**CS**

#include <iostream>

#include <ctime>

using namespace std;

//функция для counting sort

void print(int a[], int sz) {

for (int i = 0; i < sz; i++) cout << a[i] << " "; //вывод массивов

cout << endl;

}

void CountingSort(int arr[], int sz) {

int i, j, k;

int idx = 0;

int min, max;

//нахождение экстремумов массива

min = max = arr[0];

for (i = 1; i < sz; i++) {

min = (arr[i] < min) ? arr[i] : min;

max = (arr[i] > max) ? arr[i] : max;

}

k = max - min + 1;

//подсчет количества элементов в arr, имеющих значение i

int\* B = new int[k];

for (i = 0; i < k; i++) B[i] = 0;

//сортировка

for (i = 0; i < sz; i++) B[arr[i] - min]++;

for (i = min; i <= max; i++)

for (j = 0; j < B[i - min]; j++) arr[idx++] = i;

print(arr, sz);

delete[] B;

}

int main() {

setlocale(LC\_ALL, "ru");

int a[] = { 5,9,3,9,10,9,2,4,13,10,11,12,23,34,45 };

const size\_t sz = sizeof(a) / sizeof(a[0]);

cout << "вывод изначального массива\n";

print(a, sz);//вывод изначального массива

cout << "----------------------\n";

cout << "вывод отсортированного массива\n";

CountingSort(a, sz);//вывод отсортированного массива

cout << "----------------------\n";

cout << "время выполнения работы = " << clock() / 1000.0 << " сек." << endl;

}

**QS**

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <ctime>

using namespace std;

#define SIZE 20

// Функция быстрой сортировки

void quickSort(int\* numbers, int left, int right)

{

int pivot; // разрешающий элемент

int l\_hold = left; //левая граница

int r\_hold = right; // правая граница

pivot = numbers[left];

while (left < right) // пока границы не сомкнутся

{

while ((numbers[right] >= pivot) && (left < right))

right--; // сдвигаем правую границу пока элемент [right] больше [pivot]

if (left != right) // если границы не сомкнулись

{

numbers[left] = numbers[right]; // перемещаем элемент [right] на место разрешающего

left++; // сдвигаем левую границу вправо

}

while ((numbers[left] <= pivot) && (left < right))

left++; // сдвигаем левую границу пока элемент [left] меньше [pivot]

if (left != right) // если границы не сомкнулись

{

numbers[right] = numbers[left]; // перемещаем элемент [left] на место [right]

right--; // сдвигаем правую границу вправо

}

}

numbers[left] = pivot; // ставим разрешающий элемент на место

pivot = left;

left = l\_hold;

right = r\_hold;

if (left < pivot) // Рекурсивно вызываем сортировку для левой и правой части массива

quickSort(numbers, left, pivot - 1);

if (right > pivot)

quickSort(numbers, pivot + 1, right);

}

int main() {

setlocale(LC\_ALL, "ru");

int a[SIZE];

// Заполнение массива случайными числами

cout << "Вывод элементов массива до сортировки:\n";

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

a[i] = rand() % 201 - 100;

// Вывод элементов массива до сортировки

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

printf("%4d ", a[i]);

printf("\n");

quickSort(a, 0, SIZE - 1); // вызов функции сортировки

cout << " Вывод элементов массива после сортировки:\n";

// Вывод элементов массива после сортировки

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

printf("%4d ", a[i]);

printf("\n");

cout << "время выполнения работы = " << clock() / 1000.0 << " сек." << endl;

getchar();

return 0;

}

**RS**

#include <algorithm>

#include <iostream>

#include <iterator>

#include <ctime>

using namespace std;

// класс с условиями для сортировки

class radix\_test{

const int bit; // битовая позиция [0..31] для проверки

public:

radix\_test(int offset) : bit(offset) {} // конструктор

bool operator()(int value) const // оператор вызова функции

{

if (bit == 31) // знаковый бит

return value < 0; // отрицательное int в левую секцию

else

return !(value & (1 << bit)); // 0 бит в левый раздел

}

};

// Сортировка по наименьшей значащей цифре

void lsd\_radix\_sort(int\* first, int\* last){

for (int lsb = 0; lsb < 32; ++lsb) // младший бит

{

stable\_partition(first, last, radix\_test(lsb));

}

}

// Сортировка по основанию счисления по старшим разрядам (рекурсивная)

void msd\_radix\_sort(int\* first, int\* last, int msb = 31){

if (first != last && msb >= 0)

{

int\* mid = std::partition(first, last, radix\_test(msb));

msb--; // уменьшить самый старший бит

msd\_radix\_sort(first, mid, msb); // отсортировать левый раздел

msd\_radix\_sort(mid, last, msb); // отсортировать правый раздел

}

}

// вывод массивов

int main(){

setlocale(LC\_ALL, "ru");

cout << "исходный массив\n";

int data[8] = { 170, 45, 75, -90, -802, 24, 2, 66 };

for (size\_t i = 0; i < 8; i++)

{

cout << data[i] << " ";

}

cout << endl << endl;

cout << "отсортированный массив\n";

lsd\_radix\_sort(data, data + 8);

copy(data, data + 8, std::ostream\_iterator<int>(cout, " "));

cout << endl << endl;

cout << "время выполнения работы = " << clock() / 1000.0 << " сек." << endl;

cout << endl << endl;

return 0;

}