1. **Алгоритмы поиска в массиве**
   1. **Алгоритмы сортировки массива** 
      1. **Бинарный поиск**

#include <iostream>

#include <Windows.h>

using namespace std;

/// <summary>

/// Находит индекс числа num в отсортированном массиве mass с помощью бинарного поиска

/// </summary>

/// <param name="mass">Набор чисел</param>

/// <param name="num">Искомое числа в наборе чисел mass</param>

/// <param name="length">Размер массива mass</param>

int BinarySearch(int\* mass, int num, int length)

{

int left = 0,

right = length - 1;

//пока не вышли за пределы массива

do

{

int mid = (right + left) / 2;//находим средний индекс с заданном интервале [left, right]

if (mass[mid] == num)

return mid;

else if (mass[mid] > num)

right = mid - 1;

else if (mass[mid] < num)

left = mid + 1;

} while (left <= right);

return -1;//указываем, что элемент не найден

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

const int N = 3;

int a[N] = { 1, 2, 3 };

cout << BinarySearch(a, 3, N) << endl;//2

cout << BinarySearch(a, 2, N) << endl;//1

cout << BinarySearch(a, 1, N) << endl;//0

cout << BinarySearch(a, 11, N) << endl;//-1

return 0;

}

* + 1. **Быстрая сортировка**

#include <iostream>

#include <Windows.h>

using namespace std;

/// <summary>

/// Сортирует массив mass с помощью алгортима быстрой сортировки

/// </summary>

/// <param name="mass">Массив из целых чисел</param>

/// <param name="left">Индекс левой границе сортирвоки массива</param>

/// <param name="right">Индекс правой границе сортирвоки массива</param>

void QuickSort(int\* mass, int left, int right)

{

int item = mass[(left + right)/2];//текущий сортируемый элемент (может быть любой, не оязательно брать по середке)

int i = left,

j = right;

do {

//Находим индекс первого элемента, который будет больше item

//в пределах интервала [left, right]

while (mass[i] < item) i++;

//Находим индекс первого элемента, который будет меньше item

//в пределах интервала [left, right]

while (mass[j] > item) j--;

//Если элемент, который больше item расположен

//левее элемента, который меньше item

if (i < j) {

//Меняем эти элементы местами

int copy = mass[i];

mass[i] = mass[j];

mass[j] = copy;

}

else {

i++;

j--;

}

} while (i < j);//повторяем цикл до тех пор, пока оба указателя (i и j) не перескочат через друг друга

if (left < j) QuickSort(mass, left, j);

if (i < right) QuickSort(mass, i, right);

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

const int N = 3;

int a[N] = { 3, 1, 4 };

QuickSort(a, 0, N-1);

return 0;

}

* + 1. **Шейкер сортировка**

#include <iostream>

#include <Windows.h>

using namespace std;

/// <summary>

/// Сортирует массив чисел mass с помощью алгоритма сортировки "Шэйкер"

/// </summary>

/// <param name="mass">Массив чисел</param>

/// <param name="length">Размер массива</param>

void ShakerSort(int\* mass, int length)

{

bool the\_end;//флаг завершения сортировки массива

int left = 0,//указатель на неотсортированный крайний слева элемент

right = length-1;//указатель на неотсортированный крайний справа элемент

do

{

the\_end = true;

//Берем 2 соседних элемента

//шагая с слева - направо

for (int i = left; i < right; i++) {

//Если текущий элемент больше следующего

if (mass[i] > mass[i + 1]){

//Меняем их местами

swap(mass[i + 1], mass[i]);

the\_end = false;

}

}

//Сдвигаем указатель на крайний неотсортированный элемент справа

right--;

//Берем 2 соседних элемента

//шагая с право - налево

for (int i = right; i > left; i--) {

//Если текущий элемент больше следующего

if (mass[i] < mass[i - 1]) {

//Меняем их местами

swap(mass[i - 1], mass[i]);

the\_end = false;

}

}

//Сдвигаем указатель на крайний неотсортированный элемент слева

left++;

} while (!the\_end);

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

const int N = 7;

int a[N] = { 1, 1, 4, 12, 6, 2,3532 };

ShakerSort(a, N);

return 0;

}

* + 1. **Сортировка вставками**

#include <iostream>

#include <Windows.h>

using namespace std;

/// <summary>

/// Сортирует массив чисел mass с помощью метода "Сортироввка вставками"

/// </summary>

/// <param name="mass">Массив чисел</param>

/// <param name="length">Размер массива</param>

void InsertSort(int\* mass, int length)

{

//Пробегаемся по всем элементам для сортировки

for (int i = 0; i < length-1; i++)

{

//Берем самый близжайший к отсортированным элементам item

// item = mass[i + 1];

//Сравниваем item с отсортированными элементами

for (int j = i + 1; j > 0; j--)

{

//Если текущий элемент меньше предыдущего

if (mass[j] < mass[j - 1])

//меняем их местами

swap(mass[j], mass[j - 1]);

else

//Останавливаем сортировку данного элемента

break;

}

}

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

const int N = 8;

int a[N] = { 1,2,3,6,2,6,4,65 };

InsertSort(a, N);

return 0;

}

* + 1. **Обменная сортировка или Сортировка пузырьком**

#include <iostream>

#include <Windows.h>

using namespace std;

/// <summary>

/// Сортирует массив чисел mass с помощью метода "Сортироввка вставками"

/// </summary>

/// <param name="mass">Массив чисел</param>

/// <param name="length">Размер массива</param>

void InsertSort(int\* mass, int length)

{

//Пробегаемся по всем элементам для сортировки

for (int i = 0; i < length-1; i++)

{

//Берем самый близжайший к отсортированным элементам item

// item = mass[i + 1];

//Сравниваем item с отсортированными элементами

for (int j = i + 1; j > 0; j--)

{

//Если текущий элемент меньше предыдущего

if (mass[j] < mass[j - 1])

//меняем их местами

swap(mass[j], mass[j - 1]);

else

//Останавливаем сортировку данного элемента

break;

}

}

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

const int N = 8;

int a[N] = { 1,2,3,6,2,6,4,65 };

InsertSort(a, N);

return 0;

}

* 1. **Алгоритмы удаления и вставки элементов в массив**

**Вставка по определенному индексу** требует сдвига всех элементов, начиная с этого индекса, на одну позицию вправо. Если внутренний массив заполнен, вставка потребует увеличения его размера.

**Удаление элемента по индексу** — операция, обратная вставке. Указанный элемент удаляется, а остальные сдвигаются на одну позицию влево.

* 1. **???**
  2. **???**
  3. **???**
  4. **???**
  5. **Стек в решение задач по вычислению значения выражения, построению постфиксной, префиксной записи выражения. Вычисление значения выражения, представленного в постфиксной или префиксной форме**

Вычисление постфиксного выражения

#include <iostream>

#include <Windows.h>

#include <stack>

using namespace std;

/// <summary>

/// Вычисл¤ет мат. выражение записанное в обратной польской нотации

/// </summary>

/// <param name="expression">Мат. выражение записанна¤ в обратной польской нотации</param>

/// <returns></returns>

double CalculatePostFixNotation(string expression)

{

stack<int> nums;//стек для хранения операндов (чисел)

int length = expression.length();//длина строки

//Пробегаемся по всех символам мат. выражения

for (int i = 0; i < length; i++)

{

//если это число

if(isdigit(expression[i]))

nums.push(expression[i] - '0');//добавляем его в стек

else {

//Если это знак операции

int a = nums.top(); nums.pop();//вытаскиваем первый операнд из стека

int b = nums.top(); nums.pop();//вытаскиваем второй операнд из стека

switch (expression[i])//узнаем какой знак операции нам дан и на основе этого проводим вычисления и добавляем результат в стек

{

case '+':

nums.push(a + b);

break;

case '-':

nums.push(a - b);

break;

case '\*':

nums.push(a \* b);

break;

case '/':

nums.push(a / b);

break;

default:

break;

}

}

}

//последний элеметн в стеке и будет окончательным резултатом вычисленного мат. выражения

return nums.top();

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

string exp;

cout << "Введите мат. выражение в обратной польской нотации: ";

cin >> exp;

cout<<"Результат: "<<CalculatePostFixNotation(exp)<<endl;

return 0;

}

Вычисление префиксного выражения

/// <summary>

/// Вычисляет мат. выражение записанное в прямой польской нотации

/// </summary>

/// <param name="expression">Мат. выражение записанна¤ в прямой польской нотации</param>

/// <returns></returns>

double CalculatePreFixNotation(string expression)

{

stack<double> nums;//стек для хранения операндов (чисел)

double length = expression.length();//длина строки

//Пробегаемся по всех символам мат. выражения

for (int i = length-1; i >= 0; i--)

{

//если это число

if (isdigit(expression[i]))

nums.push(expression[i] - '0');//добавляем его в стек

else {

//Если это знак операции

double a = nums.top(); nums.pop();//вытаскиваем первый операнд из стека

double b = nums.top(); nums.pop();//вытаскиваем второй операнд из стека

switch (expression[i])//узнаем какой знак операции нам дан и на основе этого проводим вычисления и добавляем результат в стек

{

case '+':

nums.push(a + b);

break;

case '-':

nums.push(a - b);

break;

case '\*':

nums.push(a \* b);

break;

case '/':

nums.push(a / b);

break;

default:

break;

}

}

}

//последний элеметн в стеке и будет окончательным резултатом вычисленного мат. выражения

return nums.top();

}