минимума, значение стартового индекса (т.е. рассматриваем первый элемент из фрагмента массива как возможный максимум и минимум).

Так как в задаче нужно найти максимальную и минимальную прибыль за указанный пользователем диапазон месяцев, то будем анализировать элементы массива из указанного диапазона, т. е. фрагмент массива.

Далее последовательно двигаясь в цикле по фрагменту массиву, сравниваем каждый следующий элемент массива с тем, который находится в позиции максимума и минимума. Если текущий элемент массива больше того, который находится

в позиции максимума, то переопределяем индекс максимум (теперь это индекс текущего элемента массива). Соответствующая проверка проводится для этого же элемента массива и элемента в позиции текущего минимума.

Например, во фрагменте массиве (диапазон месяцев от 0 до 4) находятся числа: 4 5 6 3 7. Индекс максимума — 0. Индекс минимума — 0.

Анализ элементов массива начинаем со второго элемента, т.к. первый уже использован.

Шаг 1:

- число 5 больше числа в позиции максимума (4), переопределяем индекс максимума (индекс максимума равен 1);
- число 5 не меньше числа в позиции минимума (4), индекс минимума не изменяется (0).

Шаг 2:

- число 6 больше числа в позиции максимума (5), переопределяем индекс максимума (индекс максимума равен 2);
- число 6 не меньше числа в позиции минимума (4), индекс минимума не изменяется (0).

Шаг 3:

- число 3 числа в позиции максимума (6), индекс максимума не изменяется (2);
- число 3 меньше числа в позиции минимума (4), индекс минимума переопределяется (3).

Шаг 4:

- число 7 больше числа в позиции максимума (6), переопределяем индекс максимума (индекс максимума равен 4);

- число 7 не меньше числа в позиции минимума (4), индекс минимума не изменяется (3).

Итого: номер месяца (из указанного диапазона 0 - 4) с максимальной прибылью — 4, номер месяца с минимальной прибылью — 3.

Решение

1. Выводим строку в консоль с описанием названия программы (по желанию, данная строка кода может быть пропущена), создаем необходимое число переменных и констант. Одна целочисленная константа необходима для хранения длины массива. Четыре целочисленные переменные будут использоваться для хранения границ диапазона, и индексов максимального и минимального элемента в диапазоне. Объявляем целочисленный массив, используя созданную ранее целочисленную константу для задания размера массива.

```
#include <iostream>
using namespace std;

int main()
{
    cout << "Home task #8.2.3\n\n";
    int const n = 12;
    int profit[n];
    int start, end, minPrMonth, maxPrMonth;

    return 0;
}
```

2. Далее организуем цикл (длительность которого равна длине массива), в котором будем спрашивать у пользователя величину прибыли для определенного месяца и сохранять введенное значение в соответствующий элемент массива (например, значение, введенное на первом шаге цикла — это величина прибыли в январе, сохраняем в первый элемент массива, т.е. в элемент с нулевым индексом.).

```
#include <iostream>
using namespace std;

int main()
{
    cout << "Home task #8.2.3\n\n";

    int const n = 12;
    int profit[n];
    int start, end, minPrMonth, maxPrMonth;

    for (int i = 0; i < n; i++)
    {
        cout << "Enter your profit for " << i + 1;
        cout << " month:\n";
        cin >> profit[i];
    }

    return 0;
}
```

3. После заполнения массива случайными числами просим пользователя ввести границы диапазона и сохраняем введенные им значения в соответствующие переменные.


```
#include <iostream>
using namespace std;

int main()
{
    cout << "Home task #8.2.3\n\n";

    int const n = 12;
    int profit[n];
    int start, end, minPrMonth, maxPrMonth;

    for (int i = 0; i < n; i++)
    {
        cout << "Enter your profit for " << i + 1;
        cout << " month:\n";
        cin >> profit[i];
    }

    cout << "Enter the start of the range "
           "interested for you:\n";
    cin >> start;
    cout << "Enter the end of the range interested "
           "for you:\n";
    cin >> end;

    return 0;
}
```

4. Теперь необходимо определить начальное значение индексов максимума и минимума. Устанавливаем в качестве начальных значений индексов как для максимума, так и минимума, значение стартового индекса, который ввел пользователь.

Так как в задаче нужно найти максимальную и минимальную прибыль за указанный пользователем диапазон месяцев, то будем анализировать элементы массива из указанного диапазона, т. е. фрагмент массива.

Так как пользователь работает с номерами месяцев от 1 до 12, а индексы элементы массива от 0 до 11, то заданные пользователем начальные и конечные значения (от ... месяца до месяца) следует уменьшить на единицу.

Начальное значение переменной — счетчика цикла — это стартовый индекс.

Далее последовательно двигаясь в цикле по фрагменту массиву (пока значение переменной — счетчика цикла не превысит значение конечного индекса), сравниваем каждый следующий элемент массива с тем, который находится в позиции максимума и минимума. Если текущий элемент массива больше того, который находится в позиции максимума, то переопределяем индекс максимум (теперь это индекс текущего элемент массива). Соответствующая проверка проводится для этого же элемента массива и элемента в позиции текущего минимума.

```
#include <iostream>
using namespace std;

int main()
{
    cout << "Home task #8.2.3\n\n";

    int const n = 12;
    int profit[n];
    int start, end, minPrMonth, maxPrMonth;
```

```
for (int i = 0; i < n; i++)
{
    cout << "Enter your profit for " << i + 1;
    cout << " month:\n";
    cin >> profit[i];
}

cout << "Enter the start of the range interested "
      "for you:\n";
cin >> start;
cout << "Enter the end of the range interested "
      "for you:\n";
cin >> end;
minPrMonth = start-1;
maxPrMonth = start-1;
for (int i = start; i <= end-1; i++)
{
    if (profit[i] > profit[maxPrMonth])
    {
        maxPrMonth = i;
    }
    if (profit[i] < profit[minPrMonth])
    {
        minPrMonth = i;
    }
}
return 0;
}
```

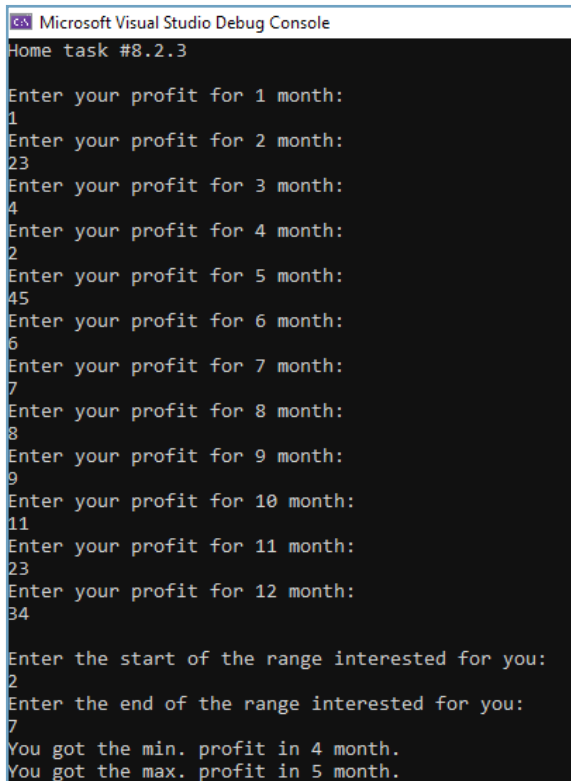
5. После окончания цикла, в котором мы проанализировали все элементы из фрагмента массива, выводим значения индексов максимума и минимума (увеличенные на единицу из-за сдвигов индексов элементов массива) в консоль.

```
#include <iostream>
using namespace std;

int main()
{
    cout << "Home task #8.2.3\n\n";
    int const n = 12;
    int profit[n];
    int start, end, minPrMonth, maxPrMonth;
    for (int i = 0; i < n; i++)
    {
        cout << "Enter your profit for " << i + 1;
        cout << " month:\n";
        cin >> profit[i];
    }
    cout << "Enter the start of the range "
           "interested for you:\n";
    cin >> start;
    cout << "Enter the end of the range interested "
           "for you:\n";
    cin >> end;
    minPrMonth = start-1;
    maxPrMonth = start-1;
    for (int i = start; i <= end-1; i++)
    {
        if (profit[i] > profit[maxPrMonth])
        {
            maxPrMonth = i;
        }
        if (profit[i] < profit[minPrMonth])
        {
            minPrMonth = i;
        }
    }
}
```

```
cout << "You got the min. profit in "  
    << minPrMonth+1 << " month.\n";  
cout << "You got the max. profit in "  
    << maxPrMonth+1 << " month.\n";  
return 0;  
}
```

Результаты работы программы (в консоли):



Microsoft Visual Studio Debug Console

Home task #8.2.3

Enter your profit for 1 month:
1
Enter your profit for 2 month:
23
Enter your profit for 3 month:
4
Enter your profit for 4 month:
2
Enter your profit for 5 month:
45
Enter your profit for 6 month:
6
Enter your profit for 7 month:
7
Enter your profit for 8 month:
8
Enter your profit for 9 month:
9
Enter your profit for 10 month:
11
Enter your profit for 11 month:
23
Enter your profit for 12 month:
34

Enter the start of the range interested for you:
2
Enter the end of the range interested for you:
7
You got the min. profit in 4 month.
You got the max. profit in 5 month.

Рисунок 3