#include <iostream>

#include <type\_traits>

#include <functional>

#include <ctime>

#include <concepts>

class Number

{

private:

int number;

public:

Number():

number(0){}

Number(const int number) :

number(number) {}

int get() const

{

return number;

}

void set(int number)

{

this->number = number;

}

bool operator>(const Number &number) const

{

return this->number > number.number;

}

};

std::ostream& operator<<(std::ostream& o, const Number &num)

{

o << num.get();

return o;

}

template<class T>

void Swap(T& a, T& b)

{

T c = a;

a = b;

b = c;

}

template<typename T>

concept numberable = requires {std::convertible\_to<T, const size\_t>; };

//функция fun должна возвразащать тру. если первый боьше второго

template<class T, numberable K>

void Selection\_sort(T\* arr, K size, const std::function<bool(const T&, const T&)> fun = std::greater<T>())

{

int minimum;

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

{

minimum = i;

for (int j = i + 1; j < size; j++)

{

if (fun(arr[minimum], arr[j]))

minimum = j;

}

Swap(arr[i], arr[minimum]);

}

}/\*суть сортировки заключается в поиске минимального и обмене текущего на самого минимального из оставшихся\*/

template<class T, numberable K>

void Quick\_sort(T\* arr, int begin, K end, const std::function<bool(const T&, const T&)> fun = std::greater<T>())

{

int left = begin;

int right = end;

T x = arr[(begin + end) / 2];

do {

while (fun(x, arr[left]))

{

left++;

}

while (fun(arr[right], x))

{

right--;

}

if (left <= right)

{

x = arr[left];

Swap(arr[left], arr[right]);

left++;

right--;

}

} while (left <= right);

if (begin < right)

{

Quick\_sort(arr, begin, right, fun);

}

if (left < end)

{

Quick\_sort(arr, left, end, fun);

}

}//начальный выбор опорного элемента на середину. дальше те, кто меньше слева, те, кто больше справава методом swap. выбор опорного элемента становится arr[left]

template<class T, typename K = size\_t>

void Randomize(T\* arr, K size)

{

for (auto i = (decltype(size))0; i < size; ++i)

{

arr[i] = rand() % 10;

}

}

template<class T, typename K = size\_t>

void Output(T\* arr, K size)

{

for (auto i = (decltype(size))0; i < size; ++i)

{

std::cout << arr[i] << ' ';

}

std::cout << std::endl;

}

int main()

{

srand((unsigned)time(NULL));

size\_t size = 5;

Number\* numbers = new Number[size];

Randomize(numbers, size);

Output(numbers, size);

Selection\_sort(numbers, size);

Output(numbers, size);

Randomize(numbers, size);

Output(numbers, size);

Quick\_sort(numbers, 0, size - 1);

Output(numbers, size);

delete[] numbers;

return 0;

}