ПНИПУ

Федеральное государственное автономное образовательное учреждение высшего образования «Пермский национальный исследовательский политехнический университет»

Лабораторная работа №8.

Программа, управляемая событиями

Выполнил студент группы РИС-23-3Б

Буковский Денис Владимирович

Проверила доцент кафедры ИТАС О.А. Полякова

2024

1. Постановка задачи

1. Определить иерархию пользовательских классов (см. лабораторную работу №5). Во главе иерархии должен стоять абстрактный класс с чисто виртуальными методами для ввода и вывода информации об атрибутах объектов.

2. Реализовать конструкторы, деструктор, операцию присваивания, селекторы и модификаторы.

3. Определить класс-группу на основе структуры, указанной в варианте.

4. Для группы реализовать конструкторы, деструктор, методы для добавления и ‘удаления элементов в группу, метод для просмотра группы, перегрузить операцию для получения информации о размере группы.

5. Определить класс Диалог — наследника группы, в котором реализовать методы для обработки событий.

6. Добавить методы для обработки событий группой и объектами пользовательских классов.

7. Написать тестирующую программу.

8. Нарисовать диаграмму классов и диаграмму объектов.

Базовый класс: ПЕЧАТНОЕ ИЗДАНИЕ(PRINT)

Название- string

Автор — string

Производный класс ЖУРНАЛ (MAGAZINE)

Количество страниц – int

Группа — Вектор (Vector).

Команды:

Создать группу (формат команды: m количество элементов группы).

Добавить элемент в группу (формат команды: +)

Удалить элемент из группы (формат команды -)

Вывести информацию об элементах группы (формат команды: s)

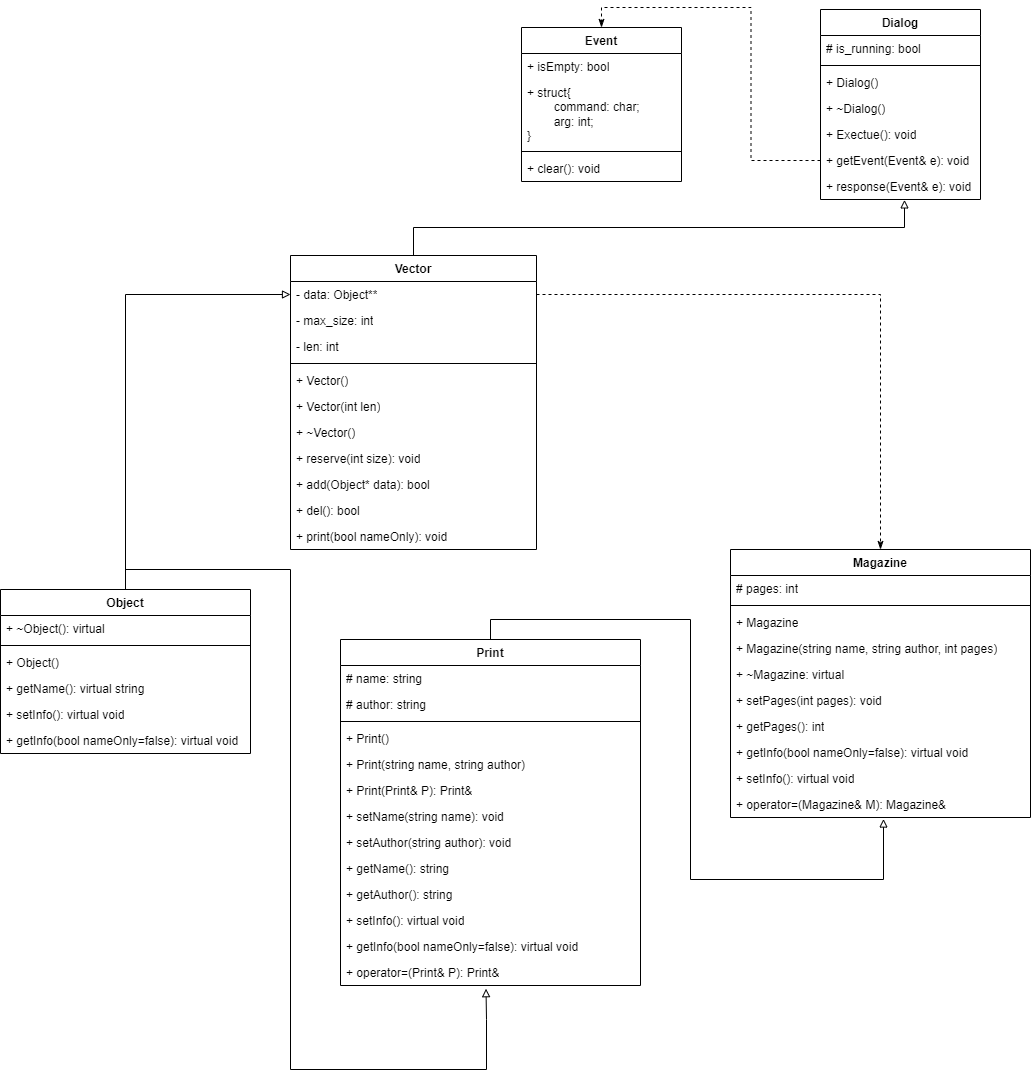
Вывести информацию о названиях всех элементов группы (формат команды : z)

Конец работы (формат команды: q)

1. Анализ задачи

Создадим класс Dialog, от него унаследуем класс Vector, оперировать будем объектами Magazine и Print, унаследованными от класса Object. Напишем определение функций. Запустим цикл событий в функции main().

1. Блок-схема



1. Код

Event.h

#pragma once

#include "Vector.h"

const bool ev\_empty = true;

const bool new\_message = false;

const char createGroup = 'm';

const char addElem = '+';

const char delElem = '-';

const char printData = 's';

const char printNames = 'z';

const char stopExec = 'q';

class Event

{

public:

bool isEmpty;

struct {

char command; //command with argument

int arg;

};

void clear() {

isEmpty = ev\_empty;

command = ' ';

arg = 0;

}

};

class Dialog : public Vector {

protected:

bool is\_running;

public:

Dialog();

~Dialog();

void execute();

void getEvent(Event&);

void response(Event&);

};

Event.cpp

#include "Event.h"

#include "Print.h"

#include <iostream>

#include <string>

using namespace std;

Dialog::Dialog() {

this->is\_running = true;

}

Dialog::~Dialog() {}

void Dialog::execute() {

Event ev;

while (this->is\_running) {

cout << "Pick an action for program:" << endl

<< "'m' - create new group; args: group size" << endl

<< "'+' - add element to current group; args: 1 - Magazine, 2 - Print" << endl

<< "'-' - delete current group" << endl

<< "'s' - print data of current group" << endl

<< "'z' - print only names of current group" << endl

<< "'q' - stop listening to input" << endl

<< endl << "~$ ";

this->getEvent(ev);

this->response(ev);

cout << endl;

}

}

void Dialog::getEvent(Event& ev) {

string cmds = "m+-szq", input;

getline(cin, input);

if (cmds.find(input[0]) >= 0) {

ev.isEmpty = new\_message;

switch (input[0])

{

case 'm': ev.command = createGroup; break;

case '+': ev.command = addElem; break;

case '-': ev.command = delElem; break;

case 's': ev.command = printData; break;

case 'z': ev.command = printNames; break;

case 'q': ev.command = stopExec; break;

}

if (input.length() > 1) {

int arg = atoi(input.substr(1, input.size() - 1).c\_str());

ev.arg = arg;

if (arg <= 0) {

cout << "Wrong input!" << endl;

ev.isEmpty = ev\_empty;

}

}

}

else ev.isEmpty = ev\_empty;

}

void Dialog::response(Event& ev) {

if (!ev.isEmpty) {

switch (ev.command) {

case createGroup: this->reserve(ev.arg);

break;

case addElem:

if (ev.arg == 1) {

Magazine\* elem = new Magazine;

elem->setInfo();

this->add(elem);

}

else {

Print\* elem = new Print;

elem->setInfo();

this->add(elem);

}

break;

case delElem: this->del(); break;

case printData: this->print(false); break;

case printNames: this->print(true); break;

case stopExec: this->is\_running = false;

default: break;

}

}

ev.clear();

}

Vector.h

#pragma once

#include "Object.h"

class Vector

{

protected:

Object\*\* data;

int max\_size;

int len;

public:

Vector();

Vector(int);

~Vector();

void reserve(int);

bool add(Object\*);

bool del();

void print(bool);

};

Vector.cpp

#include "Vector.h"

#include <iostream>

using namespace std;

Vector::Vector() {

this->data = nullptr;

this->max\_size = 0;

this->len = 0;

}

Vector::Vector(int data) {

this->data = new Object \* [data];

this->max\_size = data;

this->len = 0;

}

Vector::~Vector() {}

void Vector::reserve(int length) {

if (!(this->data == nullptr)) {

for (int i = 0; i < this->len; i++) {

delete this->data[i];

}

delete[] this->data;

}

this->max\_size = length;

this->data = new Object\*[length];

this->len = 0;

};

bool Vector::add(Object\* obj) {

if (this->len >= this->max\_size) {

cout << "GroupOverflow Error!" << endl;

return false;

}

this->data[this->len] = obj;

this->len++;

return true;

}

bool Vector::del() {

string name;

cout << "Enter name of element to delete: ";

cin >> name;

if (this->len == 0) {

cout << "NoElements Error!" << endl;

return false;

}

bool f = false;

for (int i = 0; i < this->len && !f; i++) {

if (this->data[i]->getName() == name) {

Object\* tmp = this->data[i];

for (int j = i; i < this->len-1; j++) {

this->data[j] = this->data[j + 1];

}

delete tmp;

this->len--;

}

}

if (!f) cout << "NotFound Error!" << endl;

return f;

}

void Vector::print(bool nameOnly = false) {

cout << "class 'Vector' information:" << endl;

if (this->len == 0) {

cout << "This group is empty!" << endl;

return;

}

this->data[0]->getInfo(nameOnly);

for (int i = 1; i < this->len; i++) {

cout << endl;

this->data[i]->getInfo(nameOnly);

}

}

Object.h

#pragma once

#include <iostream>

class Object

{

public:

virtual ~Object() = 0 {}

Object() {}

virtual std::string getName() = 0;

virtual void setInfo() = 0;

virtual void getInfo(bool names\_only = false) = 0;

};

Print.h

#pragma once

#include "Object.h"

#include <iostream>

using namespace std;

class Print : public Object

{

protected:

string name;

string author;

public:

Print() {}

Print(string, string) {}

virtual ~Print() {}

void setName(string name) { this->name = name; }

void setAuthor(string author) { this->author = author; }

string getName() { return this->name; }

string getAuthor() { return this->author; }

virtual void setInfo();

virtual void getInfo(bool);

Print& operator=(Print&);

};

class Magazine : public Print

{

protected:

int pages;

public:

Magazine() :Print() { this->pages = 0; }

Magazine(string, string, int);

virtual ~Magazine() {}

void setPages(int pages) { this->pages = pages; }

int getPages() { return pages; }

virtual void setInfo();

virtual void getInfo(bool);

Magazine& operator=(Magazine&);

};

Print.cpp

#include "Print.h"

#include <iostream>

#include <string>

using namespace std;

void Print::setInfo() {

cout << "Input name: ";

getline(cin, this->name);

cout << "Input author: ";

getline(cin, this->author);

}

void Print::getInfo(bool nameOnly=true) {

cout << "class 'Print' information:" << endl;

cout << "Name: " << getName() << endl;

if (!nameOnly) {

cout << "Author: " << getAuthor() << endl;

}

}

Print& Print::operator=(Print& P) {

this->name = P.name;

this->author = P.author;

return \*this;

}

Magazine::Magazine(string name, string author, int pages) {

this->name = name;

this->author = author;

this->pages = pages;

}

void Magazine::setInfo() {

cout << "Input name: ";

getline(cin, this->name);

cout << "Input author: ";

getline(cin, this->author);

cout << "Input pages count: ";

cin >> this->pages;

}

void Magazine::getInfo(bool nameOnly = true){

cout << "class 'Magazine' information:" << endl;

cout << "Name: " << getName() << endl;

if (!nameOnly) {

cout << "Author: " << getAuthor() << endl;

cout << "Pages count: " << getPages() << endl;

}

}

Magazine& Magazine::operator=(Magazine& M) {

this->name = M.name;

this->author = M.author;

this->pages = M.pages;

return \*this;

}

Main.cpp

#include <iostream>

#include "Event.h";

int main() {

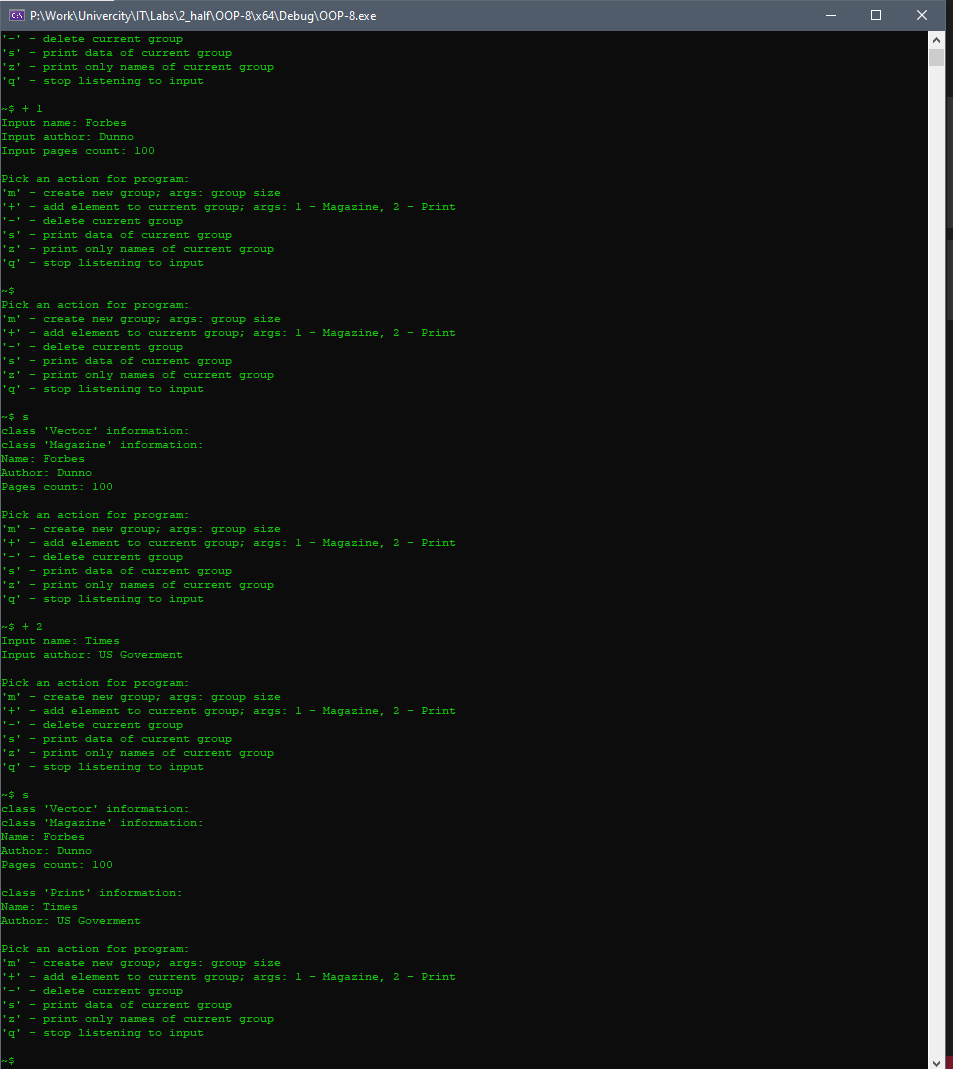
Dialog ui;

ui.execute();

return 0;

}

1. Вывод программы



1. Ответы на контрольные вопросы

1. Класс-группа – это класс, содержащий в себе набор элементов. Прим.: класс STL vector.

2. List<int> data;

3. List(){}

List(int count, int num …){for (int I = 0; i<count; i++){int\*ptr = &num; data.push\_back(\*ptr); ptr++;}}

List(List&lst){data = lst.data;}

4. ~List () {}

5. Проход итератором.

6. иерархию объектов.

7. Для обеспечения принципа подстановки.

8. Событие – это объект, содержащий в себе информацию от внешних источников (ввод пользователя, показатели датчиков) для ответа от программы.

9. Событие-сообщение должно содержать корректный код события и некоторую информацию.

10.

#pragma once

#include "Vector.h"

const bool ev\_empty = true;

const bool new\_message = false;

const char createGroup = 'm';

const char addElem = '+';

const char delElem = '-';

const char printData = 's';

const char printNames = 'z';

const char stopExec = 'q';

class Event

{

public:

bool isEmpty;

struct {

char command; //command with argument

int arg;

};

void clear() {

isEmpty = ev\_empty;

command = ' ';

arg = 0;

}

};

11. целочисленные.

12. целочисленные.

13. Определяют тип сообщения и аргумент.

14. getEvent(), response(), execute()

15. бесконечного цикла приёма сообщения и генерации ответа.

16. ClearEvent() очищает объект события, сбрасывая все значения к первоначальным.

17. HandleEvent() – набор switch-case-ов, отвечающий за действия программы.

18. GetEvent() – Метод, отвечающий за ввод события и решение обрабатывать его или нет.

19. EndState – Boolean переменная, отвечающая за работу программы. Если сменить её значения, то цикл прервётся.

20. Valid() проверяет введённые данные на правильность.