Министерство образования Республики Беларусь

Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

**Лабораторная работа №6.**
"Наследование и виртуальные функции”

Выполнил:
Ст. 2 курса гр. АС-53
Анискин Д.В.
Проверила:
Давидюк Ю. И.

Брест, 2020

**1. Цель.** Получить практические навыки создания иерархии классов и использования статических компонентов класса.

## 2. Постановка задачи (Вариант 3)

Написать программу, в которой создается иерархия классов. Включить полиморфные объекты в связанный список, используя статические компоненты класса. Показать использование виртуальных функций.

**служащий, персона, рабочий, инженер;**

Классы: Employee, Person, Worker, Engineer
Конструкторы:

- Пустой
- С параметрами
- Копирования

Деструктор.
Виртуальные фукнции:

- Добавления в список
- Вывода информации

**3.** Иерархия классов в виде графа:

- Персона
  - Рабочий
    - Инженер
  - Служащий

**4.** Определение пользовательских классов с комментариями.

```
 //Базовый класс
class Person  // базовый класс
{
public:
	static  Person* begin;   //указатель на начало списка
	Person* next = NULL;
	static void ShowList()  //список
	{
		Person* p = begin;
		while (p)
		{
			p->show();
			p = p->next;
		}
	}
	Person()  //без параметров
	{
		name = new char[81];
	}
	Person(const char* NAME, int age1)  //с параметрами
	{
		// выделение памяти для name. размер выделяемой памяти = длина строки NAME
		name = new char[strlen(NAME) + 1];
```

```
                strcpy(name, NAME);
                age = age1;

        }
        virtual ~Person()  // виртуальный деструктор
        {
                delete[] name;
        }
        virtual void show() = 0;  //Чистая виртуальная функция
        virtual void input() = 0;

protected:
        char* name;
        int age;
};

Person* Person::begin = NULL; //Инициализация статической компоненты

class Worker :public Person   // производный класс
{
public:
        Worker() : Person() {}   //без параметров
        Worker(const char* NAME, int age1, bool AddToList = false) :Person(NAME, age1) //с
параметрами
        {
                if (AddToList)
                {
                        Person* p = begin;
                        while (p->next)
                        {
                                p = p->next;
                        }
                        p->next = this;
                }
        }
        void show()
        {
                cout << "\nКласс: Рабочий";
                cout << "\nИмя: " << name;
                cout << "\nВозраст:" << age;
                cout << "\n";
        }
        void input()
        {
                cout << "\nИмя Рабочего: ";
                cin >> name;
                cout << "\nВозраст :";
                cin >> age;
                cout << "\n";
        }

};

class Employee :public Person  // производный класс
{
public:
        Employee() : Person() {}

        Employee(const char* NAME, int age1, bool AddToList = false) :Person(NAME, age1)
        {
                if (AddToList)
                {
                        Person* p = begin;
                        while (p->next)
                        {
```

```
                        p = p->next;
                }
                p->next = this;
        }

    }
    void show()
    {
        cout << "\nКласс: Служащий";
        cout << "\nИмя: " << name;
        cout << "\nВозраст:" << age;
        cout << "\n";
    }
    void input()
    {
        cout << "\nИмя Служащего: ";
        cin >> name;
        cout << "\nВозраст :";
        cin >> age;
        cout << "\n";
    }

};

class Engineer :public Worker // производный класс
{
public:
    Engineer() : Worker() {}

    Engineer(const char* NAME, int age1, bool AddToList = false) :Worker(NAME, age1)
    {
        if (AddToList)
        {
            Person* p = begin;
            while (p->next)
            {
                p = p->next;
            }
            p->next = this;
        }
    }
    void show()
    {
        cout << "\nКласс: Инженер";
        cout << "\nИмя: " << name;
        cout << "\nВозраст:" << age;
        cout << "\n";
    }
    void input()
    {
        cout << "\nИмя Инженера: ";
        cin >> name;
        cout << "\nВозраст :";
        cin >> age;
        cout << "\n";
    }

};
```

Реализация конструкторв с параметрами

и
деструктора.

- Для класса Person:

```
Person()  //без параметров
	{
		name = new char[81];
	}
	Person(const char* NAME, int age1)  //с параметрами
	{
		// выделение памяти для name. размер выделяемой памяти = длина строки NAME
		name = new char[strlen(NAME) + 1];
		strcpy(name, NAME);
		age = age1;

	}
	~Person()  // деструктор
	{
		delete[] name;
	}
```

- Для класса Worker:

```
	worker() : person() {}   //без параметров
	worker(const char* NAME, int age1, bool AddToList = false) :person(NAME, age1) //с
	параметрами
	{
		if (AddToList)
		{
			person* p = begin;
			while (p->next)
			{
				p = p->next;
			}
			p->next = this;
		}
	}
	~worker()  // деструктор
	{
		cout << "worker object deleted" << endl;
	}
```

- Для класса Employee:

```
	employee() : person() {}

	employee(const char* NAME, int age1, bool AddToList = false) :person(NAME, age1)
	{
		if (AddToList)
		{
			person* p = begin;
			while (p->next)
			{
				p = p->next;
			}
			p->next = this;
		}

	}
	~employee()  // деструктор
	{
		cout << "employee object deleted" << endl;
	}
```

- Для класса Engeneer:

```
engineer() : worker() {}

engineer(const char* NAME, int age1, bool AddToList = false) :worker(NAME, age1)
{
	if (AddToList)
	{
		person* p = begin;
		while (p->next)
		{
			p = p->next;
		}
		p->next = this;
	}
}
```

## 5. Реализация метода для просмотра списка.

```
void show()
	{
		cout << "\nКласс: Служащий";
		cout << "\nИмя: " << name;
		cout << "\nВозраст:" << age;
		cout << "\n";
	}
```

## 6. Листинг демонстрационной программы.

```
int main()
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	worker* a1;
	employee* a2;
	engineer* a3;
	a1 = new worker;
	a2 = new employee;
	a3 = new engineer;
	a1->input();
	a2->input();
	a3->input();
	cout << "---------------------------------\n";
	person::begin = a1;
	a1->next = a2;
	a2->next = a3;
	engineer* x4 = new engineer("Dimon-patron", 20, true); // Создание объекта класса
	person::ShowList();

	return 0;
}
```

# 8.Вывод программы:

```
---------------------------------

Класс: Рабочий
Имя: Vitalii
Возраст:21

Класс: Служащий
Имя: Vasilii
Возраст:34

Класс: Инженер
Имя: Petya
Возраст:24

Класс: Инженер
Имя: Dimon-patron
Возраст:20
```

**Объяснение необходимости виртуальных функций. Следует показать,какие**

**результаты будут в случае виртуальных и не виртуальных функций.**
*При наследовании бывает необходимо, чтобы поведение некоторых методов базового класса и классов-наследников различалось, именно для этого и требуется наличие виртуальных функций* `virtual void Show() = 0;virtual void Add() = 0;`
*В данном коде, в случае отсутствия виртуальной функции нельзя будет переопределить поведение.*

**9.Вывод:**

Получил практические навыки реализации классов на C++.