Министерство высшего образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (ПНИПУ)

Электротехнический факультет

Дисциплина: основы алгоритмизации и программирования, 2 семестр

**ОТЧЁТ**

Тема: «Шаблоны классов»

Выполнил

Студент РИС-22-2б

Зубов Р.А.

Проверил

Доцент кафедры ИТАС

Полякова О. А.

Пермь 2023

**Постановка задачи**

1. Определить шаблон класса-контейнера (см. лабораторную работу №6).
2. Реализовать конструкторы, деструктор, операции ввода-вывода, операцию присваивания.
3. Перегрузить операции, указанные в варианте.
4. Инстанцировать шаблон для стандартных типов данных (int, float, double).
5. Написать тестирующую программу, иллюстрирующую выполнение операций для контейнера, содержащего элементы стандартных типов данных.
6. Реализовать пользовательский класс (см. лабораторную работу №3).
7. Перегрузить для пользовательского класса операции ввода-вывода.
8. Перегрузить операции необходимые для выполнения операций контейнерного класса.
9. Инстанцировать шаблон для пользовательского класса.
10. Написать тестирующую программу, иллюстрирующую выполнение операций для контейнера, содержащего элементы пользовательского класса.

Класс- контейнер СПИСОК с ключевыми значениями типа int. Реализовать операции:

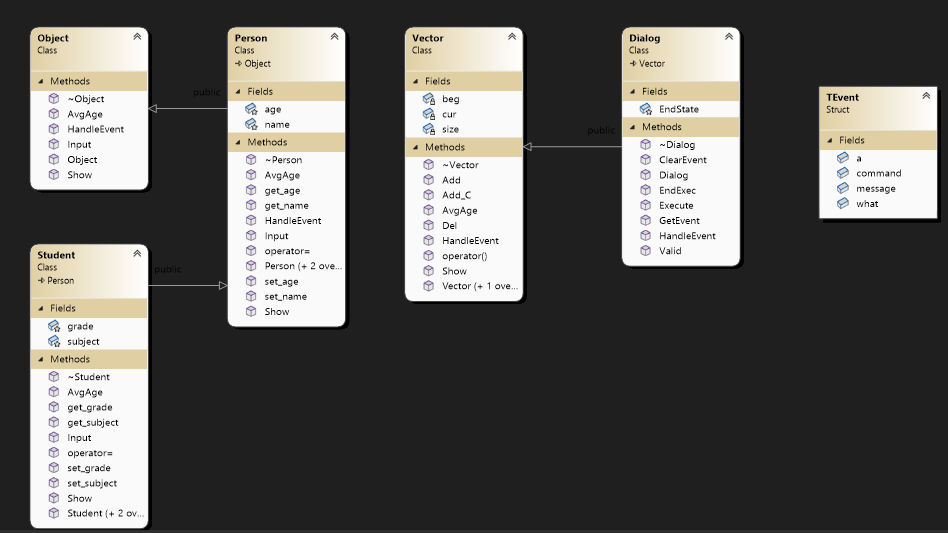
[] – доступа по индексу;

int() – определение размера списка;

\* вектор – умножение элементов списков a[i]\*b[i];

Пользовательский класс Pair (пара чисел). Пара должна быть представлено двумя полями: типа int для первого числа и типа double для второго. Первое число при выводе на экран должно быть отделено от второго числа двоеточием.

**UML**

****

**Код программы**

**Dialog.h:**

#pragma once

#include "Vector.h"

#include <string>

class Dialog :

public Vector

{

public:

Dialog(void);

virtual ~Dialog(void);

virtual void GetEvent(TEvent& event);

virtual int Execute();

virtual void HandleEvent(TEvent& event);

virtual void ClearEvent(TEvent& event);

int Valid();

void EndExec();

protected:

int EndState;

};

**Dialog.cpp:**

#include "Dialog.h"

#include <string>

#include <iostream>

Dialog::Dialog(void) :Vector()

{

EndState = 0;

}

Dialog::~Dialog(void)

{

}

void Dialog::GetEvent(TEvent& event)

{

std::string OpInt = "+-mszq";

std::string s;

std::string param;

char code;

std::cout << '>';

std::cin >> s; code = s[0];

if (OpInt.find(code) >= 0)

{

event.what = evMessage;

switch (code)

{

case 'm': event.command = cmMake; break;

case 's': event.command = cmShow; break;

case 'z': event.command = cmGet; break;

case '+': event.command = cmAdd; break;

case '-': event.command = cmDel; break;

case 'q': event.command = cmQuit; break;

}

if (s.length() > 1)

{

param = s.substr(1, s.length() - 1);

int A = atoi(param.c\_str());

event.a = A;

}

}

else event.what = evNothing;

}

int Dialog::Execute()

{

TEvent event;

do {

EndState = 0;

GetEvent(event);

HandleEvent(event);

} while (!Valid()); return EndState;

}

int Dialog::Valid()

{

if (EndState == 0) return 0; else return 1;

}

void Dialog::ClearEvent(TEvent& event)

{

event.what = evNothing;

}

void Dialog::EndExec()

{

EndState = 1;

}

void Dialog::HandleEvent(TEvent& event)

{

if (event.what == evMessage)

{

switch (event.command)

{

case cmMake:

std::cout << "Size of group: ";

std::cin >> size;

beg = new Object \* [size];

cur = 0;

ClearEvent(event);

break;

case cmAdd:

Add\_C();

ClearEvent(event);

break;

case cmDel:

Del();

ClearEvent(event);

break;

case cmShow:

Show();

ClearEvent(event);

break;

case cmQuit:

EndExec();

ClearEvent(event);

break;

case cmGet:

AvgAge();

ClearEvent(event);

break;

default:

ClearEvent(event);

HandleEvent(event);

};

};

}

**Event.h:**

#pragma once

//обработка событий

const int evNothing = 0;// пустое сообщение

const int evMessage = 100;// непустое

const int cmAdd = 1;// добавить объект

const int cmDel = 2;// удалить объект

const int cmGet = 3;// вывести атрибут объектов

const int cmShow = 4;// вывести группу

const int cmMake = 6; // создать группу

const int cmQuit = 101; // выход

struct TEvent

{

int what; // тип события

int command; // код команды

int message;

int a; // параметр команды

};

**Vector.h:**

#pragma once

#include "Student.h"

#include <string>

#include <iostream>

class Vector

{

public:

Vector(void);

Vector(int);

~Vector(void);

void Add(Object\* p);

void Add\_C();

void Del();

void Show();

int operator ()();

friend std::ostream& operator <<(std::ostream& out, const Vector&);

virtual void HandleEvent(const TEvent& e);

void AvgAge();

protected:

Object\*\* beg;

int size;

int cur;

};

**Vector.cpp:**

#include "Vector.h"

#include <iostream>

Vector::Vector(void)

{

beg = 0;

size = 0;

cur = 0;

}

Vector::~Vector()

{

if (beg != 0) delete[] beg;

beg = 0;

}

Vector::Vector(int n)

{

beg = new Object \* [n];

cur = 0;

size = n;

}

void Vector::Add(Object\* p)

{

if (cur < size)

{

beg[cur] = p;

cur++;

}

}

void Vector::Add\_C()

{

Object\* a;

std::cout << "Who u need to add?\n";

std::cout << "1. Person\n";

std::cout << "2. Student\n";

int ch;

std::cin >> ch;

switch (ch)

{

case 1:

{

Person\* p = new Person();

p->Input();

a = p;

Add(a);

break;

}

case 2:

{

Student\* s = new Student();

s->Input();

a = s;

Add(a);

break;

}

default:

{

std::cout << "Wrong num\n";

break;

}

}

}

void Vector::Show()

{

if (size == 0)

std::cout << "Empty\n";

Object\*\* p = beg;

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

{

(\*p)->Show();

++p;

}

}

void Vector::Del()

{

if (cur == 0)

return;

cur--;

}

int Vector::operator()()

{

return size;

}

std::ostream& operator<<(std::ostream& out, const Vector& v)

{

if (v.size == 0) out << "Empty\n";

Object\*\* p = v.beg;

for (int i = 0; i < v.cur; i++)

{

(\*p)->Show();

p++;

}

return out;

}

void Vector::HandleEvent(const TEvent& e)

{

if (e.what == evMessage)

{

Object\*\* p = beg;

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

{

(\*p)->HandleEvent(e);

p++;

}

}

}

void Vector::AvgAge()

{

int a = 0;

if (cur == 0)

{

std::cout << "Empty!\n";

return;

}

Object\*\* p = beg;

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

{

a += (\*p)->AvgAge();

p++;

}

std::cout << "Average age is: " << a / cur << '\n';

}

**Obeject.h:**

#pragma once

#include "Event.h"

class Object

{

public:

Object();

virtual void Show() = 0;

virtual void Input() = 0;

virtual ~Object();

virtual void HandleEvent(const TEvent& e) = 0;

virtual int AvgAge() = 0;

};

**Obeject.cpp:**

#include "Object.h"

Object::Object()

{

}

Object::~Object()

{

}

**Person.h:** #pragma once

#include <string>

#include <iostream>

#include "Object.h"

class Person : public Object {

public:

Person(void);

Person(std::string, int);

Person(const Person&);

virtual ~Person() {};

void Show();

void Input();

std::string get\_name() { return name; }

int get\_age() { return age; }

void set\_name(std::string);

void set\_age(int);

Person& operator=(const Person&);

void HandleEvent(const TEvent& e);

int AvgAge() { return age; }

protected:

std::string name;

int age;

};

**Person.cpp:**

#include "Person.h"

Person::Person(void)

{

name = "";

age = 0;

}

Person::Person(std::string N, int A)

{

name = N;

age = A;

}

Person::Person(const Person& p)

{

name = p.name;

age = p.age;

}

void Person::set\_name(std::string N)

{

name = N;

}

void Person::set\_age(int A)

{

age = A;

}

Person& Person::operator=(const Person& p)

{

if (&p == this) return \*this;

name = p.name;

age = p.age;

return \*this;

}

void Person::Show()

{

std::cout << "\nNAME: " << name;

std::cout << "\nAGE: " << age;

std::cout << "\n";

}

void Person::Input()

{

std::cout << "\nName: "; std::cin >> name;

std::cout << "\nAge: "; std::cin >> age;

}

void Person::HandleEvent(const TEvent& e)

{

if (e.what == evMessage)

{

switch (e.command)

{

case cmGet:

Show();

break;

}

}

}

**Student.h:**

#pragma once

#include "Person.h"

#include "Object.h"

#include <iostream>

#include <string>

class Student :

public Person

{

public:

Student(void);

Student(std::string, int, std::string, int);

Student(const Student&);

~Student() {};

void Show();

void Input();

std::string get\_subject() { return subject; }

int get\_grade() { return grade; }

void set\_subject(std::string);

void set\_grade(int);

Student& operator=(const Student&);

int AvgAge() { return age; }

protected:

std::string subject;

int grade;

};

**Student.cpp:**

#include "Student.h"

#include <iostream>

Student::Student(void) : Person()

{

subject = "";

grade = 0;

}

Student::Student(std::string N, int A, std::string S, int G) : Person(N, A)

{

subject = S;

grade = G;

}

Student::Student(const Student& c)

{

name = c.name;

age = c.age;

subject = c.subject;

grade = c.grade;

}

void Student::set\_subject(std::string S)

{

subject = S;

}

void Student::set\_grade(int G)

{

grade = G;

}

Student& Student::operator=(const Student& c)

{

if (&c == this) return \*this;

name = c.name;

age = c.age;

subject = c.subject;

grade = c.grade;

return \*this;

}

void Student::Show()

{

std::cout << "\nName: " << name;

std::cout << "\nAge: " << age;

std::cout << "\nSubject: " << subject;

std::cout << "\nGrade: " << grade;

std::cout << "\n";

}

void Student::Input()

{

std::cout << "\nName: "; std::cin >> name;

std::cout << "\nAge: "; std::cin >> age;

std::cout << "\nSubject: "; std::cin >> subject;

std::cout << "\nGrade: "; std::cin >> grade;

}

**main.cpp:**

#include <iostream>

#include "Dialog.h"

using namespace std;

int main()

{

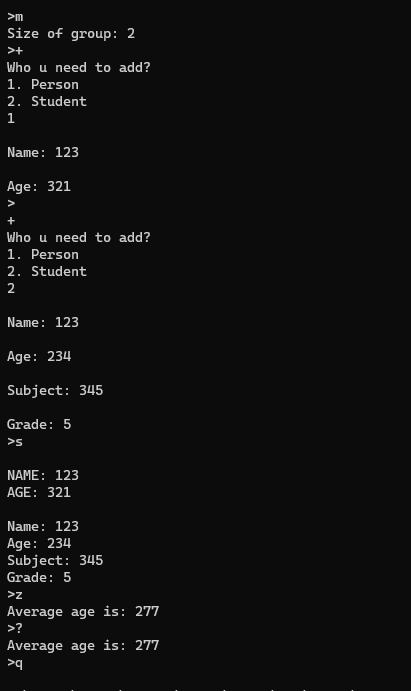
Dialog D;

D.Execute();

return 0;

}

**Результаты программ**



**Контрольные вопросы**

1. Что такое класс-группа? Привести примеры таких классов.

Класс-группа - это класс, который представляет собой группу объектов одного типа. Примерами таких классов могут быть классы "Массив", "Очередь", "Стек", "Список" и т.д.

Привести пример описания класса-группы Список (List). template<typename T>

class List {

Node<T>\* head;

Node<T>\* tail;

int size;

public:

List() {

head = NULL;

tail = NULL;

size = 0;

}

~List() {

clear();

}

T& get(int index);

T& operator[](int index) {

return get(index); // âîçâðàùàåì ññûëêó íà ýëåìåíò ïî èíäåêñó

}

// Ìåòîä äîáàâëåíèÿ ýëåìåíòà â êîíåö ñïèñêà

void push\_back(T k);

void pop(T key);

int list\_size() {

return size;

}

void clear();

void print() const;

friend std::ostream& operator<<(std::ostream& out, const List<T>& list)

{

list.print();

return out;

}

friend std::istream& operator>>(std::istream& in, List<T>& list)

{

int k;

in >> k;

list.push\_back(k);

return in;

1. };
2. Привести пример конструктора (с параметром, без параметров, копирования) для класса-группы Список.

template<typename T>

class List {

Node<T>\* head;

Node<T>\* tail;

int size;

public:

List() {

head = NULL;

tail = NULL;

size = 0;

}

// Конструктор с параметрами

List(T value) {

head = new Node<T>(value);

tail = head;

size = 1;

}

// Конструктор копирования

List(const List& other) {

// Копируем каждый узел списка other в новый список

head = NULL;

tail = NULL;

size = 0;

Node<T>\* temp = other.head;

while (temp != NULL) {

push\_back(temp->data);

temp = temp->next;

}

}

};

1. Привести пример деструктора для класса-группы Список.

template<typename T>

class List {

Node<T>\* head;

Node<T>\* tail;

int size;

public:

List() {

head = NULL;

tail = NULL;

size = 0;

}

~List() {

clear();

}

1. Привести пример метода для просмотра элементов для класса-группы Список.

template<typename T>

T& List<T>::get(int index)

{

Node<T>\* current = head; // íà÷èíàåì ñ ãîëîâû ñïèñêà

int i = 0;

while (i < index && current->next != NULL) {

current = current->next; // ïåðåõîäèì ê ñëåäóþùåìó óçëó

}

return current->key; // âîçâðàùàåì ññûëêó íà äàííûå óçëà

}

1. Какой вид иерархии дает группа?

Группа объектов может давать иерархию, например, если у нас есть группа объектов "Фигуры", то мы можем создать подклассы "Круг", "Прямоугольник", "Треугольник" и т.д., которые будут наследоваться от базового класса "Фигура".

1. Почему во главе иерархии классов, содержащихся в группе объектов должен находится абстрактный класс?

Абстрактный класс должен быть во главе иерархии классов, содержащихся в группе объектов, потому что он определяет общий интерфейс для всех подклассов и не может быть создан сам по себе. Это позволяет упростить работу с группой объектов и использовать полиморфизм для работы с различными типами объектов.

1. Что такое событие? Для чего используются события?

Событие - это сообщение о том, что что-то произошло в программе или в системе. События используются для уведомления других частей программы о произошедших изменениях или событиях, например, о нажатии кнопки пользователем или о завершении операции.

1. Какие характеристики должно иметь событие-сообщение?

Событие-сообщение должно иметь уникальный идентификатор, описание события, данные, связанные с событием (если есть) и время, когда событие произошло.

1. Привести пример структуры, описывающей событие.

struct TEvent

{

int what; // тип события

int command; // код команды

int message;

int a; // параметр команды

};

1. Полю what присваиваются значения, которые определяют тип события. Например:

0 – событие связано с мышью;

1 – событие связано с нажатием клавиши;

2 – событие связано с сообщением.

1. Полю command присваиваются значения, которые определяют код команды. Например:

0 – команда открытия файла;

1 – команда сохранения файла;

2 – команда печати документа.

Значения полей message и a используются для передачи параметров команды.

1. Поля "a" и "message" используются как параметры команды, которые могут быть переданы в структуру Kevent. Например, если тип события "what" указывает на выполнение определенной команды, то код команды может быть записан в поле "command", а ее параметры могут быть переданы в поля "message" и "a". Таким образом, эти поля используются для передачи дополнительной информации, необходимой для выполнения определенной команды.
2. Какие методы необходимы для организации обработки сообщений?

Для организации обработки сообщений необходимы методы, обрабатывающие конкретные типы событий, например, OnClick для обработки нажатия на кнопку, OnKeyDown для обработки нажатия клавиши и т.д.

1. Какой вид имеет главный цикл обработки событий-сообщений?

Главный цикл обработки событий-сообщений имеет вид:

int Dialog::Execute()

{

TEvent event;

do {

EndState = 0;

GetEvent(event);

HandleEvent(event);

} while (!Valid()); return EndState;

}

1. Какую функцию выполняет метод ClearEvent()? Каким образом?

Метод ClearEvent() очищает содержимое структуры TEvent, устанавливая все ее поля в значения по умолчанию.

1. Какую функцию выполняет метод HandleEvent ()?Каким образом?

Метод HandleEvent() выполняет обработку конкретного события в зависимости от его типа. Для этого он вызывает соответствующий метод, например, OnClick для обработки нажатия на кнопку, OnKeyDown для обработки нажатия клавиши и т.д.

1. Какую функцию выполняет метод GetEvent ()?

Метод GetEvent() получает следующее событие из очереди сообщений и возвращает его в виде структуры TEvent.

1. Для чего используется поле End State? Какой класс (объект) содержит это поле?

Поле End State используется для указания состояния приложения после завершения работы главного цикла обработки событий-сообщений. Это поле содержится в классе TApplication.

1. Для чего используется функция Valid()?

Функция Valid() используется для проверки корректности значений полей структуры TEvent. Она возвращает true, если значения полей находятся в допустимых пределах, и false в противном случае.