Лабораторна робота №6

КОНТЕЙНЕРИ.

***Мета роботи***: Отримати навики розробки власних контейнерів на базі існуючих класів.

**1 ІНДИВІДУАЛЬНЕ ЗАВДАННЯ**

Визначити клас CapacityList для зберігання об’єктів даних. У колекції повинен бути перевизначений оператор [].

**2 ОПИС ПРОГРАМИ**

**2.1 Ієрархія та структура класів**

**class Capacity –** клас, який містить атрибути **data\_field1 data\_field2.**

**class CapacityList –** клас – список.

**2.2 Опис програми**

На рисунку 2.1 наведена структура розробленої програми

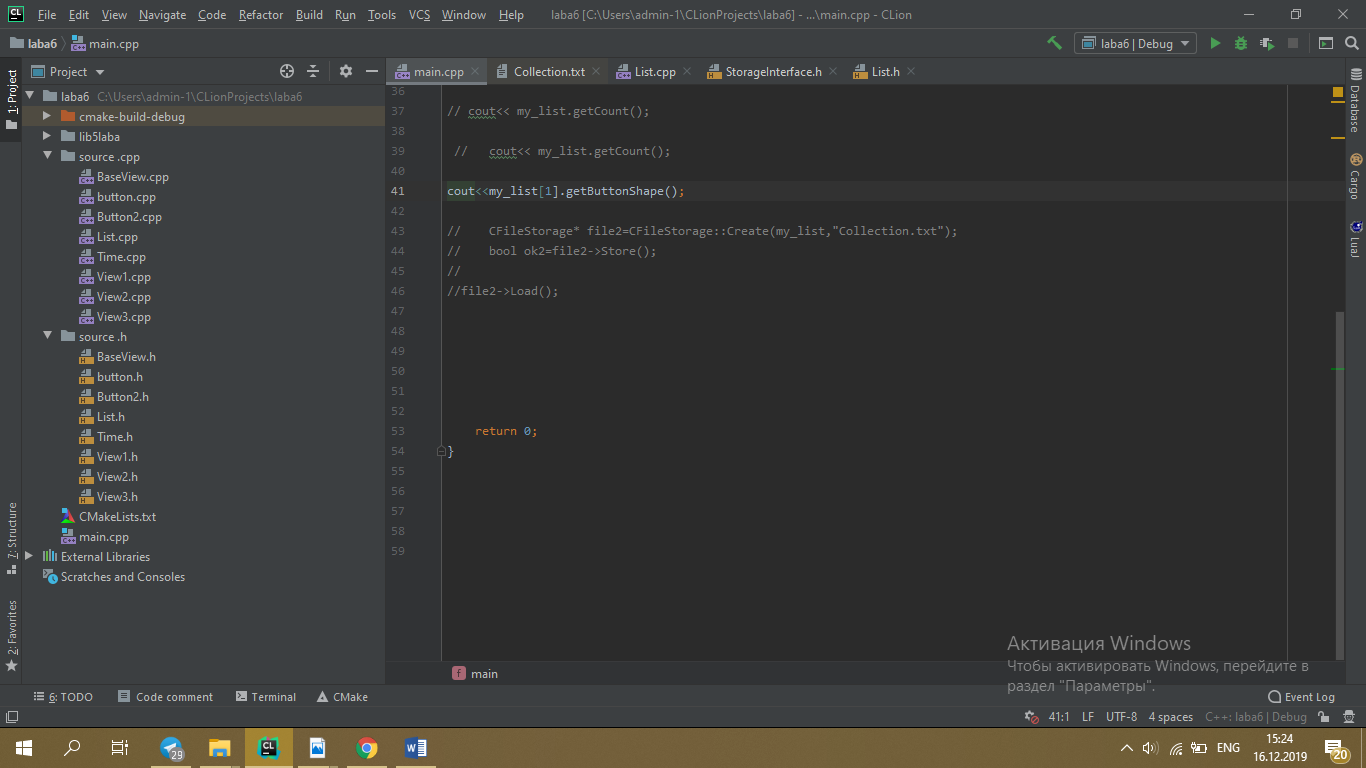


Рис 2.1 – структура програми

**2.3 Важливі фрагменти програми**

2.3.1 Файл*main.cpp*

#define \_CRT\_SECURE\_NO\_WARNINGS  
  
  
#include "source .cpp/List.cpp"  
#include "source .h/List.h"  
#include <iostream>  
#include "source .cpp/button.cpp"  
#include "source .cpp/Button2.cpp"  
#include "source .h/button.h"  
#include "source .h/Button2.h"  
#include "source .cpp/View1.cpp"  
#include "source .cpp/View2.cpp"  
#include "source .cpp/View3.cpp"  
#include "source .cpp/BaseView.cpp"  
#include "source .h/BaseView.h"  
#include "source .cpp/Time.cpp"  
#include "source .h/Time.h"  
#include <chrono>  
#include "lib5laba/FileStorage.cpp"  
#include "lib5laba/FileStorage.h"  
#include "lib5laba/StorageInterface.h"  
#include <stdlib.h>  
  
int main ()  
{  
 List<button> my\_list;  
  
button b,a,c;  
a.setButtonShape(*Oval*);  
b.setButtonShape(*Rectangular*);  
b.setButtonState(*OFF*);  
c.setButtonShape(*Oval*);  
my\_list.add(b);  
my\_list.add(a);  
my\_list.add(c);  
  
// cout<< my\_list.getCount();  
  
 // cout<< my\_list.getCount();  
  
cout<<my\_list[1].getButtonShape();  
  
// CFileStorage\* file2=CFileStorage::Create(my\_list,"Collection.txt");  
// bool ok2=file2->Store();  
//  
//file2->Load();  
  
  
  
  
  
  
 return 0;  
}

2.3.1 Файл*button.cpp*

#define \_CRT\_SECURE\_NO\_WARNINGS  
  
//  
// Created by admin-1 on 06.09.2019.  
//  
#pragma once  
#include "../source .h/button.h"  
#include <iostream>  
#include <fstream>  
 void button::setButtonState(ButtonState b) {  
 bt = b;  
 }  
  
  
  
  
bool button::IsRoundPressed() {  
 return getButtonState() == 0;  
}  
  
void button::setButtonShape(ButtonShape b) {  
bs=b;  
}  
  
ButtonShape button::getButtonShape() const {  
 return this->bs;  
}  
  
void button::ButtonOperation() {  
 if(IsRoundPressed()){  
 cout<<"System block works";  
 }  
 else  
 {  
 cout<<"System block doesn't work";  
 }  
  
}  
  
ButtonState button::getButtonState() const {  
 return bt;  
}  
  
void button::OnStore(std::ofstream &aStream) {  
  
 aStream << "Shape of this button is " <<bs<<endl;  
 aStream << "State of this button is " <<bt<<endl;  
  
aStream.close();  
  
}  
  
void button::OnLoad(std::ifstream &aStream) {  
 char line[50];  
 for (int j = 0; j < 2; ++j) {  
 aStream.getline(( char \*) line, sizeof(line));  
 for (int i = 0; i <50 ; ++i) {  
  
 cout<<line[i];  
 }  
 cout<<endl;  
 }  
}

2.3.1 Файл*button.h*

#define \_CRT\_SECURE\_NO\_WARNINGS  
  
#pragma once  
#include <string>  
#include "lib5laba/StorageInterface.h"  
  
using namespace std;  
enum ButtonState  
{  
 *ON*,  
 *OFF*};  
enum ButtonShape  
 {  
*Oval*,  
*Rectangular* };  
  
 class button :public MStorageInterface{  
 protected:  
 ButtonState bt;  
 ButtonShape bs;  
 int sa;  
 public:  
 void OnStore(std::ofstream &aStream) override;  
  
 void OnLoad(std::ifstream &aStream) override;  
  
 public:  
  
 void setButtonState(ButtonState b);  
 void setButtonShape(ButtonShape b);  
 ButtonShape getButtonShape() const;  
 ButtonState getButtonState() const;  
  
 bool IsRoundPressed();  
  
 virtual void ButtonOperation();  
  
  
};

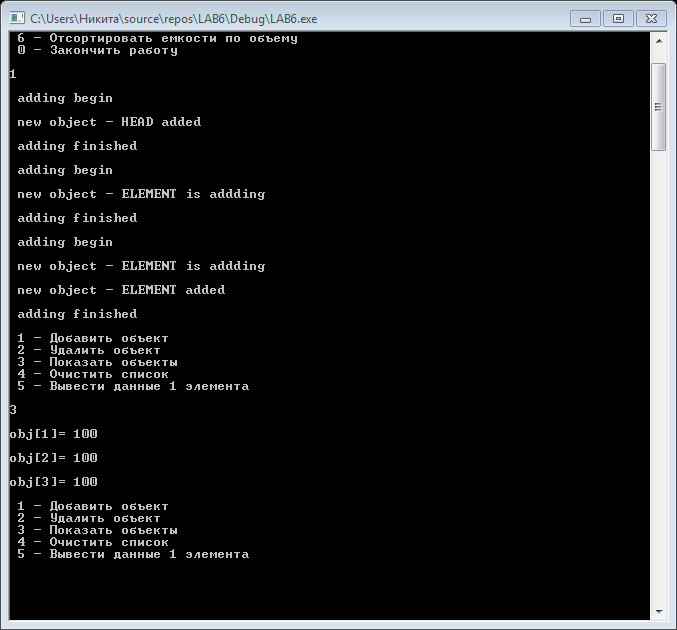
2.3.2 Файл*List.cpp*

#define \_CRT\_SECURE\_NO\_WARNINGS  
  
//  
// Created by admin-1 on 29.09.2019.  
//  
#include <iostream>  
#include "../source .h/List.h"  
using namespace std;  
template<typename T>  
List<T>::node::node() : next(NULL){}  
  
template<typename T>  
List<T>::node::~node() {}  
  
template<typename T>  
List<T>::List(): head(NULL), count(0){}  
  
template<typename T>  
List<T>::~List() {  
clear();  
}  
  
template<typename T>  
int List<T>::getCount() const {  
 return count;  
}  
  
template<typename T>  
int List<T>::add(T data) {  
 node \*to\_add = new node;  
 to\_add->next=NULL;  
 to\_add->val=data;  
 if(head==NULL) //Если в списке нет элементов  
 head=to\_add;  
 else  
 {  
 node \*current;  
 for(current=head;current->next!=0;current=current->next);  
 current->next=to\_add;  
 }  
 count++;  
 return count;  
}  
  
template<typename T>  
int List<T>::del(int x) { //Возвращает -1, если произошла ошибка  
 if (x>count) return -1;  
 node \*to\_del=head;  
 if (x==1) //Если нужно удалить первый элемент  
 {  
 head=head->next;  
 delete to\_del;  
 }  
 else  
 {  
 node \*current=head;  
 for(int i=1;i<x-1;i++)  
 current=current->next;  
 to\_del=current->next;  
 current->next=current->next->next;  
 delete to\_del;  
 }  
 count--;  
 return count;  
}  
  
template<typename T>  
void List<T>::clear() {  
 node \*current = head;  
 node \*to\_del = head;  
 while(to\_del!=NULL)  
 {  
 current=current->next;  
 delete to\_del;  
 to\_del=current;  
 }  
 head=NULL;  
 count=0;  
}  
  
template<typename T>  
T List<T>::getData(int x) const {  
  
 node \*current;  
 for(current=head;x>1;x--)  
 current=current->next;  
 return current->val;  
  
}  
  
template<typename T>  
void List<T>::OnStore(std::ofstream &aStream) {  
aStream.write((char\* )&this->getData(0), sizeof(T));  
}  
  
template<typename T>  
void List<T>::OnLoad(std::ifstream &aStream) {  
 button list;  
aStream.read((char\*)&list, sizeof(T));  
  
cout<<list.getButtonShape();  
cout<<list.getButtonState();  
  
}  
  
template<typename T>  
T List<T>::operator[](int num) {  
node\* current=head;  
 for (int i = 0; i < num; ++i) {  
 current->next;  
 }  
 return current->val;  
}

2.3.3 Файл*List.h*

#pragma once  
#include "button.h"  
#include "../lib5laba/StorageInterface.h"  
  
#define \_CRT\_SECURE\_NO\_WARNINGS  
  
//  
// Created by admin-1 on 29.09.2019.  
//  
  
#ifndef LABA6\_LIST\_H  
#define LABA6\_LIST\_H  
  
  
template <typename T>  
class List:public MStorageInterface  
{  
public:  
 T operator[](int num);  
 void OnStore(std::ofstream &aStream) override;  
  
 void OnLoad(std::ifstream &aStream) override;  
  
private:  
 class node //Класс node содержит данные списка  
 {  
 friend class List<T>;  
 private:  
 node \*next; //Указатель на следующий элемент в списке  
 T val; //Данные списка  
  
 node();  
 ~node();  
 };  
  
 node \*head; //Указатель на начало списка  
 int count; //Количество элементов в списке  
public:  
 List();  
 ~List();  
 int getCount() const ;  
  
 int add(T data);  
  
 int del(int x);  
  
 void clear();  
  
  
 T getData(int x) const ;  
  
  
  
  
};  
  
  
  
  
#endif //LABA6\_LIST\_H

**3 РЕЗУЛЬТАТИ**

****

**ВИСНОВКИ**

Отримали навики розробки власних контейнерів на базі існуючих класів.