Министерство Образования Республики Беларусь
УО Брестский Государственный Технический Университет
Кафедра интеллектуальных информационных технологий

Лабораторная работа № 3

По дисциплине «Объектно-ориентированное программирование и проектирование»

«Перегрузка операций»

Выполнил:
Студент 2-го курса
Группы АС-53
Вожейко М.В.
Проверил:
Давидюк Ю. И.

Брест, 2020

Цель: получить практические навыки создания абстрактных типов данных и перегрузки операций в языке C++

Вариант 6

Постановка задачи:

АДТ-очередь. Дополнительно перегрузить следующие операции:

+ - добавить элемент

- - извлечь элемент

Bool() – проверка, пустая ли очередь

Определение класса:

```
#ifndef MQUEUE_H
#define MQUEUE_H
class mqueue {
private:
  int* p;
  int count;
public:
  mqueue() : count(0) { }
  ~mqueue();
  inline bool empty() { return count == 0; }
  inline int size() const { return count; }
  void push(int item);
  void pop();
  void print();
  void input(int size);
  mqueue& operator+=(int item);
  mqueue& operator-=(int count);
  mqueue& operator=(const mqueue &);
  friend mqueue operator+(const mqueue &, int item);
  friend mqueue operator-(const mqueue &, int count);
};
#endif
```

Функции и операторы класса:

```
#include "mqueue.h"
#include <iostream>

void mqueue::input(int size) {
	int key;
	for (int k = 0; k < size; k++) {
		std::cout << "Enter element #" << k << ": ";
		std::cin >> key;
		this->push(key);
	}
}

void mqueue::print() {
	for (int i = 0; i < count; i++)
		std::cout << p[i] << "\t";
	std::cout << std::endl;
	std::cout << "Print Done.\n" << std::endl;
}

void mqueue::push(int item)
{
	int* p2;
	p2 = p;
```

```
	try {
		p = new int[count + 1];
		for (int i = 0; i < count; i++)
			p[i] = p2[i];
		p[count] = item;
		count++;
		if (count > 1)
			delete[] p2;
	}
	catch (std::bad_alloc e) {
		std::cout << e.what() << std::endl;
		p = p2;
	}
}

void mqueue::pop() {
	if (count == 0)
		return;
	try {
		int* p2;
		p2 = new int[--count];
		for (int i = 0; i < count; i++)
			p2[i] = p[i + 1]; // Копируется всё, кроме первого элемента
		if (count > 0)
			delete[] p;
		p = p2;
	}
	catch (std::bad_alloc e)
	{
		std::cout << e.what() << std::endl;
	}
}

mqueue::~mqueue() {
	if (count > 0)
		delete[] p;
}

mqueue& mqueue::operator=(const mqueue& obj) {
	int* p2;

	try {
		p2 = new int[obj.count];
		if (count > 0)
			delete[] p;
		p = p2;
		count = obj.count;
		for (int i = 0; i < count; i++)
			p[i] = obj.p[i];
	}
	catch (std::bad_alloc e)
	{
		std::cout << e.what() << std::endl;
	}
	return *this;
}

mqueue& mqueue::operator+=(int item) {
	this->push(item);
	return *this;
}
mqueue& mqueue::operator-=(int count) {
	for (int k = 0; k < count; k++)
		this->pop();
	return *this;
}
mqueue operator+(const mqueue& lhs, int item) {
	mqueue bufferq = lhs;
	bufferq += item;
	return bufferq;
}
mqueue operator-(const mqueue& lhs, int count) {
	mqueue bufferq = lhs;
```

```
	bufferq -= count;
	return bufferq;
}
```

Обоснование включения в класс нескольких конструкторов, деструктора и операции присваивания:

Деструктор:

mqueue::~mqueue() {
if (count > 0)
delete[] p;
}

Очищает выделенную для работы память.

Конструктор:

mqueue() : count(0) { }

«Пустой» конструктор, нужен для создания объекта класса.

Оператор присваивания:

`mqueue& mqueue::operator=(const mqueue& obj)`

Нужен для полного копирования объекта.

Объяснить выбранное представление памяти для объектов реализуемого класса:

Динамические массив int *, т.к. требуется изменять размер при добавлении и удалении элементов.

6. Реализация перегруженных операций с обоснованием выбранного способа (функция – член класса, внешняя функция, внешняя дружественная функция).

`mqueue& mqueue::operator=(const mqueue& obj)` – оператор присваивания, «копирует» объект.

`mqueue& mqueue::operator+=` - оператор добавления элемента в очередь.

`mqueue& mqueue::operator-=` - удаляет n элементов из очереди.

Операторы выше являются членами класса, т.к. требуется доступ к private-полям класса.

`mqueue operator+` - добавляет элемент в «копию» очереди.

`mqueue operator-` - удаляет n элементов из «копии» очереди.

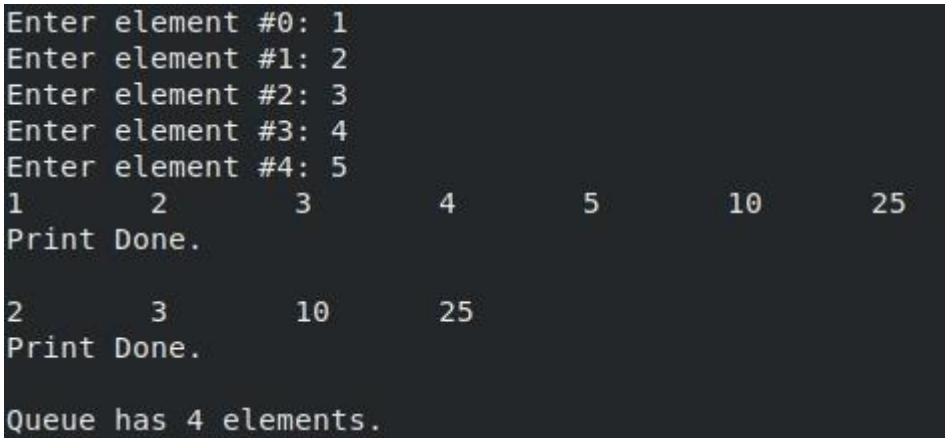
Данные операторы являются дружественными, т.к. получают доступ к приватным полям, но не изменяют содержимое, а возвращают копию.

7. Тестовая программа

```
#include <iostream>
#include "mqueue.h"

int main()
{
	mqueue queue1;
	queue1.input(5);
	queue1 = queue1 + 10;
	queue1 = queue1 + 25;
	queue1.print();
	queue1 = queue1 - 3;
	queue1.print();
	if (queue1.empty())
		std::cout << "Queue is empty." << std::endl;
	else
		std::cout << "Queue has " << queue1.size() << " elements." << std::endl;

	return 0;
}
```



Алгоритм действия на словах:
Создаем очередь -> Вводим 5 элементов через консоль -> Добавляем 10 -> Добавляем 25 -> Выводим -> Удаляем 3 элемента -> Выводим -> Проверяем пуста ли очередь.

Вывод: получил практические навыки создания абстрактных типов данных и перегрузки операций в языке C++.