Балтийский государственный технический университет  
«ВОЕНМЕХ» им. Д. Ф. Устинова

Кафедра И5 «Информационные системы и программная инженерия»

**Лабораторная работа №3**по дисциплине «Программирование на языке высокого уровня»  
по теме «Шаблоны»

Выполнил:  
Студент Альков В. С.  
Группа И407Б  
  
Преподаватель:  
Кимсанбаев К. А.

Санкт-Петербург  
2021 г.

Задача: 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 — повышенный. Создать требуемый АТД с помощью двух структур хранения: векторной и списковой, реализацию оформить в виде шаблонов классов с единым интерфейсом.

Типы аргументов int и float.

1. Поиск максимального отрицательного элемента в массиве.

2. АТД Очередь. Структура хранения – циклический массив.

**Задание 1**

#include <iostream>

/\*шаблон функции\*/

template<class T, class Tnumber>

T maxNegative(T \*arr, Tnumber n)

{

T max;

Tnumber i;

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

if(arr[i] < 0)

{

max = arr[i++];

break;

};

for(; i<n; i++)

if(arr[i]>max && arr[i]<0)

max = arr[i];

return max;

}

int main()

{

int \*arr1, choice, count;

float \*arr2;

/\*реализация меню\*/

setlocale(LC\_ALL, "rus");

std::cout << "1. Тип int\n";

std::cout << "2. Тип float\n";

std::cin >> choice;

std::cout << "Введите кол-во элементов массива: ";

std::cin >> count;

if(choice!=1 && choice != 2 || count<0)

{

std::cout<<"Неправильный ввод\n";

return 0;

};

if(choice == 1)

{

arr1 = new int[count];

std::cout<<"Введите "<<count<<" элементов: ";

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

std::cin>>arr1[i];

std::cout<<"Если 0, значит нет отрицательных\nНаибольший из отрицательных: " << maxNegative(arr1, count);

delete[] arr1;

}

else

{

arr2 = new float[count];

std::cout<<"Введите "<<count<<" элементов: ";

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

std::cin>>arr2[i];

std::cout<<"Если 0, значит нет отрицательных\nНаибольший из отрицательных: " << maxNegative(arr2, count);

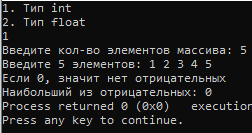
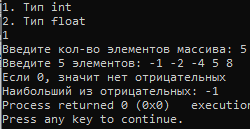
delete[] arr2;

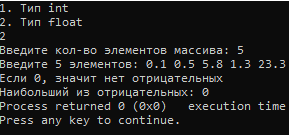
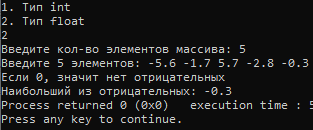
};

return 0;

}

**Результат работы программы**

**Задание 2**

**Класс векторной реализации очереди**

#include <iostream>

#include <cstdlib>

template<class T> class Queue

{

/\*индексы головы, хвоста очереди\*/

int head, tail;

/\*максимальная размер очереди\*/

int maxLength;

/\*массив для хранения очереди\*/

T \*data;

public:

/\*конструктор\*/

Queue(int n=10);

/\*деструктор\*/

~Queue();

/\*проверка на пустоту\*/

bool isEmpty(void);

/\*проверка на заполненность\*/

bool isFull(void);

/\*получение первого элемента\*/

T Front(void);

/\*добавление в очередь\*/

bool EnQueue(T x);

/\*извлечение из очереди\*/

T DeQueue(void);

};

template<class T>

Queue<T>::Queue(int n)

{

maxLength = n;

data=new T[maxLength];

head=0;

tail=maxLength-1;

}

template<class T>

Queue<T>::~Queue()

{

delete[] data;

}

template<class T>

bool Queue<T>::isEmpty(void)

{

return (tail+1)%maxLength == head;

}

template<class T>

bool Queue<T>::isFull(void)

{

return (tail+2)%maxLength == head;

}

template<class T>

T Queue<T>::Front (void)

{

return data[head];

}

template<class T>

bool Queue<T>::EnQueue (T x)

{

if (this->isFull()) return false;

tail = (tail+1)%maxLength;

data[tail] = x;

return true;

}

template<class T>

T Queue<T>::DeQueue (void)

{

int temp = head;

head = (head+1)%maxLength;

return data[temp];

}

**Класс связной реализации очереди**

#include <iostream>

#include <cstdlib>

template<class T> class Queue2

{

/\*структурный тип для элемента связной очереди\*/

struct element

{

T data;

element \* next;

} \*head, \*tail; //индексы головы и хвоста

public:

/\*конструктор\*/

Queue2 () {head=tail=NULL;}

/\*деструктор\*/

~Queue2 ();

/\*проверка на пустоту\*/

bool isEmpty (void);

/\*проверка на заполненность\*/

bool isFull (void);

/\*получение первого элемента\*/

T Front (void);

/\*добавление в очередь\*/

bool EnQueue (T x);

/\*извлечение из очереди\*/

T DeQueue (void);

};

template<class T>

bool Queue2<T>::isEmpty(void)

{

return head==NULL;

}

template<class T>

bool Queue2<T>::isFull(void)

{

element \*temp = new (std::nothrow) element;

if (temp==NULL) return 1;

delete temp;

return 0;

}

template<class T>

T Queue2<T>::Front (void)

{

return head->data;

}

template<class T>

bool Queue2<T>::EnQueue (T x)

{

element \* temp = new (std::nothrow) element;

if (temp==NULL) return 1;

temp->data = x;

temp->next = NULL;

if (head==NULL)

head = tail = temp;

else

{

tail->next = temp;

tail = tail->next;

}

return 1;

}

template<class T>

T Queue2<T>::DeQueue (void)

{

T temp = head->data;

element \* tmp = head;

head = head->next;

delete tmp;

return temp;

}

template<class T>

Queue2<T>::~Queue2 ()

{

element \* temp = head;

while (head)

{

temp = head;

head = head->next;

delete temp;

}

}

**Реализация меню**

#include <iostream>

#include "queue1.cpp"

#include "queue2.cpp"

/\*шаблон функции меню, принимает указатель на Queue, Queue2\*/

template<template<typename> class T, typename C>

void menu(T<C> \*object)

{

setlocale(LC\_ALL, "rus");

int menu;

C num;

do

{

system("cls");

std::cout<<"1. Добавить в очередь.\n";

std::cout<<"2. Неразрушающее чтение\n";

std::cout<<"3. Убрать из очереди\n";

std::cout<<"4. Проверить на пустоту\n";

std::cout<<"5. Проверить на заполненность\n";

std::cout<<"6. Выйти\n";

std::cin>>menu;

getchar();

switch(menu)

{

case 1: if(object->isFull())

std::cout<<"Очередь заполнена\n";

else

{

std::cin>>num;

getchar();

object->EnQueue(num);

};

break;

case 2: if(object->isEmpty())

std::cout<<"Очередь пуста\n";

else

std::cout<<object->Front();

break;

case 3: if(object->isEmpty())

std::cout<<"Очередь пуста\n";

else

std::cout<<object->DeQueue();

break;

case 4: std::cout<<object->isEmpty();

break;

case 5: std::cout<<object->isFull();

break;

case 6: break;

default: std::cout<<"Неправильный ввод\n";

break;

}; getchar();

}

while(menu!=6);

};

int main()

{

system("chcp 1251");

system("cls");

int choice, choice2;

std::cout << "1. Векторная очередь\n";

std::cout << "2. Связная очередь\n";

std::cin >> choice;

std::cout << "1. Тип int\n";

std::cout << "2. Тип float\n";

std::cin >> choice2;

/\*проверяем введенные данные\*/

if(choice != 1 && choice != 2 || choice2 != 1 && choice2 != 2)

{

std::cout << "Неправильный ввод";

return 0;

};

/\*определяем структуру, выбранную пользователем\*/

if (choice == 1)

{

if(choice2 == 1)

{

Queue<int> a;

menu(&a);

}

else

{

Queue<float> a;

menu(&a);

}

}

else

{

if(choice2 == 1)

{

Queue2<int> a;

menu(&a);

}

else

{

Queue2<float> a;

menu(&a);

}

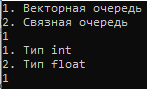
};

return 0;

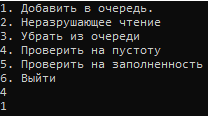
}

**Результат работы программы**

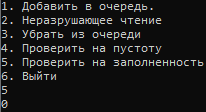
**Векторная очередь, int**



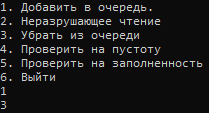
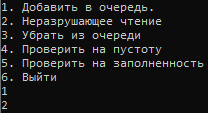
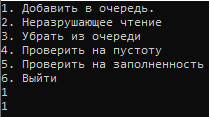
**4. Проверить на пустоту**

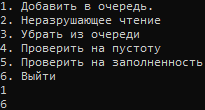
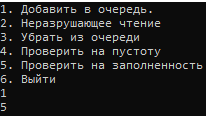
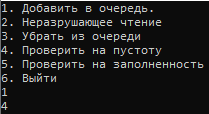


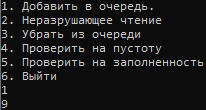
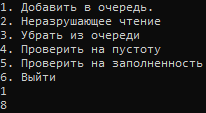
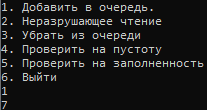
**5. Проверить на заполненность**

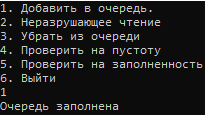


1. **Добавить в очередь.**

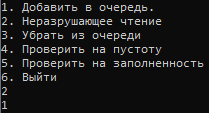




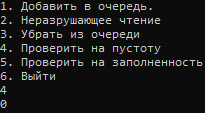




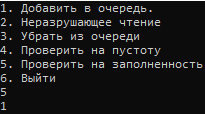
**2. Неразрушающее чтение**



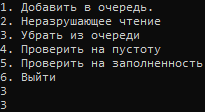
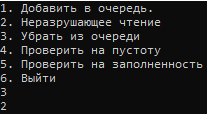
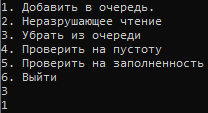
**4. Проверить на пустоту**

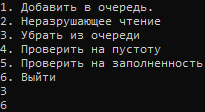
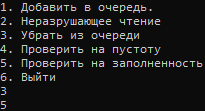
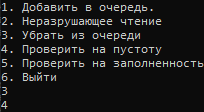


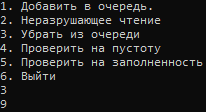
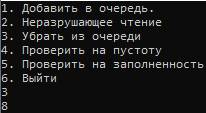
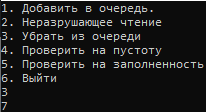
**5. Проверить на заполненность**

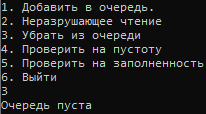


**3. Убрать из очереди**

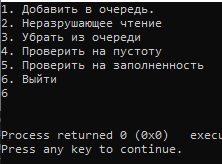




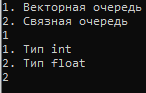




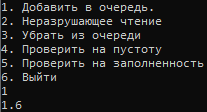
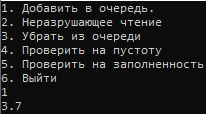
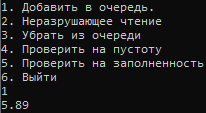
**6. Выйти**



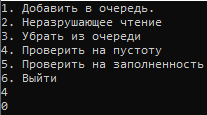
**Векторная очередь, flaot**



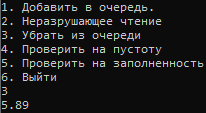
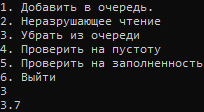
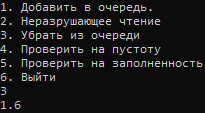
**1. Добавить в очередь.**

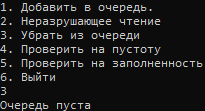
  

**4. Проверить на пустоту**



**3. Убрать из очереди**





**6. Выйти**

