Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Институт информационных технологий и управления в технических системах

ОТЧЕТ

по лабораторной работе №1

КЛАССЫ. КОНСТРУКТОРЫ И ДЕСТРУКТОРЫ.

КОНСТРУКТОРЫ КОПИРОВАНИЯ.

Выполнил: ст. гр. ИТб-21

Куркчи А.Э.

Проверила:

Тимофеев И.С.

Севастополь

2015

# 1. Цель

Изучить основные средства определения класса, создания объектов класса, работы с такими объектами и их уничтожения после использования.

2. Постановка задач

Необходимо для заданного по варианту динамического типа данных описать класс, содержащий указатель на динамический тип как поле данных. Для этого класса описать конструкторы (в том числе и конструктор копирования), деструктор, функцию печати данных. Создать экземпляр полученного класса и проиллюстрировать его корректную работу: распечатать данные, изменить данные и распечатать вновь. Создать второй экземпляр класса как копи. Первого и проиллюстрировать корректную работу конструктора копирования: распечатать и изменить данные объекта-копии, распечатать данные обоих объектов, сравнить результат.

Вариант №1

Динамическая структура – очередь. В списке хранится информация о событиях: день, месяц, год и название. Предусмотреть функции добавления элементов в список и удаления из него, а также поиска введенной даты.

3. Текст программы

#include <iostream>

#include <stdio.h>

#include <string.h>

#include <math.h>

**using** **namespace** std**;**

#define STR\_LEN 128

struct event **{**

char name**[**STR\_LEN**];**

int day**,**month**,**year**;**

**};**

struct node **{**

event e**;**

node**\*** next**;**

**};**

event empty**()** **{**

event empty**;**

strcpy**(**empty**.**name**,**"empty"**);**

**return** empty**;**

**}**

class queue **{**

private**:**

node**\*** head **=** **NULL;**

node**\*** tail **=** **NULL;**

public**:**

queue**()** **{}**

queue**(**event e**)** **{**

head **=** tail **=** **new** node**;**

strcpy**(**tail**->**e**.**name**,**e**.**name**);**

tail**->**e**.**day **=** e**.**day**;**

tail**->**e**.**month **=** e**.**month**;**

tail**->**e**.**year **=** e**.**year**;**

tail**->**next **=** **NULL;**

**}**

queue**(**const queue **&**q**)** **{**

**if(**q**.**head **!=** **NULL)** **{**

node**\*** p **=** q**.**head**;**

**while(**p **!=** **NULL)** **{**

node**\*** cur **=** **new** node**;**

**if(**head **==** **NULL)** **{**

head **=** tail **=** cur**;**

**}** **else** **{**

tail**->**next **=** cur**;**

tail **=** cur**;**

**}**

cur**->**e**.**day **=** p**->**e**.**day**;**

cur**->**e**.**month **=** p**->**e**.**month**;**

cur**->**e**.**year **=** p**->**e**.**year**;**

strcpy**(**cur**->**e**.**name**,**p**->**e**.**name**);**

p **=** p**->**next**;**

**}**

**}**

**}**

**~**queue**()** **{**

**if(**head **!=** **NULL)** **{**

node**\*** cur **=** head**;**

node**\*** next **=** head**->**next**;**

**while(**cur **!=** **NULL)** **{**

**delete** cur**;**

cur **=** next**;**

next **=** cur **==** **NULL** **?** **NULL** **:** cur**->**next**;**

**}**

**}**

**}**

void push**(**event e**)** **{**

**if(**head **==** **NULL)** **{**

head **=** tail **=** **new** node**;**

**}** **else** **{**

node**\*** tmp **=** **new** node**;**

tail**->**next **=** tmp**;**

tail **=** tmp**;**

**}**

strcpy**(**tail**->**e**.**name**,**e**.**name**);**

tail**->**e**.**day **=** e**.**day**;**

tail**->**e**.**month **=** e**.**month**;**

tail**->**e**.**year **=** e**.**year**;**

tail**->**next **=** **NULL;**

**}**

event pop**()** **{**

**if(**head **==** **NULL)** **{**

cout **<<** "Queue is empty" **<<** endl**;**

**return** empty**();**

**}**

node**\*** tmp **=** head**;**

event ret\_val **=** tmp**->**e**;**

head **=** tmp**->**next**;**

**delete** tmp**;**

**return** ret\_val**;**

**}**

event pop**(**const char**\*** n**)** **{**

**if(**head **==** **NULL)** **{**

cout **<<** "Queue is empty" **<<** endl**;**

**return** empty**();**

**}**

**if(**strcmp**(**n**,**head**->**e**.**name**)** **==** 0**)** **{**

**return** pop**();**

**}**

node**\*** prev **=** head**;**

node**\*** cur **=** head**->**next**;**

**while(**cur **!=** **NULL)** **{**

**if(**strcmp**(**n**,**cur**->**e**.**name**)** **==** 0**)** **{**

event ret\_val **=** cur**->**e**;**

prev**->**next **=** cur**->**next**;**

**delete** cur**;**

**return** ret\_val**;**

**}**

prev **=** cur**;**

cur **=** cur**->**next**;**

**}**

cout **<<** "No such event in queue" **<<** endl**;**

**return** empty**();**

**}**

event pop**(**const int d**,**const int m**,**const int y**)** **{**

**if(**head **==** **NULL)** **{**

cout **<<** "Queue is empty" **<<** endl**;**

**return** empty**();**

**}**

**if(**head**->**e**.**day **==** d **&&** head**->**e**.**month **==** m **&&** head**->**e**.**year **==** y**)** **{**

**return** pop**();**

**}**

node**\*** prev **=** head**;**

node**\*** cur **=** head**->**next**;**

**while(**cur **!=** **NULL)** **{**

**if(**cur**->**e**.**day **==** d **&&** cur**->**e**.**month **==** m **&&** cur**->**e**.**year **==** y**)** **{**

event ret\_val **=** cur**->**e**;**

prev**->**next **=** cur**->**next**;**

**delete** cur**;**

**return** ret\_val**;**

**}**

prev **=** cur**;**

cur **=** cur**->**next**;**

**}**

cout **<<** "No such event in queue" **<<** endl**;**

**return** empty**();**

**}**

event find**(**const char**\*** n**)** **{**

**if(**head **==** **NULL)** **{**

cout **<<** "Queue is empty" **<<** endl**;**

**return** empty**();**

**}**

**if(**strcmp**(**n**,**head**->**e**.**name**)** **==** 0**)** **{**

**return** head**->**e**;**

**}**

node**\*** prev **=** head**;**

node**\*** cur **=** head**->**next**;**

**while(**cur **!=** **NULL)** **{**

**if(**strcmp**(**n**,**cur**->**e**.**name**)** **==** 0**)** **{**

**return** cur**->**e**;**

**}**

prev **=** cur**;**

cur **=** cur**->**next**;**

**}**

cout **<<** "No such event in queue" **<<** endl**;**

**return** empty**();**

**}**

event find**(**const int d**,**const int m**,**const int y**)** **{**

**if(**head **==** **NULL)** **{**

cout **<<** "Queue is empty" **<<** endl**;**

**return** empty**();**

**}**

**if(**head**->**e**.**day **==** d **&&** head**->**e**.**month **==** m **&&** head**->**e**.**year **==** y**)** **{**

**return** head**->**e**;**

**}**

node**\*** prev **=** head**;**

node**\*** cur **=** head**->**next**;**

**while(**cur **!=** **NULL)** **{**

**if(**cur**->**e**.**day **==** d **&&** cur**->**e**.**month **==** m **&&** cur**->**e**.**year **==** y**)** **{**

**return** cur**->**e**;**

**}**

prev **=** cur**;**

cur **=** cur**->**next**;**

**}**

cout **<<** "No such event in queue" **<<** endl**;**

**return** empty**();**

**}**

void print**()** **{**

**if(**head **==** **NULL)** **{**

cout **<<** "Queue is empty" **<<** endl**;**

**}**

node**\*** p **=** head**;**

**while(**p **!=** **NULL)** **{**

cout **<<** p**->**e**.**day **<<** "." **<<** p**->**e**.**month **<<** "." **<<** p**->**e**.**year **<<** ": " **<<** p**->**e**.**name **<<** endl**;**

p **=** p**->**next**;**

**}**

**}**

**};**

event input**()** **{**

event e**;**

cout **<<** "Name:" **<<** endl**;**

cin**.**getline**(**e**.**name**,**STR\_LEN**);**

cout **<<** "Date (d m y):" **<<** endl**;**

cin **>>** e**.**day **>>** e**.**month **>>** e**.**year**;**

**return** e**;**

**}**

int main**()** **{**

cout **<<** "SevGU semestr.03 OOP lab.01" **<<** endl**;**

cout **<<** "Please input 3 events" **<<** endl**;**

cout **<<** "Event a: " **<<** endl**;**

event a **=** input**();**

cout **<<** "Event b: " **<<** endl**;**

event b **=** input**();**

cout **<<** "Event c: " **<<** endl**;**

event c **=** input**();**

cout **<<** "Demonstration start" **<<** endl**;**

// cout << "Creating queue q1 with element constructor on event a" << endl;

cout **<<** "> queue q1(a);" **<<** endl**;**

queue q1**(**a**);**

// cout << "Pushing event b to queue q1" << endl;

cout **<<** "> q1.push(b);" **<<** endl**;**

q1**.**push**(**b**);**

// cout << "Printing queue q1" << endl;

cout **<<** "> q1.print();" **<<** endl**;**

q1**.**print**();**

// cout << "Creating queue q2 with copy constructor" << endl;

cout **<<** "> queue q2(q1);" **<<** endl**;**

queue q2**(**q1**);**

// cout << "Pushing event c to queue q2" << endl;

cout **<<** "> q2.push(c);" **<<** endl**;**

q2**.**push**(**c**);**

// cout << "Poping event a from queue q2 by day, month and year" << endl;

cout **<<** "> q2.pop(a.day,a.month,a.year);" **<<** endl**;**

q2**.**pop**(**a**.**day**,**a**.**month**,**a**.**year**);**

// cout << "Poping event b from queue q2 by name" << endl;

cout **<<** "> q2.pop(b.name);" **<<** endl**;**

q2**.**pop**(**b**.**name**);**

// cout << "Pushing event a to queue q2" << endl;

cout **<<** "> q2.push(a);" **<<** endl**;**

q2**.**push**(**a**);**

// cout << "Printing queue q1" << endl;

cout **<<** "> q1.print();" **<<** endl**;**

q1**.**print**();**

// cout << "Printing queue q2" << endl;

cout **<<** "> q2.print();" **<<** endl**;**

q2**.**print**();**

cout **<<** "Demonstration end" **<<** endl**;**

**return** 0**;**

**}**

4. Тестовые примеры

На рисунке 1 представлен вывод программы

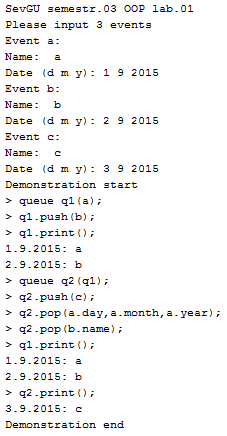


Рисунок 1 – Вывод программы

После ввода 3х событий начинается демонстрация корректности программы. В строках, начинающихся с «>», печатается выполняемая в программном коде строка, в остальных строках – вывод программы. Метод push отвечает за вставку элемента в конец очереди, pop за его изъятие из неё и print за печать элементов очереди начиная с головы.

Вывод

В ходе лабораторной работы были изучены основные средства определения класса, создания объектов класса, работы с такими объектами и их уничтожения после использования.