Міністерство освіти і науки України
Національний технічний університет України “Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 5 з дисципліни
«Основи програмування 2. Модульне програмування»

«Успадкування та поліморфізм»

Варіант 29

Виконав студент ІП-15 Рибалка Ілля Сергійович
(шифр, прізвище, ім'я, по батькові)

Перевірив Вєчерковська Анастасія Сергіївна
(прізвище, ім'я, по батькові)

Київ 2022

Лабораторна робота 4

Успадкування та поліморфізм

Мета - вивчити механізм створення і використання класів і об’єктів.

Індивідуальне завдання

1.1.1 Вимоги до програми

1) Реалізація коду відповідно до модульного підходу.
2) Виведення усіх вхідних, проміжних і вихідних даних.
3) У кожному варіанті завдання при описі класів самостійно визначити необхідні поля та методи вводу/виводу. Деякі методи класу-предка можуть бути віртуальними і абстрактними. У програмі-клієнті для збереження сукупності об’єктів використати масив.

Варіант 29

Розв’язання

Код

C++

main.cpp

```cpp
#include "func.h"



int main()

{

   int n, m;

   Student *stud = crstud(&n);

   Lecturer *lect = crlect(&m);

   print(stud, n);

   print(lect, m);

   calcsc(stud, n);
```

```
    calcpay(lect, m);

    delete []stud;

    delete []lect;

    return 0;

}
```

func.h

```
#pragma once

#include "Person.h"

#include "Student.h"

#include "Lecturer.h"



Student *crstud(int *); // Створення масиву студентів

Lecturer *crlect(int *); // Створення масиву викладачів

bool valstud(std::string); // Валідація рядку, що описує студента

bool vallec(std::string); // Валідація рядку, що описує викладача

bool dateval(std::string); // Перевірка коректності введеної дати

bool isnum(std::string, std::string); // Перевірка - чи є рядок числом

void print(Student *, int); // Виведення студентів

void print(Lecturer *, int); // Виведення икладачів

void calcsc(Student *, int); // Підрахунок стипендії студентів

void calcpay(Lecturer *, int); // Підрахунок заробітньої плати викладачів

std::ostream& operator<< (std::ostream &, Student); // Виведення атрибутів класу студент

std::ostream& operator<< (std::ostream &, Lecturer); // Виведення атрибутів класу викладач
```

func.cpp

```
#include "func.h"



Student *crstud(int *n)

{

   int num;

   std::string str;

   std::cout << "Введіть кількість студентів - ";

   std::cin >> num;

   while(num<=0)

   {

       std::cout << "Введіть дійсне число - ";

       std::cin >> num;

   }

   *n = num;

   Student *res = new Student[num];

   std::cout << "Введіть студентів за форматом - Ім'я;Дата народження(DD.MM.YYYY);Номер групи;Середній рейтинговий бал" << std::endl;

   std::cin.ignore();

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

   {

       std::getline(std::cin, str);

       if (valstud(str))

       {

           Student temp(str);

           res[i]= temp;

       }
```

```
        else i--;
    }
    return res;
}


bool valstud(std::string str)
{
    std::vector<std::string> words;
    int n1 = 0, n2 = str.find(';'), n=0;
    while(n2!=-1 && n<4)
    {
        words.push_back(str.substr(n1,n2-n1));
        n++;
        n1 = n2+1;
        n2 = str.find(';', n1+1);
    }
    words.push_back(str.substr(n1,str.size()-n1));
    if(n!=3)
    {
        std::cout << "Некоректно введені дані" << std::endl;
        return false;
    }
    if(!dateval(words[1]))
    {
        std::cout << "Некоректно введена дата" << std::endl;
```

```
        return false;
    }
    if(!isnum(words[2], "int"))
    {
        std::cout << "Некоректно вказаний номер групи" << std::endl;
        return false;
    }
    if(!isnum(words[3], "float"))
    {
        std::cout << "Некоректно введений середній бал" << std::endl;
        return false;
    }
    if(stoi(words[3])<0 || stoi(words[3])>100)
    {
        std::cout << "Некоректно введений середній бал" << std::endl;
        return false;
    }
    return true;
}


Lecturer *crlect(int *m)
{
    int num;
    std::string str;
    std::cout << "Введіть кількість викладачів - ";
```

```
    std::cin >> num;

    while(num<=0)

    {

        std::cout << "Введіть дійсне число - ";

        std::cin >> num;

    }

    *m = num;

    Lecturer *res = new Lecturer[num];

            std::cout    <<    "Введіть    викладачів    за    форматом    -    Ім'я;Дата
народження(DD.MM.YYYY);Назва дисципліни;Кількість годин" << std::endl;

    std::cin.ignore();

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

    {

        std::getline(std::cin, str);

        if (vallec(str))

        {

            Lecturer temp(str);

            res[i]= temp;

        }

        else i--;

    }

    return res;

}



bool vallec(std::string str)

{
```

```
std::vector<std::string> words;

int n1 = 0, n2 = str.find(';');

while(n2!=-1)

{

    words.push_back(str.substr(n1,n2-n1));

    n1 = n2+1;

    n2 = str.find(';', n1+1);

}

words.push_back(str.substr(n1,str.size()-n1));

if(words.size()%2!=0)

{

    std::cout << "Некоректно введені дані" << std::endl;

    return false;

}

if(!dateval(words[1]))

{

    std::cout << "Некоректно введена дата" << std::endl;

    return false;

}

for (int i = 3; i < words.size(); i+=2)

{

    if(!isnum(words[i], "int"))

    {

        std::cout << "Некоректно введена кількість годин" << std::endl;

        return false;
```

```
        }

    }

    return true;

}



bool dateval(std::string date)

{

    for (int i = 0; i < date.size(); i++) if (date[i]!='.' && !isdigit(date[i])) return
false;

    int t1=date.find("."), t2=date.find(".", t1+1);

    if (t1==std::string::npos) return false;

    int day, mon, year;

    day = stoi(date.substr(0,t1));

    mon = stoi(date.substr(t1+1,t2-t1));

    year = stoi(date.substr(t2+1,date.size()-t2));

    if(day < 1 || mon < 1 || mon > 12 || year <= 1900) return false;

     int val[13]={0,31,28,31,30,31,30,31,31,30,31,30,31}; // Кількість днів у кожному
місяці року

     if(mon == 2 && (year%400==0)||((year%4==0)&&(year%100!=0)) && day > 29) return
false;

    if(day > val[mon]) return false;

    return true;

}



bool isnum(std::string str, std::string type)

{

    for (int i = 0; i < str.size(); i++)
```

```
    {
        if(type=="int")
        {
            if(!std::isdigit(str[i]))
            {
                return false;
            }
        }
        else
        {
            if(!std::isdigit(str[i])&& str[i]!='.')
            {
                return false;
            }
        }
    }
    return true;
}


void print(Student *studs, int n)
{
    std::cout << "Список студентів:" << std::endl;
    for (int i = 0; i < n; i++)
    {
        std::cout << studs[i] << std::endl;
```

```
    }

}



void print(Lecturer *lects, int m)

{

    std::cout << "Список викладачів:" << std::endl;

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

    {

        std::cout << lects[i] << std::endl;

    }

}



void calcsc(Student *studs, int n)

{

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

    {

         std::cout << "Розмір місячної стипендії студента " << studs[i].GetName() << " -
" << studs[i].earn() << "(" << studs[i].GetStat() << ")" << std::endl;

    }

}



void calcpay(Lecturer *lects, int m)

{

    std::vector<float> nums;

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

    {
```

```
        nums.push_back(lects[i].earn());

        std::cout << "Розмір місячної заробіньої плати викладача " << lects[i].GetName()
<< " - " << nums[i] << std::endl;

    }

    int maxindx = distance(nums.begin(), max_element(nums.begin(), nums.end()));

     std::cout << "Найбільшу заробітню плату має викладач " << lects[maxindx].GetName()
<< "(" << lects[maxindx].years() << ") - " << nums[maxindx] << std::endl;

}



std::ostream& operator<< (std::ostream &out, Student stud)

{

    return out << stud.GetName() << ";" << stud.GetDate() << ";" << stud.GetGrnum() <<
";" << stud.GetAvrat();

}



std::ostream& operator<< (std::ostream &out, Lecturer lect)

{

    out << lect.GetName() << ";" << lect.GetDate();

    std::vector<disciplines> dis = lect.GetDis();

    for (int i = 0; i < dis.size(); i++)

    {

        out << ";";

        out << dis[i].disname << ";" << dis[i].hours;

    }

    return out;

}
```

Person.h

```h
#pragma once

#include <iostream>

#include <string>

#include <vector>

#include <ctime>



class Person

{

protected:

   std::string fname; // Ім'я

   std::string date; // Дата народження

   int years(std::string); // Знаходження віку за рядком

public:

   int years(); // Знаходження віку

   virtual float earn() = 0;  // Підрахунок місячної заробітньої плати

   std::string GetName(); // Повертає ім'я

   std::string GetDate(); // Повертає дату народження

};
```

Person.cpp

```cpp
#include "Person.h"



int Person::years()

{

   std::string date = this->date;

   int age, day, mon, year;
```

```
    int t1=date.find("."), t2=date.find(".", t1+1);

    struct tm inpdate;

    day = stoi(date.substr(0,t1));

    mon = stoi(date.substr(t1+1,t2-t1));

    year = stoi(date.substr(t2+1,date.size()-t2));

    inpdate.tm_mday = day;

    inpdate.tm_mon = mon-1;

    inpdate.tm_year = year-1900;

    inpdate.tm_hour = 0;

    inpdate.tm_min = 0;

    inpdate.tm_sec = 0;

    time_t dt1, dt2 = mktime(&inpdate);

    time(&dt1);

    age = difftime(dt1, dt2)/(60*60*24*365.2425);

    return age;

}



int Person::years(std::string date)

{

    int age, day, mon, year;

    int t1=date.find("."), t2=date.find(".", t1+1);

    struct tm inpdate;

    day = stoi(date.substr(0,t1));

    mon = stoi(date.substr(t1+1,t2-t1));

    year = stoi(date.substr(t2+1,date.size()-t2));
```

```
    inpdate.tm_mday = day;

    inpdate.tm_mon = mon-1;

    inpdate.tm_year = year-1900;

    inpdate.tm_hour = 0;

    inpdate.tm_min = 0;

    inpdate.tm_sec = 0;

    time_t dt1, dt2 = mktime(&inpdate);

    time(&dt1);

    age = difftime(dt1, dt2)/(60*60*24*365.2425);

    return age;

}



std::string Person::GetName()

{

    return this->fname;

}



std::string Person::GetDate()

{

    return this->date;

}
```

Student.h

```
#pragma once

#include "Person.h"
```

```
class Student : public Person

{

private:

   int grpnum; // Номер групи

   float avratsc; // Середній рейтинговий бал

   std::string scholstat; // Статус отримання стипендії

public:

   Student();

   Student(std::string, std::string, int, float);

   Student(std::string);

   float earn(); // Підрахунок стипендії

   int GetGrnum(); // Повертає номер групи

   float GetAvrat(); // Повертає середній рейтинговий бал

   std::string GetStat(); // Повертає статус отримання стипендії

};
```

Student.cpp

```
#include "Student.h"



Student::Student()

{

   this->fname = "Noname";

   this->date = "01.01.1901";

}



Student::Student(std::string fname, std::string date, int grpnum, float avratsc)
```

```
{

    this->fname = fname;

    this->date = date;

    this->grpnum = grpnum;

    this->avratsc = avratsc;

    if (avratsc > 95) this->scholstat = "Підвищена";

    else if(avratsc > 85) this->scholstat = "Звичайна";

    else this->scholstat = "Немає";

}



Student::Student(std::string str)

{

    std::string fname, date;

    int grpnum;

    float avratsc;

    std::vector<std::string> words;

    int n1 = 0, n2 = str.find(';');

    for(int i=0; i<4; i++)

    {

        words.push_back(str.substr(n1,n2-n1));

        n1 = n2+1;

        n2 = str.find(';', n1+1);

    }

    words.push_back(str.substr(n1,str.size()-n1));

    fname = words[0];
```

```
    date = words[1];

    grpnum = stoi(words[2]);

    avratsc = stoi(words[3]);

    this->fname = fname;

    this->date = date;

    this->grpnum = grpnum;

    this->avratsc = avratsc;

    if (avratsc > 95) this->scholstat = "Підвищена";

    else if(avratsc > 85) this->scholstat = "Звичайна";

    else this->scholstat = "Немає";

}



float Student::earn()

{

    std::string stat = this->scholstat;

    if (stat == "Підвищена") return 2000;

    else if(stat == "Звичайна") return 1000;

    else return 0;

}



int Student::GetGrnum()

{

    return this->grpnum;

}
```

```
float Student::GetAvrat()

{

   return this->avratsc;

}



std::string Student::GetStat()

{

   return this->scholstat;

}
```

Lecturer.h

```
#pragma once

#include "Person.h"



struct disciplines

{

   std::string disname; // Назва дисципліни

   int hours; // Кількість годин

};



class Lecturer : public Person

{

private:

   std::vector<disciplines> dis; // Дисципліни

public:

   Lecturer();
```

```
    Lecturer(std::string, std::string, std::vector<std::string>, std::vector<int>);

    Lecturer(std::string);

    float earn(); // Підрахунок місячної заробітньої плати

    std::vector<disciplines> GetDis(); // Повертає дисципліни

};
```

Lecturer.cpp

```
#include "Lecturer.h"



Lecturer::Lecturer()

{

    this->fname = "Noname";

    this->date = "01.01.1901";

}



Lecturer::Lecturer(std::string fname, std::string date, std::vector<std::string>
disname, std::vector<int> hours)

{

    this->fname = fname;

    this->date = date;

    disciplines dat;

    for (int i = 0; i < disname.size(); i++)

    {

        dat.disname = disname[i];

        dat.hours = hours[i];

        this->dis.push_back(dat);

    }
```

```
}



Lecturer::Lecturer(std::string str)

{

   std::string fname, date;

   std::vector<std::string> disname;

   std::vector<int> hours;

   int n1 = 0, n2 = str.find(';');

   fname = str.substr(n1,n2-n1);

   n1 = n2+1;

   n2 = str.find(';', n1+1);

   date = str.substr(n1,n2-n1);

   n1 = n2+1;

   n2 = str.find(';', n1+1);

   std::vector<std::string> words;

   while(n2!=-1)

   {

       words.push_back(str.substr(n1,n2-n1));

       n1 = n2+1;

       n2 = str.find(';', n1+1);

   }

   words.push_back(str.substr(n1,str.size()-n1));

   for(int i = 0; i<words.size(); i+=2)

   {

       disname.push_back(words[i]);
```

```
    }

    for(int i = 1; i<words.size(); i+=2)

    {

        hours.push_back(stoi(words[i]));

    }

    this->fname = fname;

    this->date = date;

    disciplines dat;

    for (int i = 0; i < disname.size(); i++)

    {

        dat.disname = disname[i];

        dat.hours = hours[i];

        this->dis.push_back(dat);

    }

}



float Lecturer::earn()

{

    int hours = 0;

    for (int i = 0; i < this->dis.size(); i++)

    {

        hours += this->dis[i].hours;

    }

    return 100*hours;

}
```

```
std::vector<disciplines> Lecturer::GetDis()

{

   return this->dis;

}
```

Python

main.py

```
from func import *



stud = crstud()

lect = crlect()

prints(stud)

printl(lect)

calcsc(stud)

calcpay(lect)
```

func.py

```
from curses.ascii import isdigit

from Student import *

from Lecturer import *



# Створення масиву студентів

def crstud():

   res = []

   num = int(input("Введіть кількість студентів - "))

   while num<=0:
```

```
        num = int(input("Введіть дійсне число - "))

    print("Введіть студентів за форматом - Ім'я;Дата народження(DD.MM.YYYY);Номер групи;Середній рейтинговий бал")

    while num!=0:

        str = input()

        if valstud(str):

            res.append(Student(str))

            num-=1

    return res



# Створення масиву викладачів

def crlect():

    res = []

    num = int(input("Введіть кількість викладачів - "))

    while num<=0:

        num = int(input("Введіть дійсне число - "))

    print("Введіть викладачів за форматом - Ім'я;Дата народження(DD.MM.YYYY);Назва дисципліни;Кількість годин")

    while num!=0:

        str = input()

        if vallec(str):

            res.append(Lecturer(str))

            num-=1

    return res



# Валідація рядку, що описує студента
```

```
def valstud(string):

    if ";" not in string or string.count(";")!=3:

        print("Некоректно введені дані")

        return False

    lst = string.split(";")

    if not dateval(lst[1]):

        print("Некоректно введена дата")

        return False

    if not isnum(lst[2]):

        print("Некоректно вказаний номер групи")

        return False

    if not isnum(lst[3], "float"):

        print("Некоректно введений середній бал")

        return False

    if int(lst[3])<0 or int(lst[3])>100:

        print("Некоректно введений середній бал")

        return False

    return True



# Валідація рядку, що описує викладача

def vallec(string):

    if ";" not in string:

        print("Некоректно введені дані")

        return False

    lst = string.split(";")
```

```
    if len(lst)%2 != 0:
        print("Некоректно введені дані")
        return False
    if not dateval(lst[1]):
        print("Некоректно введена дата")
        return False
    for i in range(3, len(lst), 2):
        if not isnum(lst[i]):
            print("Некоректно введена кількість годин")
            return False
    return True


# Перевірка коректності введеної дати
def dateval(string):
    for ch in string:
        if not isdigit(ch) and ch!='.': return False
    if '.' not in string or string.count(".")>2 or string.count(".")<2: return False
    date = string.split(".")
    day = int(date[0])
    mon = int(date[1])
    year = int(date[2])
    if day < 1 or mon < 1 or mon > 12 or year <= 1900: return False
    val = [0,31,28,31,30,31,30,31,31,30,31,30,31] # Кількість днів у кожному місяці року
     if mon == 2 and (year%400==0)or((year%4==0)or(year%100!=0)) and day > 29:return
False
    if day > val[mon]: return False
```

```
    return True



# Перевірка - чи є рядок числом

def isnum(string, type = "int"):

    for ch in string:

        if type == "int":

            if not isdigit(ch): return False

        else:

            if not isdigit(ch) and ch != ".": return False

    return True



# Виведення студентів

def prints(studs):

    print("Список студентів:")

    for stud in studs:

        print(stud)



# Виведення икладачів

def printl(lects):

    print("Список викладачів:")

    for lect in lects:

        print(lect)



#  Підрахунок стипендії студентів

def calcsc(studs):
```

```
    for stud in studs:

                    print(f"Розмір  місячної  стипендії  студента  {stud.GetName()}  -
{stud.earn()}({stud.GetStat()})")



# Підрахунок заробітньої плати викладачів

def calcpay(lects):

    nums = []

    for lect in lects:

        nums.append(lect.earn())

              print(f"Розмір  місячної  заробіньої  плати  викладача  {lect.GetName()}  -
{nums[len(nums)-1]}")

    maxindx = nums.index(max(nums))

        print(f"Найбільшу  заробітню  плату  має  викладач  {lects[maxindx].GetName()}
({lects[maxindx].years()}) - {nums[maxindx]}")
```

Person.py

```
from datetime import datetime

from abc import ABC, abstractmethod

class Person(ABC):

    def __init__(self, name, date):

        self._name = name # Ім'я

        self._date = date # Дата народження



    # Підрахунок віку

    def years(self):

        curdate = datetime.now()

        inpdate = datetime.strptime(self._date, "%d.%m.%Y")

        res = (curdate-inpdate).days
```

```
        return int(res/365.2425)



    @abstractmethod

    def earn(self):

        pass



    # Повертає ім'я

    def GetName(self):

        return self._name



    # Повертає дату народження

    def GetDate(self):

        return self._date
```

Student.py

```
from Person import *

class Student(Person):

    def __init__(self, string):

        lst = string.split(";")

        super().__init__(lst[0], lst[1])

        self.__grpnum = int(lst[2]) # Номер групи

        self.__avratsc = int(lst[3]) # Середній рейтинговий бал

        if self.__avratsc > 95: self.__scholstat = "Підвищена"

        elif self.__avratsc > 85: self.__scholstat = "Звичайна"

        else: self.__scholstat = "Немає"
```

```
    # Підрахунок стипендії

    def earn(self):

        stat = self.__scholstat

        if stat == "Підвищена": return 2000

        elif stat == "Звичайна": return 1000

        else: return 0



    # Повертає номер групи

    def GetGrnum(self):

        return self.__grpnum



    # Повертає Середній рейтинговий бал

    def GetAvrat(self):

        return self.__avratsc



    # Повертає статус отримання стипендії

    def GetStat(self):

        return self.__scholstat



    # Перевантаження виведення

    def __str__(self):

        return f"{self._name};{self._date};{self.__grpnum};{self.__avratsc}"
```

Lecturer.py

```
from Person import *

class Lecturer(Person):
```

```
def __init__(self, string):

    lst = string.split(";")

    super().__init__(lst[0], lst[1])

    self.__dis = {"disname": [], "hours": []}

    for i in range(2, len(lst), 2):

        d, h = lst[i:i + 2][0], lst[i:i + 2][1]

        self.__dis["disname"].append(d)

        self.__dis["hours"].append(int(h))



# Підрахунок місячної заробітньої плати

def earn(self):

    hours = 0

    for hour in self.__dis["hours"]:

        hours += hour

    return 100*hours



# Повертає список дисциплін

def GetDis(self):

    return self.__dis



# Перевантаження виведення

def __str__(self):

    res = f"{self._name};{self._date}"

    dis = self.__dis

    for i in range(len(dis["hours"])):
```

```
            res += ";{};{}".format(dis["disname"][i],dis["hours"][i])


        return res
```

Тестування

C++

```
Введіть кількість студентів – 0
Введіть дійсне число – 1
Введіть студентів за форматом – Ім'я;Дата народження(DD.MM.YYYY);Номер групи;Середній рейтинговий бал
asd;12.12.2000;15;101
Некоректно введений середній бал
qwe;12.12.2000;15
Некоректно введені дані
qwe;12.12.2000;15;100
Введіть кількість викладачів – 1
Введіть викладачів за форматом – Ім'я;Дата народження(DD.MM.YYYY);Назва дисципліни;Кількість годин
qwe;12.12.2001;qwe;10
Список студентів:
qwe;12.12.2000;15;100
Список викладачів:
qwe;12.12.2001;qwe;10
Розмір місячної стипендії студента qwe – 2000(Підвищена)
Розмір місячної заробіньої плати викладача qwe – 1000
Найбільшу заробітню плату має викладач qwe(20) – 1000
```

Python

```
Введіть кількість студентів – 3
Введіть студентів за форматом – Ім'я;Дата народження(DD.MM.YYYY);Номер групи;Середній рейтинговий бал
qwe;qwe;qwe;qwe
Некоректно введена дата
qwe;12.12.2000;15;100
asd;12.12.2000;15;90
zxc;12.12.2000;15;80
Введіть кількість викладачів – 2
Введіть викладачів за форматом – Ім'я;Дата народження(DD.MM.YYYY);Назва дисципліни;Кількість годин
qwe;12.12.1990;qwe;10;asd;20;zxc;30
zxc;12.12.1980;fgh;5
Список студентів:
qwe;12.12.2000;15;100
asd;12.12.2000;15;90
zxc;12.12.2000;15;80
Список викладачів:
qwe;12.12.1990;qwe;10;asd;20;zxc;30
zxc;12.12.1980;fgh;5
Розмір місячної стипендії студента qwe – 2000(Підвищена)
Розмір місячної стипендії студента asd – 1000(Звичайна)
Розмір місячної стипендії студента zxc – 0(Немає)
Розмір місячної заробіньої плати викладача qwe – 6000
Розмір місячної заробіньої плати викладача zxc – 500
Найбільшу заробітню плату має викладач qwe (31) – 6000
```

Висновок

Я дослідив механізм створення використання класів і об’єктів. В ході роботи над лабораторною роботою було створено програму для обробки 2 масивів - студентів та викладачів.