**Шараев Владислав Николаевич 020602**

**Лабораторная работа №14**

**Вариант 14**

***Условие:*** Найти среднее значение всех элементов очереди и удалить все элементы, которые меньше среднего значения.

***Текст программы:***

#define \_CRT\_SECURE\_NO\_WARNINGS // Предпроцессор для отключения предупреждений при работе со scanf и printf

#include <iostream>

#include <cstdio>

using namespace std;

struct Queue { // Структура элемента очереди

Queue\* child = NULL; // Указатель на родительский элемент

int value; // Значение элемента

};

void AddItemQueue(Queue\*\*, Queue\*\*, int); // Добавление элемента в очередь

void FindUnderAverageQueue(Queue\*\*, double); // Поиск и удаление элементов очереди, которые меньше среднего значения

void DeleteNextQueue(Queue\*); // Удаление следующего за этим элемента очереди

void PrintQueue(Queue\*); // Вывести все элементы очереди на экран

int main()

{

setlocale(LC\_ALL, "Russian"); // Поддержка кириллицы

Queue\* begin;

Queue\* end; // Указатели на начало и конец очереди

begin = end = new Queue;

int n;

double average = 0;

printf("Введите количество элементов и элементы очереди: "); // Чтение количества элементов

scanf("%d", &n);

for (int i = 0; i < n; i++) { // Чтение элементов очереди

int temp;

scanf("%d", &temp);

average += temp; // Суммируем все элементы очереди

AddItemQueue(&begin, &end, temp); // Добавление элемента в очередь

}

average /= n; // Находим среднее значение всех элементов

printf("Содержимое очереди до удаления: \n");

PrintQueue(begin->child); // Вывод содержания основной очереди на экран

printf("Среднее значение элементов: %lf\n", average);

FindUnderAverageQueue(&begin, average); // Удаление всех элементов, которые меньше среднего значения

printf("Содержимое очереди после удаления: \n");

PrintQueue(begin->child); // Вывод содержания очереди после удаления на экран

return 0;

}

void AddItemQueue(Queue\*\* begin, Queue\*\* end, int val) { // Добавление элемента в очередь

Queue\* item = new Queue; // Создаем новый элемент

item->value = val; // Присваимваем значение

(\*end)->child = item; // Присваиваем в конец очереди

\*end = item;

}

void PrintQueue(Queue\* item) { // Вывести все элементы очереди на экран

if (item != 0) {

printf("%d ", item->value);

PrintQueue(item->child); // Рекурсивно выводим все элементы очереди

}

else {

printf("\n"); // Переход на новую строку после вывода всех элементов

}

}

void DeleteNextQueue(Queue\* item) { // Удаление следующего за этим элемента

if (item->child == 0) return; // Если это конец, то ничего не делаем

Queue\* temp = item->child; // Запоминаем адрес удаляемого элемента

item->child = item->child->child;

delete temp; // Очищаем память

}

void FindUnderAverageQueue(Queue\*\* item, double aver) { // Поиск и удаление элементов, которые меньше среднего значения

if ((\*item)->child == 0) return; // Если это последний элемент, то ничего не делаем

if ((\*item)->child->value < aver) { // Если дочерний элемент меньше среднего, то удаляем его и смотрим новый дочерний элемент

DeleteNextQueue(\*item);

FindUnderAverageQueue(item, aver);

}

else { // Иначе переходим к следующему элементу

FindUnderAverageQueue(&((\*item)->child), aver);

}

}

***Результат работы программы:***



