Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
Пермский национальный исследовательский политехнический университет
Электротехнический факультет
Кафедра информационных технологий и автоматизированных систем

Лабораторная работа №8

" Программа, управляемая событиями "

Вариант: 12

Выполнил студент ИВТ-24-2б:
Шишкин Максим Григорьевич

_____
(дата, подпись)

Проверил доцент кафедры ИТАС:

Полякова Ольга Андреевна

_____
(дата, подпись)

Пермь 2025

Содержание

1 Постановка задачи 3-4

2 Код на C++ 5-7

3 Результаты работы 8

4 UML-диаграмма классов 8

5 Ответы на контрольные вопросы 8-11

6 Ссылка на github..... 11

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.

Вариант: Базовый класс: ЧЕЛОВЕК (Person)

Имя – string

Возраст – int

Производный класс СОТРУДНИК (Emloyee)

Заработная плата – float

Должность - string

Группа – Дерево (Tree).

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 об имени элемента с номером k (формат команды : z k, где k – целое число)
- Конец работы (формат команды: q)

2 Код на C++

Person.h | Person.cpp | Employee.h | Employee.cpp | Tree.h | Tree.cpp | Dialog.h | Dialog.cpp | main.cpp

8лабапеределка — (Глобальная область)

```cpp
#include "Dialog.h"
#include <locale>

int main() {
    setlocale(LC_ALL, "ru");
    Dialog d;
    d.Run();
    return 0;
}
```

Person.h | Person.cpp | Employee.h | Employee.cpp | Tree.h | Tree.cpp | Dialog.h | Dialog.cpp | main.cpp

8лабапеределка — (Глобальная область)

```cpp
#include "Dialog.h"
#include "Employee.h"
#include <iostream>
#include <string>

using namespace std;

void Dialog::Run() {
    cout << "Команды: m N | + | - | s | z k | q\n";
    string cmd;
    while (true) {
        cout << "> ";
        cin >> cmd;
        if (cmd == "q") break;
        else if (cmd == "m") {
            int n;
            cin >> n;
            for (int i = 0; i < n; ++i) {
                cout << "Добавление элемента " << i + 1 << ":\n";
                auto* e = new Employee();
                e->Input();
                Add(e);
                delete e;
            }
        }
        else if (cmd == "+") {
            auto* e = new Employee();
            e->Input();
            Add(e);
            delete e;
        }
        else if (cmd == "-") {
            RemoveLast();
        }
        else if (cmd == "s") {
            ShowAll();
        }
        else if (cmd == "z") {
            int k;
            cin >> k;
            ShowName(k);
        }
        else {
            cout << "Неизвестная команда\n";
            cin.ignore(1000, '\n'); // очистка ввода
        }
    }
}
```

Person.h | Person.cpp | Employee.h | Employee.cpp | Tree.h | Tree.cpp | **Dialog.h** | Dialog.cpp | main.cpp

8лабапеределка — (Глобальная область)

```
1  #pragma once
2  #include "Tree.h"
3
4  class Dialog : public Tree {
5  public:
6      void Run();
7  };
```

Person.h | Person.cpp | Employee.h | Employee.cpp | Tree.h | **Tree.cpp** | Dialog.h | Dialog.cpp | main.cpp

8лабапеределка — (Глобальная область)

```
1  #include "Tree.h"
2  #include <iostream>
3
4  Tree::Tree() = default;
5
6  Tree::~Tree() {
7      for (Person* p : nodes) delete p;
8  }
9
10 void Tree::Add(Person* p) {
11     nodes.push_back(p->Clone());
12 }
13
14 void Tree::RemoveLast() {
15     if (!nodes.empty()) {
16         delete nodes.back();
17         nodes.pop_back();
18     }
19 }
20
21 void Tree::ShowAll() const {
22     for (const Person* p : nodes)
23         p->Show();
24 }
25
26 int Tree::Size() const {
27     return nodes.size();
28 }
29
30 void Tree::ShowName(int k) const {
31     if (k >= 0 && k < nodes.size())
32         std::cout << "Имя [" << k << "]: " << nodes[k]->GetName() << std::endl;
33     else
34         std::cout << "Неверный индекс\n";
35 }
```

Person.h | Person.cpp | Employee.h | Employee.cpp | **Tree.h** | Tree.cpp | Dialog.h | Dialog.cpp | main.cpp

8лабапеределка — (Глобальная область)

```
1  #pragma once
2  #include "Person.h"
3  #include <vector>
4
5  class Tree {
6  protected:
7      std::vector<Person*> nodes;
8
9  public:
10     Tree();
11     virtual ~Tree();
12
13     virtual void Add(Person* p);
14     virtual void RemoveLast();
15     virtual void ShowAll() const;
16     virtual int Size() const;
17     virtual void ShowName(int k) const;
18 };
```

Person.h | Person.cpp | Employee.h | **Employee.cpp** | Tree.h | Tree.cpp | Dialog.h | Dialog.cpp | main.cpp

8лабапеределка | (Глобальная область)

```cpp
#include "Employee.h"
#include <iostream>

using namespace std;

Employee::Employee() : Person(), salary(0), position("") {}
Employee::Employee(string n, int a, float s, string p)
    : Person(n, a), salary(s), position(p) {}

void Employee::Input() {
    cout << "Имя: "; cin >> name;
    cout << "Возраст: "; cin >> age;
    cout << "Зарплата: "; cin >> salary;
    cout << "Должность: "; cin >> position;
}

void Employee::Show() const {
    cout << "Сотрудник: " << name << ", " << age
        << " лет, зарплата: " << salary
        << ", должность: " << position << endl;
}

string Employee::GetName() const {
    return name;
}

Person* Employee::Clone() const {
    return new Employee(*this);
}
```

Person.h | Person.cpp | **Employee.h** | Employee.cpp | Tree.h | Tree.cpp | Dialog.h | Dialog.cpp | main.cpp

8лабапеределка | (Глобальная область)

```cpp
#pragma once
#include "Person.h"
#include <string>

class Employee : public Person {
    float salary;
    std::string position;

public:
    Employee();
    Employee(std::string n, int a, float s, std::string p);

    void Input() override;
    void Show() const override;
    std::string GetName() const override;
    Person* Clone() const override;
};
```

Person.h | **Person.cpp** | Employee.h | Employee.cpp | Tree.h | Tree.cpp | Dialog.h | Dialog.cpp | main.cpp

8лабапеределка | (Глобальная область)

```cpp
#include "Person.h"

Person::Person() : name(""), age(0) {}
Person::Person(std::string n, int a) : name(n), age(a) {}
```

Person.h | Person.cpp | Employee.h | Employee.cpp | Tree.h | Tree.cpp | Dialog.h | Dialog.cpp | main.cpp

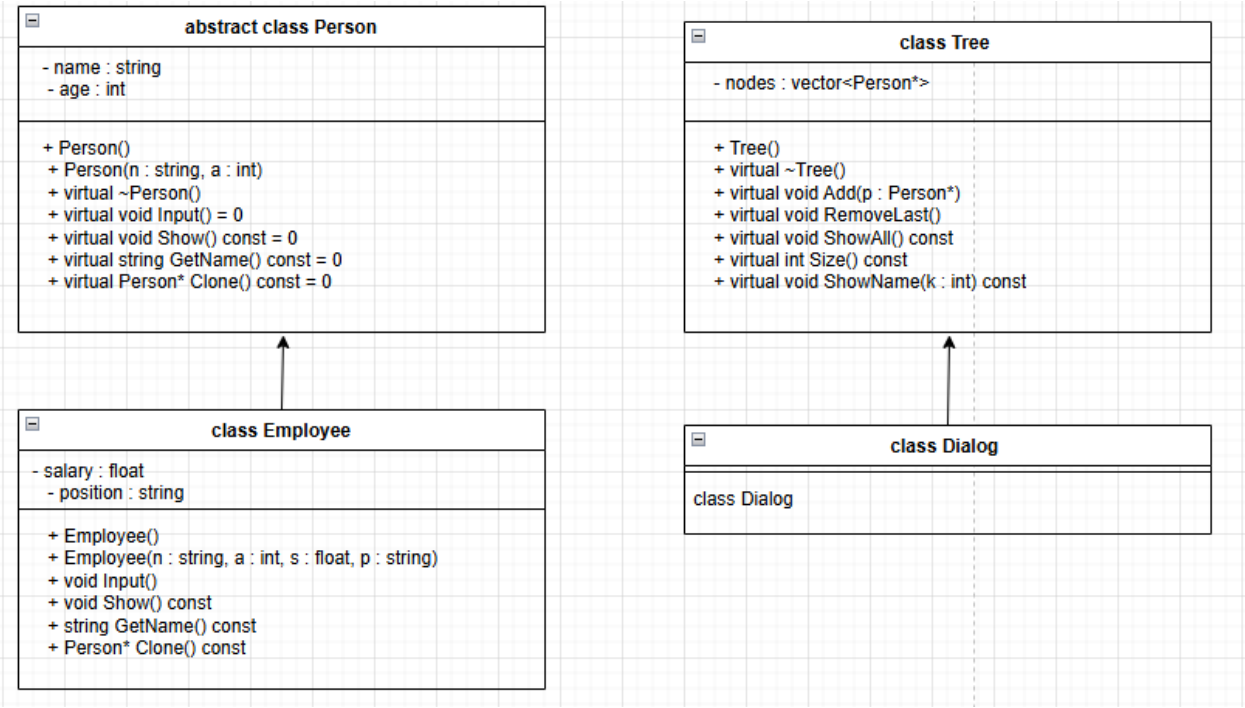
8лабапеределка | (Глобальная область)

```cpp
#pragma once
#include <string>

class Person {
protected:
    std::string name;
    int age;

public:
    Person();
    Person(std::string n, int a);
    virtual ~Person() = default;

    virtual void Input() = 0;
    virtual void Show() const = 0;
    virtual std::string GetName() const = 0;
    virtual Person* Clone() const = 0;
};
```

3 Результаты работы

```
Команды: m N | + | - | s | z k | q
> m 1
Добавление элемента 1:
Имя: bob
Возраст: 22
Зарплата: 1200
Должность: man
> z 0
Имя [0]: bob
> q
```

4 UML-диаграмма классов



5 Ответы на контрольные вопросы

1 Класс-группа — это класс, хранящий несколько объектов других классов. Примеры: List, Tree, Array, Vector, Dialog.

2
```
class List {
private:
  vector<int> data;
public:
  void Add(int value);
  void Show() const;
};
```

3
```
List() {} // без параметров
List(const vector<int>& v) : data(v) {} // с параметром
List(const List& other) : data(other.data) {} // копирования
```

4
```
~List() {
  data.clear();
}
```

5
```
void Show() const {
  for (int val : data)
    cout << val << " ";
}
```

6 Группа даёт иерархию композиции — один класс включает в себя объекты других (например, Tree содержит Person*).

7 Абстрактный класс задаёт общий интерфейс — это позволяет обрабатывать разные производные объекты через один тип.

8 Событие — это сигнал о действии (команда, ввод и т.д.). Используются для управления поведением программы.

9 Событие должно содержать тип, код команды и параметры — чтобы точно передавать и обрабатывать действия.

10 struct TEvent {

```
  int what;     // тип события

  int command;  // код команды

  int param;    // параметр
```

};

11 Поле what получает значения:

- evNothing — нет события
- evMessage — сообщение (команда)
- evKeyDown — нажатие клавиши
- evMouse — мышь

12 Поле command получает код команды: например, +, -, s, z, q — в зависимости от ввода пользователя.

13

- a — параметр команды, например, индекс элемента.
- message — код сообщения, например cmAdd, cmDelete и т.д.

14 Необходимые методы:

- GetEvent() — получение события
- HandleEvent() — обработка
- ClearEvent() — сброс события

15 while (!EndState) {

GetEvent(e);

HandleEvent(e);

}

16 ClearEvent() очищает структуру события — сбрасывает what = evNothing.

17 HandleEvent() анализирует тип и код события и вызывает соответствующие методы (Add, Show, Remove).

18 GetEvent() считывает команду пользователя с клавиатуры и формирует структуру TEvent.

19 Поле EndState используется для завершения работы цикла обработки событий. Оно находится в классе Dialog.

20 Функция Valid() проверяет корректность параметров (например, номер элемента не выходит за границы).

6 Ссылка на github

ссылка на github - https://github.com/MAKSPOWERO/mas1