|  |  |
| --- | --- |
| *voenmeh* | МИНОБРНАУКИ РОССИИ  федеральное государственное бюджетное образовательное учреждение  высшего образования  **«Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова»**  **(БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова»)** |
| БГТУ.СМК-Ф-4.2-К5-02 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Факультет |  | О |  | Естественнонаучный |
|  |  | шифр |  | наименование |
| Кафедра |  | О7 |  | Информационные системы и программная инженерия |
|  |  | шифр |  | наименование |
| Дисциплина |  | Информационные технологии и программирование | | |

|  |
| --- |
| Практическая работа №3 |
| Шаблоны |
| Вариант №4 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент группы | | | |  | И924Б |
| Антонов Н. М. | | | | | |
| Фамилия И.О. | | | | | |
| **Преподаватель:** | | | | | |
| Гладевич А. А. | |  |  | | |
| Фамилия И.О. Подпись | | | | | |
| Оценка |  | | | |  |
| «\_\_\_\_\_» |  | | | | 2023г. |

САНКТ-ПЕТЕРБУРГ

2023 г.

**Цель работы**

Научиться создавать шаблоны функций и шаблоны классов для работы с любыми типами данных без переписывания кода программы.

**Постановка задачи**

Написать шаблон функции, выполняющей указанные в вариативной части задания действия. Написать программу тестирования шаблонных функций, созданных на основе этого шаблона, с аргументами указанных типов. Разработать шаблон класса, описывающий указанный в вариативной части задания абстрактный тип данных, и написать программу тестирования объектов двух шаблонных классов. Выбор тестируемого метода должен осуществляться с помощью меню.

**Уровень сложности** – Средний.

Заданный АТД реализовать с помощью указанной структуры хранения.

**Вариант 25**

Типы аргументов: unsigned int, int.

1. Поиск наименьшего неотрицательного элемента массива.
2. АТД Очередь. Структура хранения векторная.

**Текст программы:**

Файл *queue.h*

#include <iostream>

using namespace std;

template<typename T>

class Queue {

private:

T\* data;

int front;

int rear;

int capacity;

public:

Queue(int size) {

data = new T[size];

front = rear = -1;

capacity = size;

}

Queue& operator=(const Queue& other) {

if (this == &other) {

return \*this;

}

delete[] data;

capacity = other.capacity;

front = other.front;

rear = other.rear;

data = new T[capacity];

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

data[i] = other.data[i];

}

return \*this;

}

Queue(const Queue& other) {

capacity = other.capacity;

front = other.front;

rear = other.rear;

data = new T[capacity];

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

data[i] = other.data[i];

}

}

~Queue() {

delete[] data;

}

bool isEmpty() {

return front == -1;

}

bool isFull() {

return (rear + 1) % capacity == front;

}

void enqueue(T item) {

rear = (rear + 1) % capacity;

data[rear] = item;

if (front == -1) {

front = rear;

}

}

T dequeue() {

T item;

item = data[front];

if (front == rear) {

front = rear = -1;

}

else {

front = (front + 1) % capacity;

}

return item;

}

T peek() {

return data[front];

}

};

Файл *template.h*

#include <iostream>

#include <cctype>

using namespace std;

//функция поиска наиманьшего элемента среди unsigned char

template <typename T>

void findminchar(T arr[], int size) {

T min = 10000000;

T count = 0;

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

if (!(isalpha(arr[i] % min))) {

if (arr[i] < min && arr[i] > 0) min = arr[i];

}

else count++;

}

if (count == size) cout << endl << "There're no numbers in array" << endl;

else if (min == 10000000) cout << endl << "There're no numbers more than 0" << endl;

else cout << endl << "Minimal element more than zero = " << min << endl;

}

Файл *program.cpp*

#include <iostream>

#include "queue.h"

#include "template.h"

using namespace std;

int main() {

const int size = 6;

int s = 0, x = 0, z = 0, size1 = 1, size2 = 1;

int\* arr1;

unsigned char\* arr2;

Queue<int> queue1(size), testint(size);

Queue<unsigned char> queue2(size), testchar(size);

char menu, nemu, unem, men, nem, y, f, h;

do {

cout << "---------------------------------------------------------------" << endl;

cout << "1. Finding minimal elements |" << endl;

cout << "2. Test queue |" << endl;

cout << "0. Exit |" << endl;

cout << "---------------------------------------------------------------" << endl;

cin >> menu;

switch (menu) {

case '1':

cout << "\x1B[2J\x1B[H";

do {

cout << "---------------------------------------------------------------" << endl;

cout << "1. Find minimal element in int array |" << endl;

cout << "2. Find minimal element in unsigned char array |" << endl;

cout << "0. Back |" << endl;

cout << "---------------------------------------------------------------" << endl;

cin >> nemu;

switch (nemu) {

case '1':

cout << "Firstly enter size of arr1: ";

cin >> size1;

arr1 = new int[size1];

for (int i = 0; i < size1; i++) {

cout << "Enter " << i + 1 << " element " << endl;

cin >> x;

arr1[i] = x;

}

cout << "---------------------------------------------------------------" << endl;

cout << "Int array: " << endl;

for (int i = 0; i < size1; i++) {

cout << arr1[i] << " ";

}

findminchar(arr1, size1);

delete[] arr1;

break;

case '2':

cout << "Firstly enter size of arr2: ";

cin >> size2;

arr2 = new unsigned char[size2];

for (int i = 0; i < size2; i++) {

cout << "Enter " << i + 1 << " element " << endl;

cin >> y;

arr2[i] = y;

}

cout << "---------------------------------------------------------------" << endl;

cout << "Unsigned char array: " << endl;

for (int i = 0; i < size2; i++) {

cout << "Enter " << i + 1 << " element " << endl;

cin >> x;

arr2[i] = x;

}

findminchar(arr2, size2);

break;

delete[] arr2;

case '0':

cout << "\x1B[2J\x1B[H";

break;

default:

cout << "INVALID CHOICE TRY AGAIN" << endl;

}

} while (nemu != '0');

break;

case '2':

cout << "\x1B[2J\x1B[H";

do {

cout << "---------------------------------------------------------------" << endl;

cout << "1. Test int queue |" << endl;

cout << "2. Test unsigned char queue |" << endl;

cout << "0. Back |" << endl;

cout << "---------------------------------------------------------------" << endl;

cin >> men;

switch (men) {

case '1':

cout << "Enter " << size << " elements" << endl << endl;

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

cout << "Enter " << i + 1 << " element: ";

cin >> x;

queue1.enqueue(x);

}

testint = queue1;

cout << "Queue after push: ";

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

if (testint.isEmpty() == 1) break;

else {

cout << testint.dequeue() << " ";

}

}

cout << endl;

testint = queue1;

do {

cout << "---------------------------------------------------------------" << endl;

cout << "1. Pop element from rear |" << endl;

cout << "2. Add element |" << endl;

cout << "0. Back |" << endl;

cout << "---------------------------------------------------------------" << endl;

cin >> nem;

switch (nem) {

case '1':

if (queue1.isEmpty()) {

cout << "Queue is empty:(" << endl;

break;

}

else {

cout << "Your array before: ";

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

if (testint.isEmpty()) break;

else {

cout << testint.dequeue() << " ";

}

}

cout << endl;

z = queue1.peek();

cout << "Element will be popped: " << z << endl;

queue1.dequeue();

testint = queue1;

cout << "Yor array after: ";

if (testint.isEmpty()) cout << "Queue will be empty";

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

if (testint.isEmpty()) break;

else {

cout << testint.dequeue() << " ";

}

}

testint = queue1;

cout << endl;

break;

}

case '2':

testint = queue1;

if (queue1.isFull()) {

cout << "Queue is full:(" << endl;

break;

}

cout << "Your array before: ";

if (queue1.isEmpty()) cout << "Queue is empty :(";

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

if (testint.isFull() || testint.isEmpty()) break;

else {

cout << testint.dequeue() << " ";

}

}

cout << endl << "Enter new element ";

cin >> x;

queue1.enqueue(x);

testint = queue1;

cout << "Your array after: ";

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

if (testint.isEmpty()) break;

else {

cout << testint.dequeue() << " ";

}

}

testint = queue1;

cout << endl;

break;

case '0':

cout << "\x1B[2J\x1B[H";

break;

default:

cout << "INVALID CHOICE TRY AGAIN" << endl;

}

} while (nem != '0'); break;

case '2':

cout << "Enter " << size << " elements for unsigned char array" << endl;

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

cout << "Enter " << i + 1 << " element: ";

cin >> y;

queue2.enqueue(y);

}

testchar = queue2;

cout << "Queue after push: ";

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

if (testchar.isEmpty()) break;

else {

cout << testchar.dequeue() << " ";

}

}

cout << endl;

testchar = queue2;

do {

cout << "---------------------------------------------------------------" << endl;

cout << "1. Pop element from rear |" << endl;

cout << "2. Add element |" << endl;

cout << "0. Back |" << endl;

cout << "---------------------------------------------------------------" << endl;

cin >> unem;

switch (unem) {

case '1':

if (queue2.isEmpty()) {

cout << "Queue is empty:(" << endl;

break;

}

else {

cout << "Your array before: ";

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

if (testchar.isEmpty()) break;

else {

cout << testchar.dequeue() << " ";

}

}

cout << endl;

f = queue2.peek();

cout << "Element will be popped: " << f << endl;

queue2.dequeue();

testchar = queue2;

cout << "Yor array after: ";

if (testchar.isEmpty()) cout << "Queue will be empty:(";

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

if (testchar.isEmpty()) break;

else {

cout << testchar.dequeue() << " ";

}

}

cout << endl;

testchar = queue2;

}

break;

case '2':

testchar = queue2;

if (queue2.isFull()) {

cout << "Queue is full:(" << endl;

}

else {

cout << "Your array before: ";

if (queue2.isEmpty()) cout << "Queue is empty :(";

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

if (testchar.isFull() || testchar.isEmpty()) break;

else {

cout << testchar.dequeue() << " ";

}

}

cout << endl << "Enter new element ";

cin >> y;

queue2.enqueue(y);

testchar = queue2;

cout << "Your array after: ";

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

if (testchar.isEmpty()) break;

else {

cout << testchar.dequeue() << " ";

}

}

testchar = queue2;

cout << endl;

}break;

case '0':

cout << "\x1B[2J\x1B[H";

break;

default:

cout << "INVALID CHOICE TRY AGAIN" << endl;

}

} while (unem != '0'); break;

case '0':

cout << "\x1B[2J\x1B[H";

break;

default:

cout << "INVALID CHOICE TRY AGAIN" << endl;

}

} while (men != '0'); break;

case '0':

cout << "\x1B[2J\x1B[H";

cout << "Exiting program....";

break;

default:

cout << "INVALID CHOICE TRY AGAIN" << endl;

}

} while (menu != '0');

return 0;

}

**Результат работы программы:**

При запуске программы появляется меню, что показано на рисунке 1.

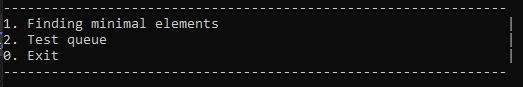


Рисунок 1 − Меню программы

При нажатии на «1» в главном меню можно найти минимальный неотрицательный элемент в двух массивах типа int и char. Появляется меню выбора между int и unsigned int массивами, что показано на рисунке 2.

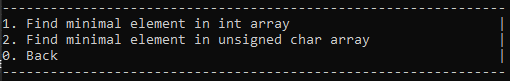


Рисунок 2 − Меню, появляющееся после нажатия «1» в главном меню

При выборе «1» вводится размер массива, затем элементы в нем с клавиатуры, после чего выводится массив и минимальный неотрицательный элемент в нем, что показано на рисунке 3.

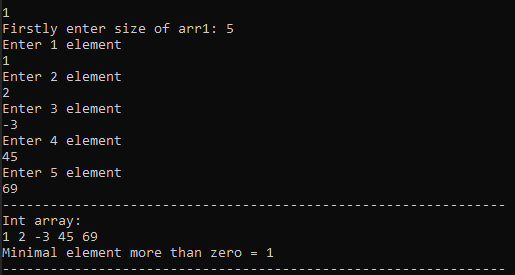


Рисунок 3 – Ввод массива типа int с клавиатуры

При вводе некорректного значения размера массива выдает ошибку, рисунок 4.

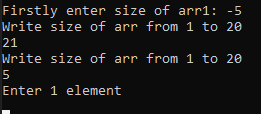


Рисунок 4 – ввод некорректного значения размера массива

При выборе «2» вводится размер массива, затем элементы в нем с клавиатуры, после чего выводится массив и минимальный неотрицательный элемент в нем, что показано на рисунке 5.

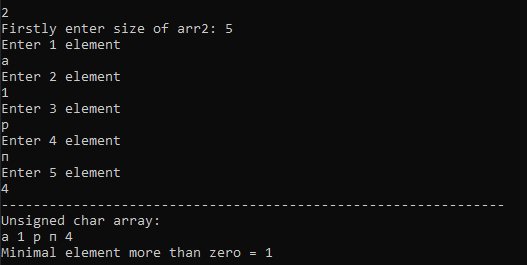


Рисунок 5 – Ввод массива типа unsigned char с клавиатуры

При вводе некорректного значения массива выводится ошибка, что показано на рисунке 6.

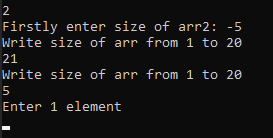


Рисунок 6 – Ввод некорректного значения размера массива

При выборе несуществующего пункта меню выведется сообщение об ошибке, что показано на рисунке 7.



Рисунок 7 − Выбор несуществующего пункта меню

При выборе «0» можно вернуться в главное меню.

При нажатии «2» в главном меню можно протестировать работу очереди. Появится выбор между очередью для int элементов и очередью для unsigned char элементов, что показано на рисунке 8.

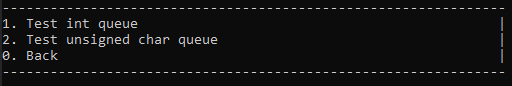


Рисунок 8 − Меню, появляющееся после нажатия «2» в главном меню

При выборе «1» можно проверить, что очередь работает верно. С клавиатуры нужно ввести значения очереди. Затем будет выведена очередь, что показано на рисунке 9.

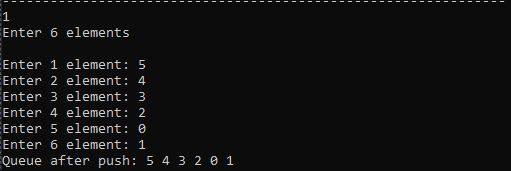


Рисунок 7 − Тестирование очереди для int элементов

Далее появляется меню действий над очередь, что показано на рисунке 8.

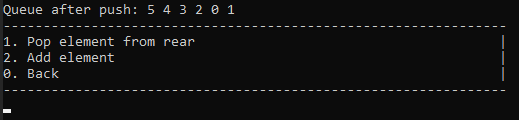


Рисунок 8 – Меню действий над очередью

При выборе «1» можно удалить элемент из очереди, затем выведется очередь до удаления, удаленный элемент и очередь после удаления, рисунок 9.

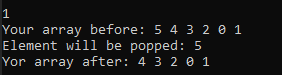


Рисунок 9 – удаление элемента из очереди

При выборе «2» можно добавить элемент в очередь, введя его с клавиатуры, рисунок 10.

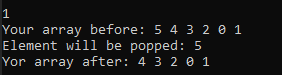


Рисунок 10 – добавление элемента в очередь

При выборе несуществующего пункта меню выводится ошибка, рисунок 11.



Рисунок 11 – выбор несуществующего пункта меню

При выборе «0» происходит выход в подменю тестирования очереди.

При выборе «2» можно проверить, что очередь работает верно. С клавиатуры нужно ввести значения очереди. Затем будет выведена очередь, что показано на рисунке 12.

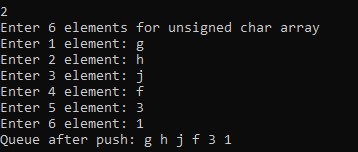


Рисунок 12 − Тестирование очереди для unsigned char элементов

Далее появляется меню действий над очередь, что показано на рисунке 13.

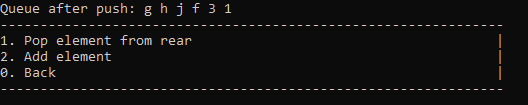


Рисунок 13 – Меню выбора действий над очередью

При выборе «1» удаляется элемент из очереди, затем выводится очередь до удаления, удаленный элемент, очередь после удаления, что показано на рисунке 14.

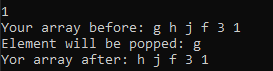


Рисунок 14 – Удаление элемента из очереди

При выборе «2» можно добавить элемент в очередь, рисунок 15.

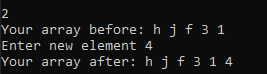


Рисунок 15 – Добавление элемента в очередь

При выборе несуществующего пункта меню выводится ошибка, что показано на рисунке 16.



Рисунок 16 – Выбор несуществующего пункта меню

При выборе «0» происходит выход в подменю.

При выборе несуществующего пункта главного меню выведется сообщение об ошибке, что показано на рисунке 17.



Рисунок 17 − Выбор несуществующего пункта меню

При нажатии на «0» в главном меню, работа программы завершится, что показано на рисунке 18.



Рисунок 18 − Завершение программы