Занятие № 6.

Реализация базовых алгоритмов (сортировка и поиск)

1. Поиск минимума и максимума

Рассмотрим массив из чисел A[n]. Для поиска минимального элемента нужно взять первый элемент и сравнивать его со всеми элементами массива.

```
#include <iostream>

using namespace std;

int main()
{
    setlocale(LC_ALL,    "Russian");//    устанавливаем    русскую
кодировку

     int N = 10; // количество элементов

     int A[N];

     A[0] = 3;
     A[1] = 2;
     A[2] = -2;
     A[3] = 0;
     A[4] = 10;
     A[5] = 3;
     A[6] = -4;
     A[7] = 5;
     A[8] = 9;
     A[9] = -1;

     int min = A[0]; // берем первый элемент массива
```

```
    for (int i = 1; i < N; i++) // перебираем все элементы
    {
         if (A[i] < min)
       {
           min = A[i]; // обновляем минимальный элемент
       }
    }

    cout << "Минимальный элемент = " << min << endl;

    return 0;
}
```

Аналогично находится и максимальный элемент.

```
#include <iostream>

using namespace std;

int main()
{
    setlocale(LC_ALL,    "Russian");//    устанавливаем    русскую
кодировку

    int N = 10; // количество элементов

    int A[N];

    A[0] = 3;
    A[1] = 2;
    A[2] = -2;
    A[3] = 0;
    A[4] = 10;
    A[5] = 3;
    A[6] = -4;
```

```
    A[7] = 5;
    A[8] = 9;
    A[9] = -1;

    int min = A[0]; // берем первый элемент массива
    int max = A[0];

    for (int i = 1; i < N; i++) // перебираем все элементы
    {
        if (A[i] < min)
       {
           min = A[i]; // обновляем минимальный элемент
       }
       if (A[i] > max)
       {
           max = A[i];
       }
    }

    cout << "Минимальный элемент = " << min << endl;
    cout << "Максимальный элемент = " << max << endl;

    return 0;
}
```

2. Сортировка методом пузырька

Рассмотрим задачу сортировки массива. Цель сортировки состоит в том, чтобы переставить элементы массива таким образом, чтобы элементы были упорядочены по возрастанию (или по убыванию). Метод пузырька реализуется в последовательных проходах, при которых сравниваются пары соседних элементов и в случае, если эти элементы имеют не правильный порядок, то они меняются местами.

Рассмотрим пример. Пусть массив (5, 3, 1, 2) нужно отсортировать по возрастанию.

Первый проход (сравниваем элементы, выделенные жирным):

(**5, 3**, 1, 2) меняем местами: (3, 5, 1, 2)

(3, **5, 1**, 2) меняем местами (3, 1, 5, 2)

(3, 1, **5, 2**) меняем местами (3, 1, 2, 5)

Второй проход:

(**3, 1**, 2, 5) меняем местами (1, 3, 2, 5)

(1, **3, 2**, 5) меняем местами (1, 2, 3, 5)

Третий проход:

(**1, 2**, 3, 5) не меняем местами (1, 2, 3, 5).

Пример программы, реализующей сортировку пузырьком.

```
#include <iostream>

using namespace std;

int main()
{
    setlocale(LC_ALL, "Russian");// устанавливаем русскую кодировку

      int N = 10; // количество элементов

      int A[N];

      A[0] = 3;
      A[1] = 2;
      A[2] = -2;
      A[3] = 0;
      A[4] = 10;
      A[5] = 3;
      A[6] = -4;
      A[7] = 5;
```

```
      A[8] = 9;
      A[9] = -1;

      for (int i = 0; i < N - 1; i++)
    {
        cout << i + 1 << " проход" << endl;
        for (int j = 0; j < N - 1 - i; j++)
        {
            cout << "сравниваем: A[" << j << "] и A[" << j + 1 << "]" << endl;
            if (A[j] > A[j + 1])
            {
                // меняем местами
                int c = A[j + 1];
                A[j + 1] = A[j];
                A[j] = c;
            }
        }
    }

    cout << "\nОтсортированный массив:" << endl;
    for (int i = 0; i < N; i++)
    {
        cout << A[i] << "\t";
    }

      return 0;
}
```

3. Поиск подстроки

В разделе, посвященном работами со строками, мы использовали метод s.find() для поиска подстроки. Сейчас мы напишем такую функцию самостоятельно. В этой функции мы будем последовательно искать подстроку в строке, сравнивая все символы подстроки.

```
#include <iostream>
#include <string>

using namespace std;
```

```
int find_substring(string s, string sub);

int main()
{
    setlocale(LC_ALL, "Russian");// устанавливаем русскую кодировку

    string s;
    string sub;

    cout << "Введите строку > ";
    getline(cin, s);
    cout << "\nВведите подстроку > ";
    getline(cin, sub);

    int pos = find_substring(s, sub);

    if (pos < 0)
    {
        cout << "\nПодстрока не найдена!";
    }
    else
    {
        cout << "\nПодстрока найдена на позиции: " << pos;
    }

      return 0;
}

int find_substring(string s, string sub)
{
    int d = s.length() - sub.length(); // вычисляем разницу между длинами
                                       // строки и подстроки

    for (int i = 0; i <= d; i++)
    {
        bool IsFind = true; // условие что подстрока найдена

        // ищем подстроку
        for (int j = i; j < i + sub.length(); j++)
        {
            if (s[j] != sub[j - i])
            {
```

```
                IsFind = false; // если хоть один символ не совпадает,
                                // то подстрока уже не найдена

                break; // оператор выхода из цикла
            }
        }

        if (IsFind)
        {
            return i; // если подстрока была найдена, то возвращаем ее позицию
        }
    }

    return -1; // строка не найдена
}
```

4. Поиск загаданного числа

Напишем программу, которая будет угадывать число от 1 до 100. При этом программа будет спрашивать больше или меньше загаданное число.
Для уменьшения количества вопросов программа будет делить возможный интервал, где есть загаданное число, на половину.

```
#include <iostream>

using namespace std;

int find_substring(string s, string sub);

int main()
{
    setlocale(LC_ALL, "Russian");// устанавливаем русскую кодировку

    int A = 1; // нижняя граница
    int B = 100; // верхняя граница

    while (A != B)
    {
        int C = (A + B) / 2;
```

```
        cout << "\nВы загадади " << C << "? Если да, то введите 0, если Ваше
число меньше, то введите -1, если Ваше число больше, то введите 1 > ";
        int ans;
        cin >> ans;
        if (ans == 0)
        {
            cout << "\nЯ угадала!";
            break;
        }

        if (ans < 0)
        {
            B = C;
        }
        else
        {
            A = C;
        }
    }

      return 0;
}
```