Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

Кафедра ПЗ

Лабораторна робота №7

з дисципліни «Операційні системи»

з теми «Спадчина і віртуальні функції»

Виконали: ст. 1ПІ-21Б Миронюк О.В.

Гиренко В.В.

Коцюбняк В.А.

Максименко О.В.

Перевірив: Рейда О. М.

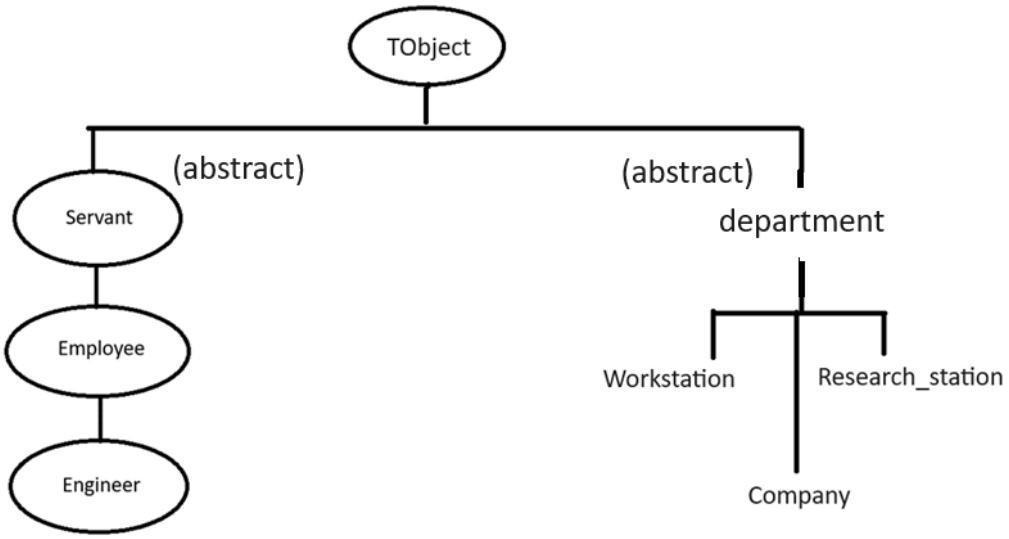
Вінниця – 2023

Мета роботи: Отримати практичні навички створення об'єктів - груп та використання методів - ітераторів.

Теоретичні відомості

Завдання: Написати демонстраційну програму, в якій створюються, показуються та руйнуються об'єкти-групи, а також демонструється використання ітератора.

Граф ієрархії класів, де TObject – базовий клас:



Код програми

Модуль TObject.h

```
#pragma once
#ifndef TOBJECT_H
#define TOBJECT_H
class TObject
{
public:
	virtual void print() = 0;
};
#endif
```

Модуль SERVANT.h

```
#pragma once
#include <string>

#include "TObject.h"

class servant : public TObject
{
protected:
	std::string name_;
	int age_;
	int experience_;
	servant* next;
public:

	servant(std::string, int, int);
	servant(servant&);

	//Properties
	__declspec(property (put = set_name, get = get_name)) std::string Name;
	__declspec(property (put = set_age, get = get_age)) int Age;
	__declspec(property (put = set_experience, get = get_experience)) int
Experience;

	//Accessors
	std::string get_name();
	void set_name(std::string name);
	int get_age();
	void set_age(int age);
	int get_experience();
	void set_experience(int experience);

	//Functions
	void print();
	void add(servant* s);
	static bool check_is_older(servant s, int n);
};
```

Модуль SERVANT.cpp

```
#include "servant.h"

#include <iostream>

//Accessors
std::string servant::get_name()
{
	return name_;
}

void servant::set_name(const std::string name)
{
	name_ = name;
}

int servant::get_age()
{
	return age_;
```

```
}

void servant::set_age(const int age)
{
	age_ = age;
}

int servant::get_experience()
{
	return experience_;
}

void servant::set_experience(const int experience)
{
	experience_ = experience;
}

//Constructors
servant::servant(std::string name = "Default_name", int age = 18, int experience
= 0)
{
	name_ = name;
	age_ = age;
	experience_ = experience;
};
servant::servant(servant& s) : servant(s.Name, s.Age, s.Experience) {}

void servant::print()
{
	std::cout << "Servant: Name: " << this->Name << "; Age: " << this->Age <<
"; Experience: " << this->Experience << ";\n";
}

void servant::add(servant* s) {
	servant* el = this;
	while (el->next) {
		el = el->next;
	}
	el->next = s;
}

bool servant::check_is_older(servant s, int n) {
	if (s.Age >= n) {
		return true;
	}
	return false;
	}
```

Модуль engineer.h

```
#pragma once
#include "employee.h"

class engineer :
	public employee
{
public:
	engineer* next;

	using employee::employee;
	virtual void print();
```

```
	void add(engineer* e);
};
```

Модуль engineer.cpp

```
#include "engineer.h"

#include <iostream>

void engineer::print()
{
	std::cout << "Engineer: Name: " << this->Name << "; Age: " << this->Age <<
"; Experience: " << this->Experience << ";\n";
}

void engineer::add(engineer* e) {
	engineer* el = this;
	while (el->next) {
		el = el->next;
	}
	el->next = e;
}
```

Модуль employee.h

```
#pragma once
#include "servant.h"

class employee :
	public servant
{
public:
	using servant::servant;
	~employee(){}

	employee* next;

	virtual void print();
	void add(employee*);
};
```

Модуль employee.cpp

```
#include "employee.h"

#include <iostream>

void employee::print()
{
	std::cout << "Employee: Name: " << this->Name << "; Age: " << this->Age <<
"; Experience: " << this->Experience << ";\n";
}

void employee::add(employee* e) {
	employee* el = this;
	while (el->next) {
```

```
			el = el->next;
		}
		el->next = e;
}
```

Модуль workstation.h

```
#pragma once
#include "department.h"
#include "employee.h"

class workstation :
    public department
{
protected:
		employee* employee_;
public:
		using department::department;
		workstation();
		workstation(std::string name, employee* employee);
		~workstation() = default;

		void print();
		void iterate(bool (f)(servant s, int), int a) override;
};
```

Модуль workstation.cpp

```
#include "workstation.h"

#include <iostream>

void workstation::iterate(bool (f)(servant e, int), int a) {
		employee* e = employee_;
		do {
			if (f(*e, a)) {
				e->print();
			} e = e->next;
		} while (e);
}

workstation::workstation(std::string n, employee* employee)
{
		name = n;
		employee_ = employee;
}

void workstation::print() {
		std::cout << "Workstation" << std::endl;
		}
```

Модуль research_station.h

```
#pragma once
#include "department.h"
#include "engineer.h"
```

```
class research_station :
    public department
{
protected:
	engineer* engineer_;
public:
	using department::department;
	research_station(std::string, engineer*);
	~research_station() = default;

	void print();
	void iterate(bool (f)(servant e, int), int a) override;
};
```

Модуль research_station.cpp

```
#include "research_station.h"

#include <iostream>

#include "engineer.h"

void research_station::iterate(bool (f)(servant e, int), int a) {
	engineer* e = engineer_;
	do {
		if (f(*e, a)) {
			e->print();
		} e = e->next;
	} while (e);
}

void research_station::print() {
	std::cout << "Workstation" << std::endl;
}

research_station::research_station(std::string name_, engineer* engineer)
{
	name = name_;
	engineer_ = engineer;
	}
```

Модуль department.h

```
#pragma once
#include "servant.h"
#include "TObject.h"
class department :
    public TObject
{
protected:
	std::string name;
public:
    department* next;
    department(std::string);
    department(department&);
    department();
    virtual ~department();

	std::string get_name();
```

```
    void set_name(char* NAME);

    void add(department* s);
    void show();
    void print();
    virtual void iterate(bool (f)(servant w, int), int a) = 0;
};
```

Модуль department.cpp

```
#include "department.h"

#include <iostream>

department::department(std::string name_)
{
	name = name_;
};
department::department(department& d) : department(d.name) {};
department::department() {};
department::~department(){};

std::string department::get_name()
{
	return name;
};
void department::set_name(char* NAME)
{
	name = NAME;
};
void department::add(department* d)
{
	department* el = this; while (el->next) {
		el = el->next;
	}
	el->next = d;
};

void department::print()
{
}

void department::iterate(bool f(servant s, int), int a)
{
	std::cout << "boo";
	}
```

Модуль company.h

```
#pragma once
#include "department.h"
class company :
    public department
{
public:
    using department::department;
    department* department_;

    company();
```

```
    company(std::string n, department* d);
    ~company() override;
    void print();
    void iterate(bool (f)(servant m, int), int a) override;
};
```

Модуль company.cpp

```
#include "company.h"
company::company(std::string n, department* d)  {
	name = n;
	department_ = d;
}
void company::print() {}
void company::iterate(bool (f)(servant m, int), int a) {
	department* u = department_;
	do {
		u->iterate(f, a);
		u = u->next;
	} while (u);
}
company::~company() = default;
```

Модуль main.cpp

```
#include <iostream>

#include "company.h"
#include "employee.h"
#include "engineer.h"
#include "research_station.h"
#include "servant.h"
#include "workstation.h"

using namespace std;

int main()
{
	employee empl1 ("rob", 25, 6);
	employee empl2 ("rob1", 26, 7);
	employee empl3 ("rob2", 56, 8);

	empl1.add(&empl2);
	empl2.add(&empl3);

	engineer engin1("chris", 30, 10);
	engineer engin2("chris1", 35, 15);
	engineer engin3("chris2", 300, 18);

	engin1.add(&engin2);
	engin2.add(&engin3);

	workstation workstation_("Workstation1", &empl1);
	research_station research_station_("Research1", &engin1);

	workstation_.add(&research_station_);
```

```
        company company_("company", &workstation_);
        company_.iterate(servant::check_is_older, 30);

}
```

Результат тестування роботи програми

```
Employee: Name: rob2; Age: 56; Experience: 8;
Engineer: Name: chris; Age: 30; Experience: 10;
Engineer: Name: chris1; Age: 35; Experience: 15;
Engineer: Name: chris2; Age: 300; Experience: 18;
```

Рисунок 1.1 – Результат тесутвання програми

Висновок: ми навчились створювати програму на мові програмування C++, працювати з класами, їх ієрархією, вказівниками та віртуальними функціями.