Навчально-науковий комплекс “Інститут прикладного системного аналізу”

при Національному технічному університеті України “КПІ”

Кафедра математичних методів системного аналізу

Роботи комп’ютерного практикуму

з курсу “Об’єктно-орієнтоване програмування”

Виконав: студент 2-го курсу

групи КА-04

Штріккер Д.Я.

Прийняв: Шолохов О.В.

**Мета роботи:** Навчитись правильно описувати абстракції предметної області   
з використанням принципу інкапсуляції мовою С++.

**Завдання (варіант 8):**

Тип «Учасник змагань» визначити як клас, що містить:

* - закриті поля з ім’ям та прізвищем, реалізовані у вигляді символьних рядків довільної довжини, з датою народження типу Date;
* усі функції реалізувати відповідно загальним вимогам.

Тип «Виступ» визначити як клас, що містить:

* закрите поле логічного типу з типом змагань(командні, індивідуальні);
* статичне поле цілого типу з кількістю проведених виступів;
* закриті поля типу « Учасник змагань», цілі з порядковим номером поточного виступу (формується автоматично) та значенням результату;
* функції реалізувати відповідно загальним вимогам.

Тип «Змагання» визначити як клас, що містить:

* закрите поле з назвою змагання, реалізоване у вигляді символьного   
   рядка довільної довжини;
* інформацію про проведені виступи оформити як динамічний масив   
   (вказівник і розмірність, поля закриті);
* передбачити функцію для додавання чергового виступу до вже проведених;
* крім стандартної функції виведення повної інформації, передбачити функцію з виведенням скороченої інформації з назвою змагань та прізвищем переможця;
* інші функції реалізувати відповідно загальним вимогам.

**Лістинг програми:**

Github (разом із історією створення): <https://github.com/dindin28/task_1>

**Класс “Date”**

**Header:**

#ifndef \_TASK\_1\_INCLUDE\_TASK\_1\_DATE\_H\_

#define \_TASK\_1\_INCLUDE\_TASK\_1\_DATE\_H\_

#include <iostream>

class Date

{

public:

Date();

Date(int day, int month, int year);

Date(const Date &copy);

~Date();

//Getters

int GetDay();

int GetMonth();

int GetYear();

//Selectors

Date &SetDay(int day);

Date &SetMonth(int month);

Date &SetYear(int year);

void Print();

friend std::ostream& operator<< (std::ostream &out, const Date &obj);

friend std::istream& operator>> (std::istream &in, Date &obj);

private:

int day\_, month\_, year\_;

}; //Class (Date)

#endif //Header Guard

**Source file:**

// This is a personal academic project. Dear PVS-Studio, please check it.

// PVS-Studio Static Code Analyzer for C, C++ and C#: http://www.viva64.com

#include <task\_1/date.h>

#include <ctime>

#include <iomanip>

#include <cstdlib>

#include <iostream>

Date::Date()

{

std::time\_t now = std::time(nullptr);

char buff[4];

strftime(buff, sizeof(buff) + 1, "%d", std::localtime(&now));

day\_ = atoi(buff);

strftime(buff, sizeof(buff) + 1, "%m", std::localtime(&now));

month\_ = atoi(buff);

strftime(buff, sizeof(buff) + 1, "%Y", std::localtime(&now));

year\_ = atoi(buff);

}

Date::Date(int day, int month, int year)

: day\_(day), month\_(month), year\_(year) {}

Date::Date(const Date &copy)

: day\_(copy.day\_), month\_(copy.month\_), year\_(copy.year\_) {}

Date::~Date() {}

//Getters

int Date::GetDay() { return day\_; }

int Date::GetMonth() { return month\_; }

int Date::GetYear() { return year\_; }

//Selectors

Date &Date::SetDay(int day)

{

day\_ = day;

return \*this;

}

Date &Date::SetMonth(int month)

{

month\_ = month;

return \*this;

}

Date &Date::SetYear(int year)

{

year\_ = year;

return \*this;

}

void Date::Print()

{

std::cout << std::setfill('0');

std::cout << std::setw(2) << day\_ << "/"

<< std::setw(2) << month\_ << "/"

<< std::setw(4) << year\_ << std::endl;

}

std::ostream& operator<< (std::ostream &out, const Date &obj)

{

out << std::setfill('0');

out << std::setw(2) << obj.day\_ << "/"

<< std::setw(2) << obj.month\_ << "/"

<< std::setw(4) << obj.year\_;

return out;

}

std::istream& operator>> (std::istream &in, Date &obj)

{

std::cout << "Enter new day of date: ";

in >> obj.day\_;

std::cout << "Enter new month of date: ";

in >> obj.month\_;

std::cout << "Enter new year of date: ";

in >> obj.year\_;

return in;

}

**Клас “Participant”**

**Header:**

#ifndef \_TASK\_1\_INCLUDE\_TASK\_1\_PARTICIPANT\_H\_

#define \_TASK\_1\_INCLUDE\_TASK\_1\_PARTICIPANT\_H\_

#include <iostream>

#include <task\_1/date.h>

class Participant

{

public:

Participant();

Participant(const char \*name, const char \*surname, const Date &date);

Participant(const Participant &copy);

Participant &operator=(const Participant &copy);

~Participant();

//Getters

const char \*GetName();

const char \*GetSurname();

Date GetDate();

//Selectors

Participant &SetName(const char \*name);

Participant &SetSurname(const char \*surname);

Participant &SetDate(const Date &date);

void Print();

friend std::ostream &operator<<(std::ostream &out, const Participant &obj);

friend std::istream &operator>>(std::istream &in, Participant &obj);

private:

char \*name\_, \*surname\_;

Date date\_;

}; //Class (Participant)

#endif //Header Guard

**Source file:**

// This is a personal academic project. Dear PVS-Studio, please check it.

// PVS-Studio Static Code Analyzer for C, C++ and C#: http://www.viva64.com

#include <task\_1/participant.h>

#include <cstring>

#include <string>

Participant::Participant()

: date\_(15, 8, 2002)

{

const char \*name = "Dima";

const char \*surname = "Shtrikker";

name\_ = new char[strlen(name) + 1];

surname\_ = new char[strlen(surname) + 1];

strcpy(name\_, name);

strcpy(surname\_, surname);

}

Participant::Participant(const char \*name, const char \*surname, const Date &date)

: date\_(date)

{

name\_ = new char[strlen(name) + 1];

surname\_ = new char[strlen(surname) + 1];

strcpy(name\_, name);

strcpy(surname\_, surname);

}

Participant::Participant(const Participant &copy)

: date\_(copy.date\_)

{

name\_ = new char[strlen(copy.name\_) + 1];

surname\_ = new char[strlen(copy.surname\_) + 1];

strcpy(name\_, copy.name\_);

strcpy(surname\_, copy.surname\_);

}

Participant &Participant::operator=(const Participant &copy)

{

if (this != &copy)

{

//Name

if (strlen(name\_) != 0)

{

delete[] name\_;

}

name\_ = new char[strlen(copy.name\_) + 1];

strcpy(name\_, copy.name\_);

//Surname

if (strlen(surname\_) != 0)

{

delete[] surname\_;

}

surname\_ = new char[strlen(copy.surname\_) + 1];

strcpy(surname\_, copy.surname\_);

//Date

date\_ = copy.date\_;

} //Condition(this != &copy)

return \*this;

}

Participant::~Participant()

{

if (strlen(name\_) != 0)

{

delete[] name\_;

}

if (strlen(surname\_) != 0)

{

delete[] surname\_;

}

}

//Getters

const char \*Participant::GetName() { return name\_; }

const char \*Participant::GetSurname() { return surname\_; }

Date Participant::GetDate() { return date\_; }

//Selectors

Participant &Participant::SetName(const char \*name)

{

if (strlen(name\_) != 0)

{

delete[] name\_;

}

name\_ = new char[strlen(name) + 1];

strcpy(name\_, name);

return \*this;

}

Participant &Participant::SetSurname(const char \*surname)

{

if (strlen(surname\_) != 0)

{

delete[] surname\_;

}

surname\_ = new char[strlen(surname) + 1];

strcpy(surname\_, surname);

return \*this;

}

Participant &Participant::SetDate(const Date &date)

{

date\_ = date;

return \*this;

}

void Participant::Print()

{

std::cout << name\_ << " "

<< surname\_ << " ("

<< date\_ << ")" << std::endl;

}

std::ostream &operator<<(std::ostream &out, const Participant &obj)

{

out << obj.name\_ << " "

<< obj.surname\_ << " ("

<< obj.date\_ << ")";

return out;

}

std::istream &operator>>(std::istream &in, Participant &obj)

{

std::string buff\_string;

//Name

std::cout << "Enter new name of participant: ";

in >> buff\_string;

if (strlen(obj.name\_) != 0)

{

delete[] obj.name\_;

}

obj.name\_ = new char[buff\_string.length() + 1];

strcpy(obj.name\_, buff\_string.c\_str());

//Surname

std::cout << "Enter new surname of participant: ";

in >> buff\_string;

if (strlen(obj.surname\_) != 0)

{

delete[] obj.surname\_;

}

obj.surname\_ = new char[buff\_string.length() + 1];

strcpy(obj.surname\_, buff\_string.c\_str());

//Date

in >> obj.date\_;

return in;

}

**Клас “Performance”**

**Header:**

#ifndef \_TASK\_1\_INCLUDE\_TASK\_1\_PERFORMANCE\_H\_

#define \_TASK\_1\_INCLUDE\_TASK\_1\_PERFORMANCE\_H\_

#include <task\_1/participant.h>

class Performance

{

public:

Performance();

Performance(bool performance\_type, Participant participant, int sequence\_number, int result);

Performance(const Performance &copy);

Performance &operator=(const Performance &copy);

~Performance();

//Getters

bool GetPerformanceType();

int GetNumberOfCompetitions();

Participant GetParticipant();

int GetSequenceNumber();

int GetResult();

//Selectors

Performance &SetPerformanceType(bool performance\_type);

Performance &SetParticipant(Participant participant);

Performance &SetSequenceNumber(int sequence\_number);

Performance &SetResult(int result);

void Print();

friend std::ostream& operator<< (std::ostream &out, const Performance &obj);

friend std::istream& operator>> (std::istream &in, Performance &obj);

private:

// performance\_type\_= 0(false) => team performance

// performance\_type\_= 1(true) => individual performance

bool performance\_type\_;

static int number\_of\_performance\_;

Participant participant\_;

int sequence\_number\_, result\_;

}; //Class (Performance)

#endif //Header Guard

**Source file:**

// This is a personal academic project. Dear PVS-Studio, please check it.

// PVS-Studio Static Code Analyzer for C, C++ and C#: http://www.viva64.com

#include <task\_1/performance.h>

//Initialization of static variables

int Performance::number\_of\_performance\_ = 0;

// performance\_type\_= 0(false) => team performance

// performance\_type\_= 1(true) => individual performance

Performance::Performance()

: performance\_type\_(true),

participant\_(Participant()),

sequence\_number\_(1),

result\_(100)

{

number\_of\_performance\_++;

}

Performance::Performance(bool performance\_type, Participant participant, int sequence\_number, int result)

: performance\_type\_(performance\_type),

participant\_(participant),

sequence\_number\_(sequence\_number),

result\_(result)

{

number\_of\_performance\_++;

}

Performance::Performance(const Performance &copy)

: performance\_type\_(copy.performance\_type\_),

participant\_(copy.participant\_),

sequence\_number\_(copy.sequence\_number\_),

result\_(copy.result\_)

{

number\_of\_performance\_++;

}

Performance &Performance::operator=(const Performance &copy)

{

performance\_type\_ = copy.performance\_type\_;

participant\_ = copy.participant\_;

sequence\_number\_ = copy.sequence\_number\_;

result\_ = copy.result\_;

return \*this;

}

Performance::~Performance() {}

//Getters

bool Performance::GetPerformanceType() { return performance\_type\_; }

int Performance::GetNumberOfCompetitions() { return number\_of\_performance\_; }

Participant Performance::GetParticipant() { return participant\_; }

int Performance::GetSequenceNumber() { return sequence\_number\_; }

int Performance::GetResult() { return result\_; }

//Selectors

Performance &Performance::SetPerformanceType(bool performance\_type)

{

performance\_type\_ = performance\_type;

return \*this;

}

Performance &Performance::SetParticipant(Participant participant)

{

participant\_ = participant;

return \*this;

}

Performance &Performance::SetSequenceNumber(int sequence\_number)

{

sequence\_number\_ = sequence\_number;

return \*this;

}

Performance &Performance::SetResult(int result)

{

result\_ = result;

return \*this;

}

void Performance::Print()

{

// performance\_type\_= 0(false) => team performance

// performance\_type\_= 1(true) => individual performance

if (performance\_type\_ == false)

{

std::cout << "Team performance. ";

}

else

{

std::cout << "Individual performance. ";

}

std::cout << participant\_ << "(" << sequence\_number\_ << ", " << result\_ << ")" << std::endl;

}

std::ostream &operator<<(std::ostream &out, const Performance &obj)

{

// performance\_type\_= 0(false) => team performance

// performance\_type\_= 1(true) => individual performance

if (obj.performance\_type\_ == false)

{

out << "Team performance. ";

}

else

{

out << "Individual performance. ";

}

out << obj.participant\_ << "(" << obj.sequence\_number\_ << ", " << obj.result\_ << ")";

return out;

}

std::istream &operator>>(std::istream &in, Performance &obj)

{

//Performance\_type

std::cout << "Enter new performance type (1 - individual; 0 - team): ";

in >> obj.performance\_type\_;

//Participant

in >> obj.participant\_;

//Sequence number

std::cout << "Enter new sequence number for participant: ";

in >> obj.sequence\_number\_;

//Result

std::cout << "Enter new result for participant: ";

in >> obj.result\_;

return in;

}

**Клас “Competition”**

**Header:**

#ifndef \_TASK\_1\_INCLUDE\_TASK\_1\_COMPETITION\_H\_

#define \_TASK\_1\_INCLUDE\_TASK\_1\_COMPETITION\_H\_

#include <task\_1/performance.h>

class Competition

{

public:

Competition();

Competition(const char \*name);

Competition(const char \*name, Performance \*performance\_pointer, int performance\_size);

Competition(const Competition &copy);

Competition &operator=(const Competition &copy);

~Competition();

void AddPerformance(const Performance &performance);

const char \*GetName();

Competition &SetName(const char \*name);

void Print();

void PrintShortly();

friend std::ostream& operator<< (std::ostream &out, const Competition &obj);

friend std::istream& operator>> (std::istream &in, Competition &obj);

private:

char \*name\_;

Performance \*performance\_pointer\_;

int performance\_size\_;

}; //Class (Competition)

#endif //Header Guard

**Source file:**

// This is a personal academic project. Dear PVS-Studio, please check it.

// PVS-Studio Static Code Analyzer for C, C++ and C#: http://www.viva64.com

#include <task\_1/competition.h>

#include <cstring>

#include <string>

Competition::Competition()

: performance\_size\_(0), performance\_pointer\_(nullptr)

{

const char \*name = "Olympic Games";

name\_ = new char[strlen(name) + 1];

strcpy(name\_, name);

}

Competition::Competition(const char \*name)

: performance\_size\_(0), performance\_pointer\_(nullptr)

{

name\_ = new char[strlen(name) + 1];

strcpy(name\_, name);

}

Competition::Competition(const char \*name, Performance \*performance\_pointer, int performance\_size)

: performance\_size\_(performance\_size)

{

//copy name

name\_ = new char[strlen(name) + 1];

strcpy(name\_, name);

//copy performance\_pointer

performance\_pointer\_ = new Performance[performance\_size];

for (int i = 0; i < performance\_size; i++)

{

performance\_pointer\_[i] = performance\_pointer[i];

}

}

Competition::Competition(const Competition &copy)

: performance\_size\_(copy.performance\_size\_)

{

//copy name

name\_ = new char[strlen(copy.name\_) + 1];

strcpy(name\_, copy.name\_);

//copy performance\_pointer

performance\_pointer\_ = new Performance[copy.performance\_size\_];

for (int i = 0; i < copy.performance\_size\_; i++)

{

performance\_pointer\_[i] = copy.performance\_pointer\_[i];

}

}

Competition &Competition::operator=(const Competition &copy)

{

if (this != &copy)

{

//Name

if (strlen(name\_) != 0)

{

delete[] name\_;

}

name\_ = new char[strlen(copy.name\_) + 1];

strcpy(name\_, copy.name\_);

//Performance pointer

if (performance\_size\_ != 0)

{

delete[] performance\_pointer\_;

}

performance\_pointer\_ = new Performance[copy.performance\_size\_];

for (int i = 0; i < copy.performance\_size\_; i++)

{

performance\_pointer\_[i] = copy.performance\_pointer\_[i];

}

//Performance size

performance\_size\_ = copy.performance\_size\_;

} //Condition (this != &copy)

return \*this;

}

Competition::~Competition()

{

if (strlen(name\_) != 0)

{

delete[] name\_;

}

if (performance\_size\_ != 0)

{

delete[] performance\_pointer\_;

}

}

void Competition::AddPerformance(const Performance &performance)

{

Performance \*new\_ptr = new Performance[performance\_size\_ + 1];

for (int i = 0; i < performance\_size\_; i++)

{

new\_ptr[i] = performance\_pointer\_[i];

}

new\_ptr[performance\_size\_] = performance;

if (performance\_size\_ != 0)

{

delete[] performance\_pointer\_;

}

performance\_pointer\_ = new\_ptr;

performance\_size\_++;

}

const char \*Competition::GetName() { return name\_; }

Competition &Competition::SetName(const char \*name)

{

if (strlen(name\_) != 0)

{

delete[] name\_;

}

name\_ = new char[strlen(name) + 1];

strcpy(name\_, name);

return \*this;

}

void Competition::Print()

{

std::cout << name\_ << std::endl;

for (int i = 0; i < performance\_size\_; i++)

{

std::cout << i + 1 << ". " << performance\_pointer\_[i] << std::endl;

}

}

void Competition::PrintShortly()

{

std::cout << name\_ << std::endl;

for (int i = 0; i < performance\_size\_; i++)

{

std::cout << i + 1 << ". " << performance\_pointer\_[i].GetParticipant().GetSurname() << std::endl;

}

}

std::ostream &operator<<(std::ostream &out, const Competition &obj)

{

out << obj.name\_ << std::endl;

for (int i = 0; i < obj.performance\_size\_; i++)

{

out << i + 1 << ") " << obj.performance\_pointer\_[i] << std::endl;

}

return out;

}

std::istream &operator>>(std::istream &in, Competition &obj)

{

std::string buff\_string;

std::cout << "Enter new name of competition: ";

in >> buff\_string;

if (strlen(obj.name\_) != 0)

{

delete[] obj.name\_;

}

obj.name\_ = new char[buff\_string.length() + 1];

strcpy(obj.name\_, buff\_string.c\_str());

return in;

}

**Main file:**

// This is a personal academic project. Dear PVS-Studio, please check it.

// PVS-Studio Static Code Analyzer for C, C++ and C#: http://www.viva64.com

#include <task\_1/date.h>

#include <task\_1/participant.h>

#include <task\_1/performance.h>

#include <task\_1/competition.h>

#include <iostream>

void EditDate(Date &date)

{

int ch = -1;

while (ch != 0)

{

do

{

std::system("clear");

std::cout << date << std::endl;

std::cout << "1) SetDay" << std::endl;

std::cout << "2) SetMonth" << std::endl;

std::cout << "3) SetYear" << std::endl;

std::cout << "4) Enter new date" << std::endl;

std::cout << "0) Previous menu" << std::endl;

std::cin >> ch;

} while (ch != 0 && ch != 1 && ch != 2 && ch != 3 && ch != 4);

switch (ch)

{

case (1):

{

std::cout << "Current day: " << date.GetDay() << std::endl

<< "Enter new: ";

int day;

std::cin >> day;

date.SetDay(day);

break;

} //Case (1)

case (2):

{

std::cout << "Current month: " << date.GetMonth() << std::endl

<< "Enter new: ";

int month;

std::cin >> month;

date.SetMonth(month);

break;

} //Case (2)

case (3):

{

std::cout << "Current year: " << date.GetYear() << std::endl

<< "Enter new: ";

int year;

std::cin >> year;

date.SetYear(year);

break;

} //Case (3)

case (4):

{

std::cout << "Current date: " << date << std::endl

<< "Enter new: " << std::endl;

Date new\_date;

std::cin >> new\_date;

date = new\_date;

break;

} //Case (4)

} //Switch (ch)

} //While(ch != 0)

} //Function (EditDate)

void EditParticipant(Participant &participant)

{

int ch = -1;

while (ch != 0)

{

do

{

std::system("clear");

std::cout << participant << std::endl;

std::cout << "1) SetName" << std::endl;

std::cout << "2) SetSurname" << std::endl;

std::cout << "3) SetDate" << std::endl;

std::cout << "0) Previous menu" << std::endl;

std::cin >> ch;

} while (ch != 0 && ch != 1 && ch != 2 && ch != 3);

switch (ch)

{

case (1):

{

std::cout << "Current name: " << participant.GetName() << std::endl

<< "Enter new: ";

std::string name;

std::cin >> name;

participant.SetName(name.c\_str());

break;

} //Case (1)

case (2):

{

std::cout << "Current surname: " << participant.GetSurname() << std::endl

<< "Enter new: ";

std::string surname;

std::cin >> surname;

participant.SetSurname(surname.c\_str());

break;

} //Case (2)

case (3):

{

std::cout << "Current date: " << participant.GetDate() << std::endl

<< "Enter new: ";

Date date;

std::cin >> date;

participant.SetDate(date);

break;

} //Case (3)

} //Switch (ch)

} //While(ch != 0)

} //Function (EditParticipant)

void EditPerformance(Performance &performance)

{

int ch = -1;

while (ch != 0)

{

do

{

std::system("clear");

std::cout << performance << std::endl;

std::cout << "1) SetPerformanceType" << std::endl;

std::cout << "2) SetParticipant" << std::endl;

std::cout << "3) SetSequenceNumber" << std::endl;

std::cout << "4) SetResult" << std::endl;

std::cout << "0) Previous menu" << std::endl;

std::cin >> ch;

} while (ch != 0 && ch != 1 && ch != 2 && ch != 3 && ch != 4);

switch (ch)

{

case (1):

{

std::cout << "Current performance type(0 -> team, 1 -> individual): " << performance.GetPerformanceType() << std::endl

<< "Enter new(0 -> team, 1 -> individual): ";

bool performance\_type;

std::cin >> performance\_type;

performance.SetPerformanceType(performance\_type);

break;

} //Case (1)

case (2):

{

std::cout << "Current participant: " << performance.GetParticipant() << std::endl;

Participant participant;

std::cin >> participant;

performance.SetParticipant(participant);

break;

} //Case (2)

case (3):

{

std::cout << "Current sequence number: " << performance.GetSequenceNumber() << std::endl

<< "Enter new: ";

int sequence\_number;

std::cin >> sequence\_number;

performance.SetSequenceNumber(sequence\_number);

break;

} //Case (3)

case (4):

{

std::cout << "Current result: " << performance.GetResult() << std::endl

<< "Enter new: ";

int result;

std::cin >> result;

performance.SetResult(result);

break;

} //Case (4)

} //Switch (ch)

} //While(ch != 0)

} //Function (EditPerformance)

void EditCompetition(Competition &competition)

{

bool print\_shortly = false;

int ch = -1;

while (ch != 0)

{

do

{

std::system("clear");

if (print\_shortly == true)

{

competition.Print();

}

else

{

competition.PrintShortly();

}

std::cout << std::endl;

std::cout << "1) Switch (Print : PrintShortly)" << std::endl;

std::cout << "2) SetName" << std::endl;

std::cout << "3) AddPerformance" << std::endl;

std::cout << "0) Previous menu" << std::endl;

std::cin >> ch;

} while (ch != 0 && ch != 1 && ch != 2 && ch != 3);

switch (ch)

{

case (1):

{

print\_shortly = (!print\_shortly);

break;

} //Case (1)

case (2):

{

std::cout << "Current name: " << competition.GetName() << std::endl

<< "Enter new: ";

std::string name;

std::cin >> name;

competition.SetName(name.c\_str());

break;

} //Case (2)

case (3):

{

std::cout << "Enter performance to add: " << std::endl;

Performance performance;

std::cin >> performance;

competition.AddPerformance(performance);

break;

} //Case (3)

} //Switch (ch)

} //While(ch != 0)

} //Functon (EditCompetition)

int main()

{

Date date;

Participant participant;

Performance performance;

Competition competition;

int ch = -1;

while (ch != 0)

{

do

{

std::system("clear");

std::cout << "1) Date edit" << std::endl;

std::cout << "2) Participant edit" << std::endl;

std::cout << "3) Performance edit" << std::endl;

std::cout << "4) Competition edit" << std::endl;

std::cout << "0) Exit" << std::endl;

std::cin >> ch;

} while (ch != 0 && ch != 1 && ch != 2 && ch != 3 && ch != 4);

switch (ch)

{

case (1):

{

EditDate(date);

break;

} //case (1)

case (2):

{

EditParticipant(participant);

break;

} //case (2)

case (3):

{

EditPerformance(performance);

break;

} //case (3)

case (4):

{

EditCompetition(competition);

break;

} //case (4)

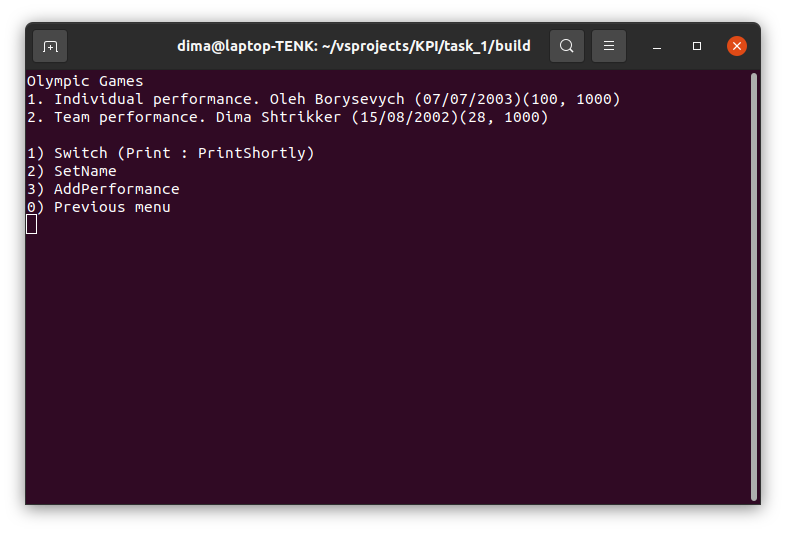
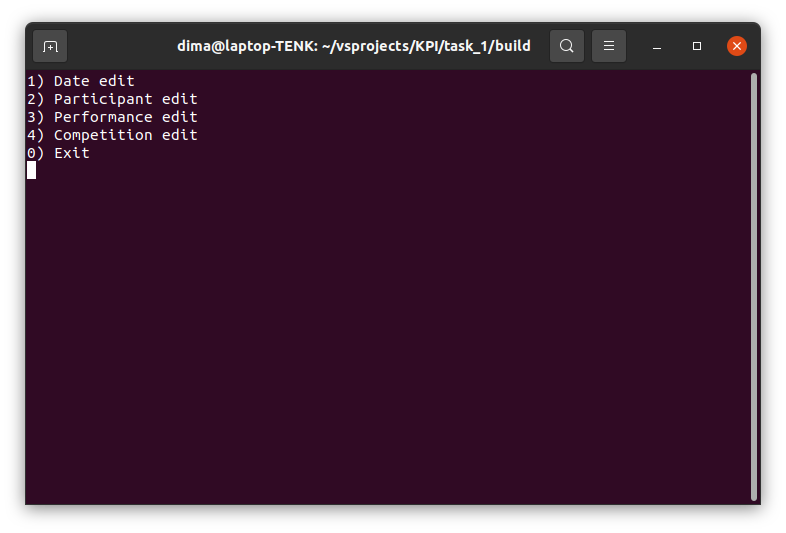
} //Switch (ch)

} //While (ch != 0)

return 0;

}

**Результати роботи програми:**

****

**Висновок:** Навчилися правильно описувати абстракції предметної області   
з використанням принципу інкапсуляції мовою С++.