***Решим следующую задачу: вывести в стандартный поток квадраты натуральных чисел от 1 до 100. Следующий код показывает, что эта задача может быть решена в четырех строчках кода и без явного использования циклов:***

Способ из модуля 1: (n вводим вручную)

#include <iostream>

using namespace std;

int main()

{

int n, i;

n = 100;

for (i = 1; i < n; i++)

{

cout << i \* i << endl;

}

}

Способ с использованием алгоритмов: (n автоматически)

#include <vector>

#include <iostream>

#include <numeric> // iota

#include <algorithm> // for\_each

using namespace std;

int main() {

vector<int> v(100);

iota(v.begin(), v.end(), 1); // v = [1, 2, 3, ..., 100]

for\_each(v.begin(), v.end(), [](int& a) {a = a \* a; }); // v = [1, 4, 9, ..., 10000]

for\_each(v.begin(), v.end(), [](int a) {cout << a << ' '; });

return 0;

}

**Пузырьковая сортировка (Bubble sort)**

#include <iostream>

using namespace std;

void bubbleSort(int list[], int listLength)

{

while (listLength--)

{

bool swapped = false;

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

{

if (list[i] > list[i + 1])

{

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

swapped = true;

}

}

if (swapped == false)

break;

}

}

int main()

{

int list[5] = { 3,19,8,0,48 };

cout << "Input array ..." << endl;

for (int i = 0; i < 5; i++)

cout << list[i] << '\t';

cout << endl;

bubbleSort(list, 5);

cout << "Sorted array ..." << endl;

for (int i = 0; i < 5; i++)

cout << list[i] << '\t';

cout << endl;

}

**Сортировка выбором (Selection sort)**

#include <iostream>

using namespace std;

int findSmallestPosition(int list[], int startPosition, int listLength)

{

int smallestPosition = startPosition;

for (int i = startPosition; i < listLength; i++)

{

if (list[i] < list[smallestPosition])

smallestPosition = i;

}

return smallestPosition;

}

void selectionSort(int list[], int listLength)

{

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

{

int smallestPosition = findSmallestPosition(list, i, listLength);

swap(list[i], list[smallestPosition]);

}

return;

}

int main()

{

int list[5] = { 12, 5, 48, 0, 4 };

cout << "Input array ..." << endl;

for (int i = 0; i < 5; i++)

cout << list[i] << '\t';

cout << endl;

selectionSort(list, 5);

cout << "Sorted array ..." << endl;

for (int i = 0; i < 5; i++)

cout << list[i] << '\t';

cout << endl;

}

**Сортировка вставками (Insertion sort)**

#include <iostream>

using namespace std;

void insertionSort(int list[], int listLength)

{

for (int i = 1; i < listLength; i++)

{

int j = i - 1;

while (j >= 0 && list[j] > list[j + 1])

{

swap(list[j], list[j + 1]);

cout << "\ndid";

j--;

}

}

}

int main()

{

int list[8] = { 3,19,8,0,48,4,5,12 };

cout << "Input array ...\n";

for (int i = 0; i < 8; i++)

{

cout << list[i] << "\t";

}

insertionSort(list, 8);

cout << "\n\nSorted array ... \n";

for (int i = 0; i < 8; i++)

{

cout << list[i] << "\t";

}

return 0;

}

**Быстрая сортировка (Quick sort)**

#include <iostream>

using namespace std;

int partition(int list[], int start, int pivot)

{

int i = start;

while (i < pivot)

{

if (list[i] > list[pivot] && i == pivot - 1)

{

swap(list[i], list[pivot]);

pivot--;

}

else if (list[i] > list[pivot])

{

swap(list[pivot - 1], list[pivot]);

swap(list[i], list[pivot]);

pivot--;

}

else i++;

}

return pivot;

}

void quickSort(int list[], int start, int end)

{

if (start < end)

{

int pivot = partition(list, start, end);

quickSort(list, start, pivot - 1);

quickSort(list, pivot + 1, end);

}

}

int main()

{

int list[6] = { 2, 12, 5, 48, 0, 4 };

cout << "Input array ...\n";

for (int i = 0; i < 6; i++)

{

cout << list[i] << "\t";

}

quickSort(list, 0, 6);

cout << "\n\nSorted array ... \n";

for (int i = 0; i < 6; i++)

{

cout << list[i] << "\t";

}

return 0;

}

**Сортировка слиянием (Merge sort)**

#include <iostream>

using namespace std;

void merge(int list[], int start, int end, int mid);

void mergeSort(int list[], int start, int end)

{

int mid;

if (start < end) {

mid = (start + end) / 2;

mergeSort(list, start, mid);

mergeSort(list, mid + 1, end);

merge(list, start, end, mid);

}

}

void merge(int list[], int start, int end, int mid)

{

int mergedList[8];

int i, j, k;

i = start;

k = start;

j = mid + 1;

while (i <= mid && j <= end) {

if (list[i] < list[j]) {

mergedList[k] = list[i];

k++;

i++;

}

else {

mergedList[k] = list[j];

k++;

j++;

}

}

while (i <= mid) {

mergedList[k] = list[i];

k++;

i++;

}

while (j <= end) {

mergedList[k] = list[j];

k++;

j++;

}

for (i = start; i < k; i++) {

list[i] = mergedList[i];

}

}

int main()

{

int list[8] = { 3,19,8,0,48,4,5,12 };

cout << "Input array ...\n";

for (int i = 0; i < 8; i++)

{

cout << list[i] << "\t";

}

mergeSort(list, 0, 7);

cout << "\n\nSorted array ... \n";

for (int i = 0; i < 8; i++)

{

cout << list[i] << "\t";

}

}

**Сортировка Шелла (Shell sort)**

#include <iostream>

using namespace std;

void shellSort(int list[], int listLength)

{

for (int step = listLength / 2; step > 0; step /= 2)

{

for (int i = step; i < listLength; i += 1)

{

int j = i;

while (j >= step && list[j - step] > list[i])

{

swap(list[j], list[j - step]);

j -= step;

cout << "\ndid";

}

}

}

}

int main()

{

int list[8] = { 3,19,8,0,48,4,5,12 };

cout << "Input array ...\n";

for (int i = 0; i < 8; i++)

{

cout << list[i] << "\t";

}

shellSort(list, 8);

cout << "\n\nSorted array ... \n";

for (int i = 0; i < 8; i++)

{

cout << list[i] << "\t";

}

}

**Сортировка кучей(пирамидальная) (Heap sort)**

#include <iostream>

using namespace std;

void heapify(int list[], int listLength, int root)

{

int largest = root;

int l = 2 \* root + 1;

int r = 2 \* root + 2;

if (l < listLength && list[l] > list[largest])

largest = l;

if (r < listLength && list[r] > list[largest])

largest = r;

if (largest != root)

{

swap(list[root], list[largest]);

heapify(list, listLength, largest);

}

}

void heapSort(int list[], int listLength)

{

for (int i = listLength / 2 - 1; i >= 0; i--)

heapify(list, listLength, i);

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

{

swap(list[0], list[i]);

heapify(list, i, 0);

}

}

int main()

{

int list[5] = { 3,19,8,0,48 };

cout << "Input array ..." << endl;

for (int i = 0; i < 5; i++)

cout << list[i] << '\t';

cout << endl;

heapSort(list, 5);

cout << "Sorted array" << endl;

for (int i = 0; i < 5; i++)

cout << list[i] << '\t';

cout << endl;

}

**Перегрузка операторов сравнения и метод sort**

Пример для вектора:

#include <iostream>

#include <vector> // vector

#include <algorithm> // sort

using namespace std;

int main() {

setlocale(0, "");

int n; vector <int> vec;

cout << "Введите количество элементов последовательности: "; cin >> n;

int a;

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

cout << i + 1 << ") "; cin >> a;

vec.push\_back(a);

}

cout << "Вот как выглядит последовательность до: ";

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

cout << vec[i] << " ";

}

sort(vec.begin(), vec.end()); // сортировка

cout << endl << "После сортировки: ";

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

cout << vec[i] << " ";

}

sort(vec.begin() + n / 2, vec.end());

cout << endl << "А вот еще раз: ";

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

cout << vec[i] << " ";

}

system("pause");

return 0;

}

**Функция swap()**

#include <iostream>

void swap(int\* a, int\* b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

int main() {

int x1 = 10, x2 = 20;

std::cout << "x1 = " << x1 << ", x2 = " << x2 << std::endl;

swap(&x1, &x2); // Передаём ссылки

std::cout << "x1 = " << x1 << ", x2 = " << x2 << std::endl;

return 0;

}

**Поиск**

#include <algorithm>

#include <deque>

#include <iostream>

int main() {

std::deque<int> d = { 3, 14, 15, 92, 6 };

// Такой элемент есть, мы его точно найдём

auto iter1 = std::find(d.begin(), d.end(), 15);

// Итераторы дека можно вычитать, напечатается индекс найденного элемента

std::cout << (iter1 - d.begin()) << "\n";

auto start = d.begin();

// К итераторам дека и вектора можно прибавлять целые числа

auto end = start + 3;

// Полуинтервал [start; end) теперь ограничивает подпоследовательность 3, 14, 15

auto iter2 = std::find(start, end, 19);

if (iter2 == end) {

std::cout << "No such element!\n";

}

else {

std::cout << \*iter2 << "\n";

}

// Напечатает No such element

}

Вариация №2:

template <typename Iter, typename Value>

Iter find(Iter first, Iter last, const Value& value) {

while (first != last) {

if (\*first == value) {

return first;

}

++first;

}

return last;

}