Пермский Национальный Исследовательский  
Политехнический Университет

**Лабораторная работа № 8**

Основы алгоритмизации и программирования

за 2 семестр

Вариант № 1

Выполнил:

Студент группы РИС 20-1-бз

Курганов Н.В.

20-ЭТФ-631

Проверила:

Доцент кафедры ИТАС

Полякова О.А.

2022

Программа, управления событиями

1. **Цель задания**.

1.1 Создание консольного приложения, состоящего из нескольких файлов в системе программирования VisualStudio.

1.2. Разработка программы, управляемой событиями.

1. **Задание.**

Базовый класс:

ЧЕЛОВЕК (Person)

Имя – string

Возраст – int

Производный класс

Студент (Student)

Рейтинг – float

Группа – Вектор (Vector)

Команды:

1. Создать группу (формат команды: m количество элементов группы)
2. Добавить элемент в группу (формат команды: +)
3. Удалить элемент из группы (формат команды: -)
4. Вывести информацию об элементах группы (формат команды: S)
5. Вывести информацию об элементе группы (формат команды: ? номер объекта в группе)
6. Конец работы формат команды: q)
7. **Создаем:**

**Файл lab8\_main.cpp**

#include "Vector.h"

#include "Dialog.h"

using namespace std;

void main()

{

Dialog D;

D.Execute();

}

**Файл Dialog.cpp**

#include "Dialog.h"

Dialog::Dialog(void): Vector()

{

EndState = 0;

}

//деструктор

Dialog::~Dialog(void)

{

}

//получение события

void Dialog::GetEvent(TEvent& event)

{

string OpInt = "+-?/sqam"; //строка содержит коды операций

string s;

string param;

char code;

cout << '>';

cin >> s; code = s[0];//первый символ команды

if (OpInt.find(code) >= 0)//является ли символ кодом операции

{

event.what = evMessage;

switch (code)

{

case 'm':event.command = cmMake; break;//создать группу

case '+': event.command = cmAdd; break;//добавить объект в группу

case '-': event.command = cmDel; break;//удалить объект из группы

case '?': event.command = cmShowByID; break;//просмотр группы

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

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

case'q': event.command = cmQuit; break; //конец работы

}

//выделяем параметры команды, если они есть

if (s.length() > 1)

{

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

int A = atoi(param.c\_str());//преобразуем парметр в число

event.a = A;//записываем в сообщение

}

}

else event.what = evNothing;//пустое событие

}

int Dialog::Execute()

{

TEvent event;

do {

EndState = 0;

GetEvent(event); //получить событие

HandleEvent(event); //обработать событие

} while (!Valid());

return EndState;

}

int Dialog::Valid()

{

if (EndState == 0) return 0;

else return 1;

}

void Dialog::ClearEvent(TEvent& event)

{

event.what = evNothing;//пустое событие

}

void Dialog::EndExec()

{

EndState = 1;

}

//обработчик событий

void Dialog::HandleEvent(TEvent& event)

{

if (event.what == evMessage)

{

switch (event.command)

{

case cmMake://создание группы

size = event.a; //размер группы

beg = new Object \* [size];//выделяем память под массив указателей

cur = 0; //текущая позиция

ClearEvent(event);//очищаем событие

break;

case cmAdd://добавление

Add();

ClearEvent(event);

break;

case cmDel:Del(); //удаление

ClearEvent(event);

break;

case cmShow:Show(); //просмотр

ClearEvent(event);

break;

case cmShowByID:Show(event.a); //просмотр по индексу

ClearEvent(event);

break;

case cmQuit:EndExec(); //выход

ClearEvent(event);

break;

default:Vector::HandleEvent(event);

};

};

}

**Файл Human.cpp**

#include <string>

#include "Human.h"

Human::Human(void)

{

name = "";

age = 0;

}

Human::Human(string n, int a)

{

name = n;

age = a;

}

Human::Human(const Human&h)

{

name = h.name;

age = h.age;

}

Human::~Human(void)

{

}

void Human::Show()

{

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

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

cout << "\n";

}

void Human::Input()

{

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

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

}

void Human::Set\_name(string n)

{

name = n;

}

void Human::Set\_age(int a)

{

age = a;

}

Human& Human::operator=(const Human&h)

{

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

name = h.name;

age = h.age;

return \*this;

}

void Human::HandleEvent(const TEvent& e)

{

if (e.what == evMessage)//событие-сообщение

{

switch (e.command)

{

case cmGet:cout << "name=" << Get\_name() << endl;

break;

}

}

}

**Файл Object.cpp**

#include "Object.h"

Object::Object(void)

{

}

Object::~Object(void)

{

}

**Файл Student.cpp**

#include <string>

#include "Student.h"

Student::Student(void): Human()

{

rate = 0;

}

Student::~Student(void)

{

}

Student::Student(string n, int a, float r) :Human(n, a)

{

rate = r;

}

Student::Student(const Student& s)

{

name = s.name;

age = s.age;

rate = s.rate;

}

void Student::Set\_rate(float r)

{

rate = r;

}

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

{

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

name = s.name;

age = s.age;

rate = s.rate;

return \*this;

}

void Student::Show()

{

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

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

cout << "\nRate : " << rate;

cout << "\n";

}

void Student::Input()

{

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

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

cout << "\nRate : "; cin >> rate;

}

**Файл Vector.cpp**

#include "Vector.h"

#include <iostream>

using namespace std;

Vector::Vector()

{

beg = new Object \* [0];

cur = 0;

size = 0;

}

Vector::~Vector(void)

{

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

beg = 0;

}

//конструктор с параметрами

Vector::Vector(int n)

{

beg = new Object \* [n];

cur = 0;

size = n;

}

//добавление объекта, на который указывает указатель p в вектор

void Vector::Add()

{

Object\* p;

//выбор из объектов двух возможных классов

cout << "1.Human" << endl;

cout << "2.Student" << endl;

int y;

cin >> y;

if (y == 1)//добавление объекта класса Human

{

Human\* a = new (Human);

a->Input();//ввод значений атрибутов

p = a;

if (cur < size)

{

beg[cur] = p;//добавление в вектор

cur++;

}

}

else

if (y == 2) //добавление объекта класса Student

{

Student\* b = new Student;

b->Input();

p = b;

if (cur < size)

{

beg[cur] = p;

cur++;

}

}

else return;

}

//просмотр вектора

void Vector::Show()

{

if (cur == 0) cout << "Empty" << endl;

Object\*\* p = beg;//указатель на указатель типа Object

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

{

(\*p)->Show();//вызов метода Show() (позднее связывание)

p++;//передвигаем указатель на следующий объект

}

}

//просмотр вектора

void Vector::Show(int index)

{

if (cur == 0) cout << "Empty" << endl;

Object\*\* p = beg;//указатель на указатель типа Object

for (int i = 0; i < index; i++, p++);

(\*p)->Show();

}

//операция, которая возвращает размер вектора

int Vector::operator ()()

{

return size;

}

//удаление элемента из вектора, память не освобождается!

void Vector::Del()

{

if (cur == 0)return;//пустой

cur--;

}

void Vector::HandleEvent(const TEvent& e)

{

if (e.what == evMessage)

{

Object\*\* p = beg;

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

{

(\*p)->HandleEvent(e);//вызов метода (позднее связывание)

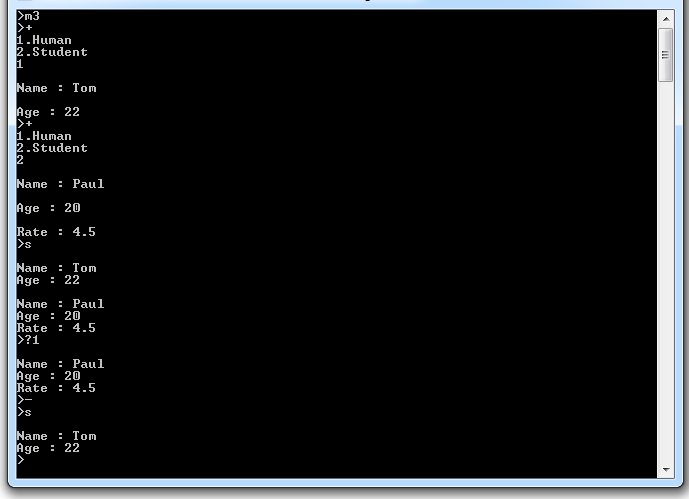
p++;//передвигаем указатель на следующий объект

}

}

}

1. **Результат работы программы**

****