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»

Варіант 25

Виконав студент ІП-14 Радзівіло Валерія Артемівна

(шифр, прізвище, ім'я, по батькові)

Перевірив

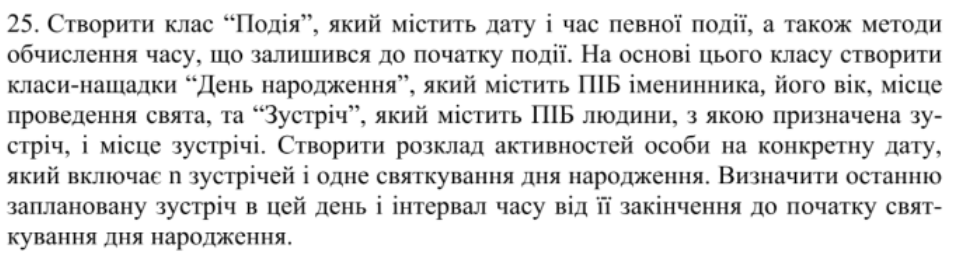
( прізвище, ім'я, по батькові)

Київ 2022

**Лабораторна робота №5**

**Тема роботи:** Успадкування та поліморфізм

**Мета роботи:** Вивчити механізми створення і використання класів та об’єктів.



**Вирішення задачі С++:**

**main:**

#include "header.cpp"  
  
int main()  
{  
 int a = 10;  
 cout << "On which day do you want to make a schedule? ";  
 string date;  
 getline(cin, date);  
 Birthday birth;  
 // for code to be reusable  
 int total\_amount= 0;  
 // all times that we have meetings on  
 string all\_time\_meets[100];  
 // list of all meetings  
 Meeting all\_meets[100];  
 int zap = 0;  
 int start = 0;  
  
 do {  
 zap++;  
 cin.ignore();  
 cout << "How much meetings do you have? ";  
 int iterator;  
 cin >> iterator;  
 start+=total\_amount-1;  
 if(zap == 1)  
 {  
 // add birthday to both meets and times  
 all\_meets[0].MakeMeeting(birth.name, birth.place,birth.time);  
 all\_time\_meets[0] = birth.time;  
 start = 1;  
 }  
 cin.ignore();  
  
 total\_amount+=iterator;  
 //create schedule  
 make\_schedule(start, iterator,all\_time\_meets,all\_meets, total\_amount);  
 if(zap==1)  
 {  
 iterator++;  
 total\_amount++;  
 }  
  
 // sort schedule by time  
 sort\_schedule(all\_meets, total\_amount);  
 // schedule output  
 schedule\_output(all\_meets, total\_amount);  
 // last meet  
 cout << endl << "The last meeting:" << endl;  
 last\_meet(all\_meets, total\_amount, birth);  
 cout << "Do you want to extend your schedule? (print '0' if no)"<< endl;  
 cin >> a;  
 } while (a!=0);  
  
  
  
}

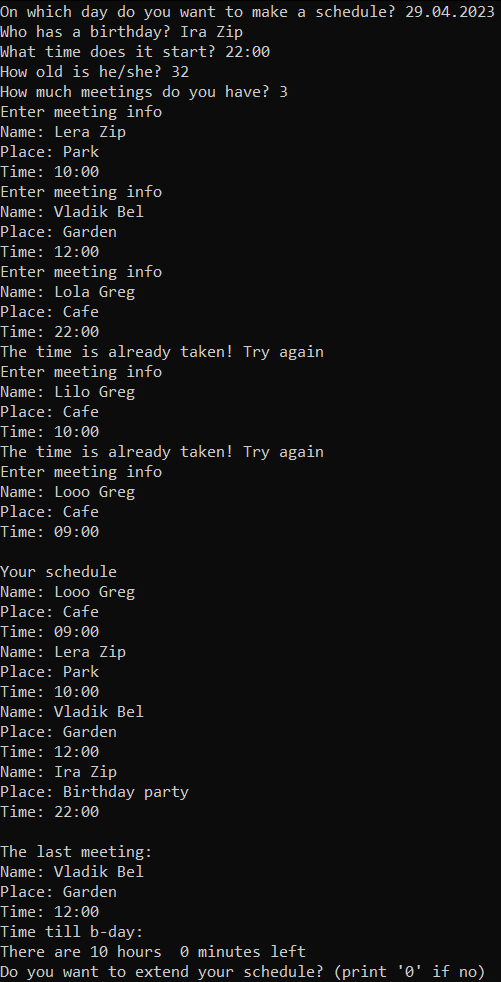
**header.h:**

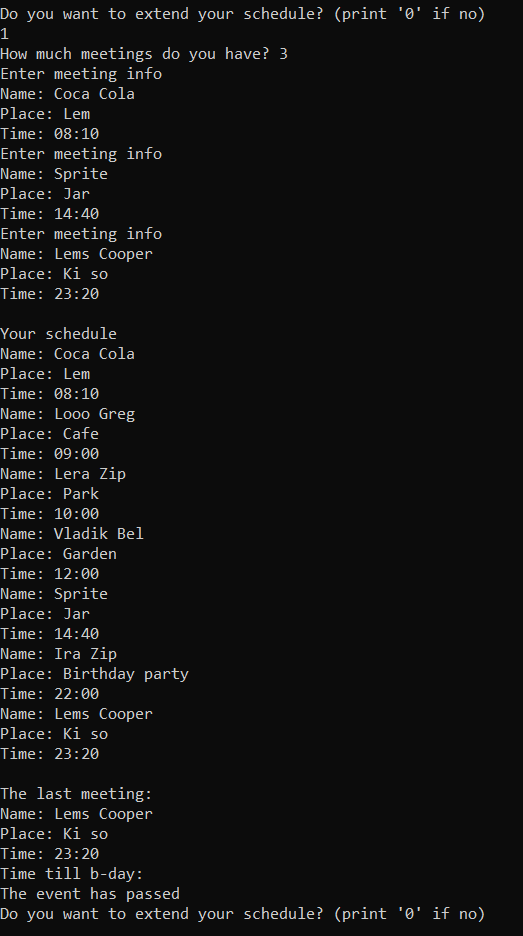
#ifndef CPP\_HEADER\_H  
#define CPP\_HEADER\_H  
#include <iostream>  
using namespace std;  
class Event  
{  
public:  
 string time;  
 string date;  
public:  
 void time\_till(string time\_start, string time\_till);  
};  
  
class Birthday: public Event{  
public:  
 string name;  
 int age;  
 string place = "Birthday party";  
public:  
 Birthday();  
 void printBirthday();  
};  
  
class Meeting: public Event{  
private:  
 string name;  
 string place;  
public:  
 void MakeMeeting(string name, string place, string time);  
 void SetMeeting();  
 void printMeeting();  
  
};  
int find\_time(string\* all\_meets, Meeting meet, int iterator);  
int string\_to\_hours(string time);  
int string\_to\_minutes(string time);  
void sort\_schedule(Meeting \*schedule, int amount);  
void schedule\_output(Meeting \* all\_meets, int iterator);  
void make\_schedule(int start\_point, int iterator, string \*all\_time\_meets, Meeting \*all\_meets, int total);  
  
#endif //CPP\_HEADER\_H

**header.cpp:**

#include "header.h"  
  
Birthday::Birthday() {  
 cout << "Who has a birthday? ";  
 getline(cin, name);  
 cout <<"What time does it start? ";  
 getline(cin, time);  
 cout << "How old is he/she? ";  
 cin >> age;  
  
}  
void Meeting::SetMeeting() {  
 cout << "Name: ";  
 getline(cin, name);  
 cout << "Place: ";  
 getline(cin, place);  
 cout << "Time: ";  
 getline(cin, time);  
}  
  
  
void Meeting::printMeeting() {  
 cout << "Name: "<< name<< endl;  
 cout << "Place: "<< place <<endl;  
 cout << "Time: "<< time << endl;  
}  
  
void Birthday::printBirthday() {  
 cout << "Name: "<< name<< endl;  
 cout << "Place: "<< place <<endl;  
 cout << "Time: "<< time << endl;  
}  
  
int find\_time(string\* all\_meets, Meeting meet, int total)  
{  
 for(int i=0;i<total;i++)  
 {  
 if(all\_meets[i]==meet.time)  
 {  
 return 1;  
 }  
 }  
}  
  
void Meeting::MakeMeeting(string name\_meet, string place\_meet, string time\_meet) {  
 this->name = name\_meet;  
 this->place = place\_meet;  
 this->time = time\_meet;  
}  
  
void Event::time\_till(string time\_start, string time\_till) {  
 int hours = string\_to\_hours(time\_start) - string\_to\_hours(time\_till);  
 int minutes = string\_to\_minutes(time\_start) - string\_to\_minutes(time\_till);  
 if(hours > 0 && minutes<0)  
 {  
 hours --;  
 minutes += 60;  
 }  
 if (hours >=0 && minutes>=0){  
 cout << "There are "<< hours <<" hours "<< minutes<< " minutes left"<< endl;  
 }  
 if(hours<0 && minutes < 0)  
 {  
 cout << "The event has passed "<< endl;  
 }  
  
}  
  
int string\_to\_hours(string time)  
{  
 int result;  
 if (stoi(time.substr(0,1)) == 0)  
 {  
 result = stoi(time.substr(1,2));  
 }  
 else {  
 result = stoi(time.substr(0, 2));  
 }  
 return result;  
}  
int string\_to\_minutes(string time)  
{  
 int result;  
 if (stoi(time.substr(3,1)) == 0)  
 {  
 result = stoi(time.substr(3,1));  
 }  
 else {  
 result = stoi(time.substr(3, 2));  
 }  
 return result;  
}  
void sort\_schedule(Meeting \*schedule, int amount)  
{  
 Meeting tmp;  
 for(int j =0; j < amount;j++) {  
 for (int i = 0; i < amount-1; i++) {  
 if (string\_to\_hours(schedule[i].time) > string\_to\_hours(schedule[i + 1].time)) {  
 tmp = schedule[i];  
 schedule[i] = schedule[i + 1];  
 schedule[i + 1] = tmp;  
 }  
 if (string\_to\_hours(schedule[i].time) == string\_to\_hours(schedule[i + 1].time)) {  
 if (string\_to\_minutes(schedule[i].time) > string\_to\_minutes(schedule[i + 1].time)) {  
 tmp = schedule[i];  
 schedule[i] = schedule[i + 1];  
 schedule[i + 1] = tmp;  
 }  
 }  
 }  
 }  
}  
  
void schedule\_output(Meeting \* all\_meets, int iterator)  
{  
 cout << endl << "Your schedule "<< endl;  
 for (int i = 0; i < iterator; i++) {  
 all\_meets[i].printMeeting();  
 }  
}  
  
void make\_schedule(int start\_point, int iterator, string \*all\_time\_meets, Meeting \*all\_meets, int total){  
 for (int i = 0; i < iterator; i++) {  
 cout << "Enter meeting info " << endl;  
 Meeting meet;  
 meet.SetMeeting();  
 if (find\_time(all\_time\_meets, meet, total) != 1) {  
 all\_meets[start\_point] = meet;  
 all\_time\_meets[start\_point] = meet.time;  
 start\_point++;  
 } else {  
 cout << "The time is already taken! Try again" << endl;  
 i--;  
  
 }  
 }  
}  
  
void last\_meet(Meeting \* all\_meets, int total\_amount, Birthday birth) {  
 if (string\_to\_hours(birth.time) != string\_to\_hours(all\_meets[total\_amount - 1].time) &&  
 string\_to\_minutes(birth.time) !=  
 string\_to\_minutes(all\_meets[total\_amount - 1].time)) {  
 all\_meets[total\_amount - 1].printMeeting();  
 cout << "Time till b-day: " << endl;  
 all\_meets[total\_amount - 1].time\_till(birth.time, all\_meets[total\_amount - 1].time);  
  
 }  
 else{  
 all\_meets[total\_amount - 2].printMeeting();  
 cout << "Time till b-day: " << endl;  
 all\_meets[total\_amount - 2].time\_till(birth.time, all\_meets[total\_amount - 2].time);  
 }  
  
}

**Результат:**





**Python:**

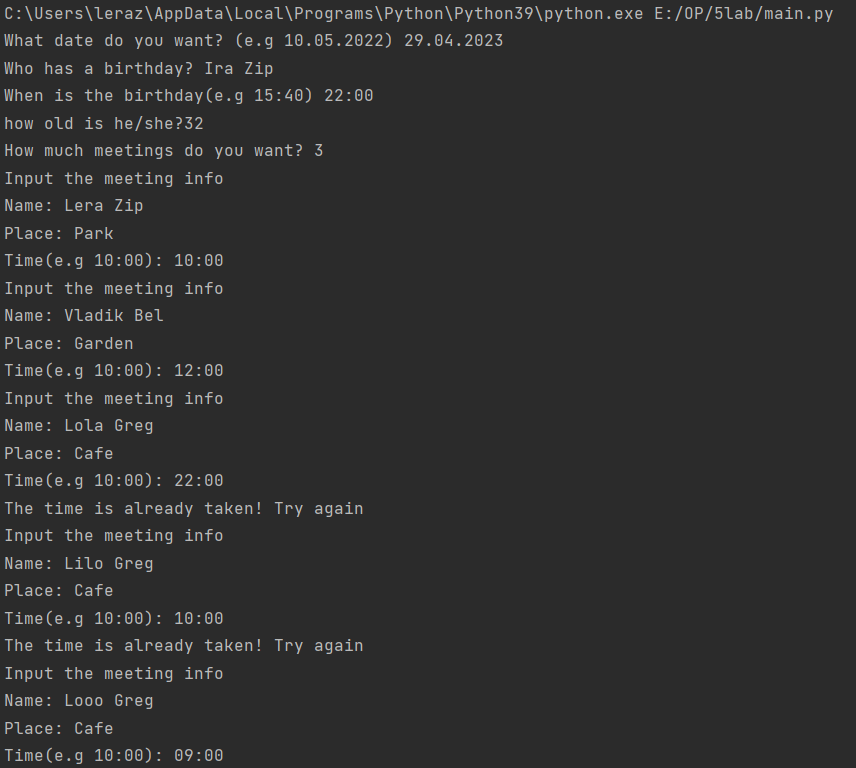
**main.py**

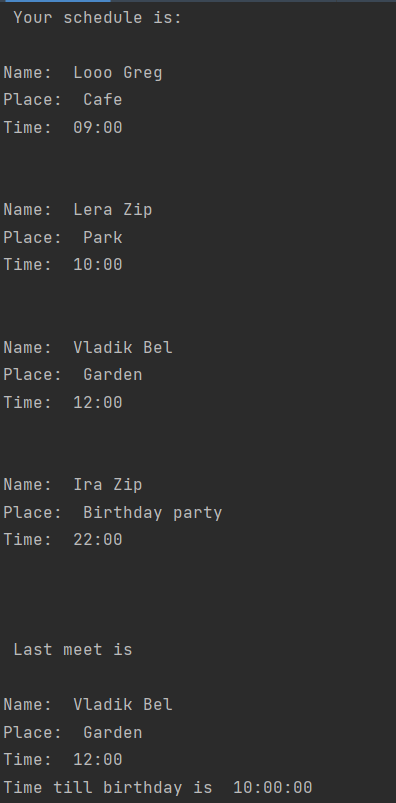
from datetime import datetime  
  
import funcs  
  
a = 'y'  
schedule = []  
taken\_times = []  
user\_date\_inp = input("What date do you want? (e.g 10.05.2022) ")  
birthday = funcs.Birthday()  
b\_time = birthday.start\_date  
numb\_meets = 0  
taken\_times.append(b\_time)  
# add birthday event to the schedule  
schedule.append(birthday)  
while a != 'n':  
 # the amount of times program has to ask info about Meetings  
 iterations = 0  
 iterations += int(input("How much meetings do you want? "))  
 # amount of all Meetings  
 numb\_meets += iterations  
 # create schedule  
 funcs.get\_schedule(iterations, taken\_times, schedule, birthday.us\_date)  
 # sort the schedule to be more user-friendly  
 schedule.sort(key=lambda x: x.start\_date)  
 # output user's schedule sorted by time  
 funcs.output\_schedule(numb\_meets, schedule)  
 # last meeting  
 funcs.last\_meet\_app(schedule, b\_time)  
 if numb\_meets == 0:  
 numb\_meets += 1  
 # the schedule can be extended  
 a = input("Do you want to append your schedule? (print 'n' if no) ")

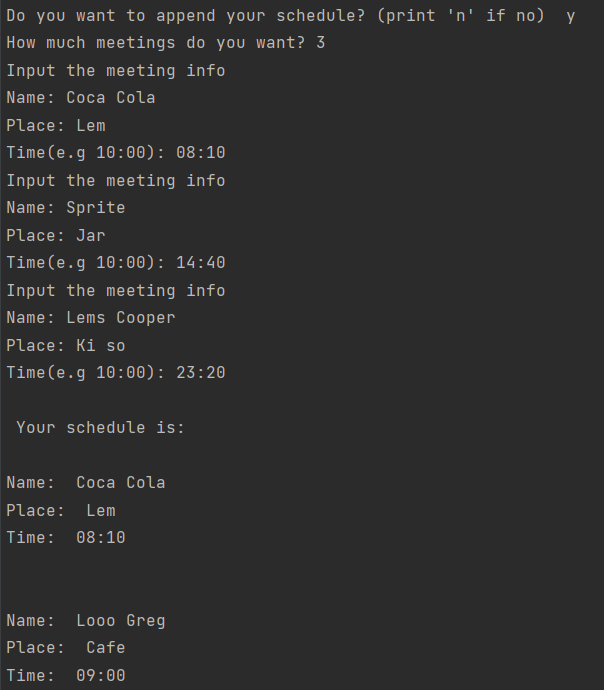
**funcs.py**

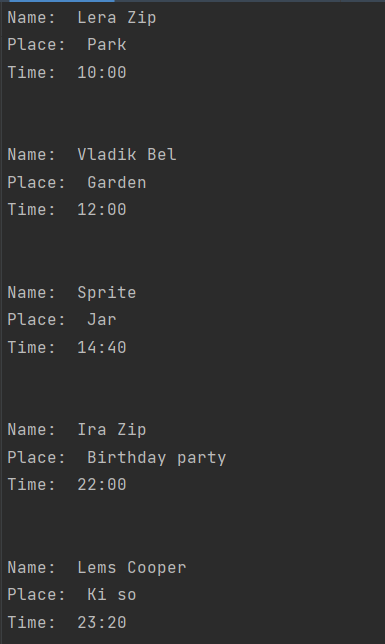
from datetime import datetime  
  
  
class Event:  
 def \_\_init\_\_(self, start\_date='None', user\_date='None'):  
 self.date\_ev = start\_date  
 self.us\_date = user\_date  
  
 # to count how much time left till the event  
 def count\_time\_left(self, us\_date, date\_ev):  
 dt\_user\_date = datetime.strptime(us\_date, '%H:%M')  
 dt\_event = datetime.strptime(date\_ev, '%H:%M')  
 day\_left = dt\_event - dt\_user\_date  
 if day\_left.days < 0:  
 return 0  
 else:  
 return day\_left  
  
  
class Birthday(Event):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.name = input("Who has a birthday? ")  
 self.start\_date = input("When is the birthday(e.g 15:40) ")  
 self.pers\_age = int(input("how old is he/she?"))  
 self.meet\_place = "Birthday party"  
  
 def print\_meet(self):  
 print("Name: ", self.name)  
 print("Place: ", self.meet\_place)  
 print("Time: ", self.start\_date)  
  
  
class Meeting(Event):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.name = input("Name: ")  
 self.meet\_place = input("Place: ")  
 self.start\_date = input("Time(e.g 10:00): ")  
  
 def print\_meet(self):  
 print("Name: ", self.name)  
 print("Place: ", self.meet\_place)  
 print("Time: ", self.start\_date)  
  
  
def get\_schedule(numb\_meets, taken\_times, schedule, bday):  
 i = 0  
 while i < numb\_meets:  
 print("Input the meeting info")  
 meeting = Meeting()  
 if meeting.start\_date not in taken\_times:  
 meeting.us\_date = bday  
 taken\_times.append(meeting.start\_date)  
 schedule.append(meeting)  
 i += 1  
 else:  
 print("The time is already taken! Try again")  
  
  
def output\_schedule(numb\_meets, schedule):  
 print("\n Your schedule is:\n")  
 for i in range(numb\_meets + 1):  
 schedule[i].print\_meet()  
 print('\n')  
  
  
def last\_meet\_app(schedule, b\_time):  
 numb\_meets = len(schedule)  
 last\_meet = schedule[numb\_meets - 1]  
 time\_till\_b = last\_meet.count\_time\_left(last\_meet.start\_date, b\_time)  
 if time\_till\_b == 0:  
 print("\n Last meet is \n")  
 last\_meet.print\_meet()  
 print("The event is after")  
 else:  
 last\_meet = schedule[numb\_meets - 2]  
 print("\n Last meet is \n")  
 last\_meet.print\_meet()  
 time\_till\_b = last\_meet.count\_time\_left(last\_meet.start\_date, b\_time)  
 if time\_till\_b.total\_seconds != 0:  
 print("Time till birthday is ", time\_till\_b)

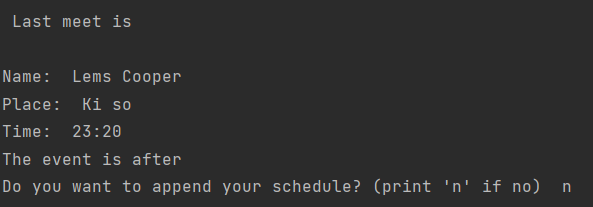
**Результат:**

****

****

****

****

****

**Висновок**: У цій лабораторній роботі було вивчено механізми створення і використання класів та об’єктів, успадкування та поліморфізм. Була постановлена задача, в якій визначалися n зустрічей одного класу, введені користувачем та 1 день народження, яке також введено користувачем. У алгоритмі були уточнені обмеження з використанням умов при яких задача буде виконана правильно. За допомоги цієї лабораторної роботи можливо складання та сортування розкладу, рахування часу від однієї події до іншої, визначення останньої події до дня народження.