Министерство науки и высшего образования Российской Федерации  
Пензенский государственный университет  
Кафедра вычислительная техника

**ОТЧËТ**  
по лабораторной работе №3  
по курсу «Объектно-ориентированное программирование»  
на тему «Механизмы наследования в языке C++»

Выполнили студенты группы 22ВВП1:  
Беляев Д.

Захаров А.

Приняли:  
Евсеева Ю.И.  
Гудков А. А.

Пенза 2024

**Название**

Механизмы наследования в языке C++

**Цель работы**

Изучить механизмы наследования в языке C++

**Лабораторное задание**

Создать абстрактный класс Испытание. На его основе реализовать классы Тест, Экзамен, Выпускной экзамен. Класс Испытание должен содержать атрибуты и методы, общие для производных классов. Основная программа должна создавать массивы объектов производных классов и выводить их на экран.

**Метод решения задачи**

Разработали классы согласно заданию, создали классы наследники и выполнили задания, связанные с ними

**Листинг**

**main.cpp**

#include <iostream>

#include <string>

#include "Challenge.h"

#include "Test.h"

#include "Exam.h"

#include "FinalExam.h"

#define ARRAY\_SIZE 2

Challenge\* CreateChallengeArray();

Test\* CreateTestArray();

Exam\* CreateExamArray();

FinalExam\* CreateFinalExamArray();

int main()

{

Challenge\* challenge = CreateChallengeArray();

Test\* test = CreateTestArray();

Exam\* exam = CreateExamArray();

FinalExam\* finalExam = CreateFinalExamArray();

std::cout << "Default print info: " << std::endl;

for (size\_t i = 0; i < ARRAY\_SIZE \* 4; i++)

{

switch (i)

{

case 0:

case 1:

challenge[i % ARRAY\_SIZE].PrintInfo();

break;

case 2:

case 3:

test[i % ARRAY\_SIZE].PrintInfo();

break;

case 4:

case 5:

exam[i % ARRAY\_SIZE].PrintInfo();

break;

case 6:

case 7:

finalExam[i % ARRAY\_SIZE].PrintInfo();

break;

}

}

std::cout << "\n";

Challenge\* allChallenges{ new Challenge[ARRAY\_SIZE \* 4] };

for (size\_t i = 0; i < ARRAY\_SIZE \* 4; i++)

{

switch (i)

{

case 0:

case 1:

allChallenges[i] = challenge[i % ARRAY\_SIZE];

break;

case 2:

case 3:

allChallenges[i] = test[i % ARRAY\_SIZE];

break;

case 4:

case 5:

allChallenges[i] = exam[i % ARRAY\_SIZE];

break;

case 6:

case 7:

allChallenges[i] = finalExam[i % ARRAY\_SIZE];

break;

}

}

std::cout << "Cast to base class (type 1): " << std::endl;

for (size\_t i = 0; i < ARRAY\_SIZE \* 4; i++)

allChallenges[i].PrintInfo();

std::cout << "\n";

std::cout << "Cast to base class (type 2): " << std::endl;

for (size\_t i = 0; i < ARRAY\_SIZE \* 4; i++)

{

switch (i)

{

case 0:

case 1:

static\_cast<Challenge>(challenge[i % ARRAY\_SIZE]).PrintInfo();

break;

case 2:

case 3:

static\_cast<Challenge>(test[i % ARRAY\_SIZE]).PrintInfo();

break;

case 4:

case 5:

static\_cast<Challenge>(exam[i % ARRAY\_SIZE]).PrintInfo();

break;

case 6:

case 7:

static\_cast<Challenge>(finalExam[i % ARRAY\_SIZE]).PrintInfo();

break;

}

}

return 0;

}

#pragma region Array create

Challenge\* CreateChallengeArray()

{

Challenge\* array{ new Challenge[ARRAY\_SIZE] };

std::cout << "Create challenge array" << std::endl;

for (int i = 0; i < ARRAY\_SIZE; i++)

{

std::string name = "";

int maxScore = -1;

std::cout << "Insert name and max score: ";

std::cin >> name >> maxScore;

array[i].SetName(name);

array[i].SetMaxScore(maxScore);

}

std::cout << "\n";

return array;

}

Test\* CreateTestArray()

{

Test\* array{ new Test[ARRAY\_SIZE] };

std::cout << "Create test array" << std::endl;

for (int i = 0; i < ARRAY\_SIZE; i++)

{

std::string name = "";

int maxScore = -1;

int passScore = -1;

std::cout << "Insert name, max score and pass score: ";

std::cin >> name >> maxScore >> passScore;

array[i].SetName(name);

array[i].SetMaxScore(maxScore);

array[i].SetPassScore(passScore);

}

std::cout << "\n";

return array;

}

Exam\* CreateExamArray()

{

Exam\* array{ new Exam[ARRAY\_SIZE] };

std::cout << "Create exam array" << std::endl;

for (int i = 0; i < ARRAY\_SIZE; i++)

{

std::string name = "";

int maxScore = -1;

int studentCount = -1;

std::cout << "Insert name, max score and student count: ";

std::cin >> name >> maxScore >> studentCount;

array[i].SetName(name);

array[i].SetMaxScore(maxScore);

array[i].SetStudentCount(studentCount);

}

std::cout << "\n";

return array;

}

FinalExam\* CreateFinalExamArray()

{

FinalExam\* array{ new FinalExam[ARRAY\_SIZE] };

std::cout << "Create final exam array" << std::endl;

for (int i = 0; i < ARRAY\_SIZE; i++)

{

std::string name = "";

int maxScore = -1;

int maxPassedCount = -1;

std::cout << "Insert name, max score and max passed count: ";

std::cin >> name >> maxScore >> maxPassedCount;

array[i].SetName(name);

array[i].SetMaxScore(maxScore);

array[i].SetPassedCount(maxPassedCount);

}

std::cout << "\n";

return array;

}

#pragma endregion

**challenge.h**

#ifndef CHALLENGE

#define CHALLENGE

#include <iostream>

#include <string>

class Challenge

{

public:

Challenge()

{

\_name = "";

\_maxScore = -1;

}

Challenge(int maxScore, std::string name);

~Challenge() { std::cout << "Challenge destroyed" << std::endl; }

virtual void PrintInfo();

// Setters

void SetName(std::string name) { \_name = name; }

void SetMaxScore(int maxScore) { \_maxScore = maxScore; }

//Getters

std::string GetName() { return \_name; }

int GetMaxScore() { return \_maxScore; }

protected:

std::string \_name;

int \_maxScore;

};

#endif // !CHALLENGE

**challenge.cpp**

#include <iostream>

#include "Challenge.h"

Challenge::Challenge(int maxScore, std::string name)

{

\_maxScore = maxScore;

\_name = name;

}

void Challenge::PrintInfo()

{

std::cout << "Challenge " << \_name << " Max score: " << \_maxScore << std::endl;

}

**test.h**

#ifndef TEST

#define TEST

#include "Challenge.h"

class Test : public Challenge

{

public:

Test() : Challenge() { \_passScore = -1; }

Test(int maxScore, std::string name, int passScore);

void PrintInfo();

// Setters

void SetPassScore(int passScore) { \_passScore = passScore; }

// Getters

int GetPassScore() { return \_passScore; }

private:

int \_passScore = -1;

};

#endif

**test.cpp**

#include "Test.h"

Test::Test(int maxScore, std::string name, int passScore) : Challenge(maxScore, name)

{

\_passScore = passScore;

}

void Test::PrintInfo()

{

std::cout << "Test " << \_name << " Max score: " << \_maxScore;

std::cout << " Pass if more then " << \_passScore << std::endl;

}

**exam.h**

#ifndef EXAM

#define EXAM

#include "Challenge.h"

class Exam : public Challenge

{

public:

Exam() : Challenge() { \_studentCount = -1; }

Exam(int maxScore, std::string name, int studentCount);

void PrintInfo();

// Setters

void SetStudentCount(int studentCount) { \_studentCount = studentCount; }

// Getters

int GetStudentCount() { return \_studentCount; }

private:

int \_studentCount = -1;

};

#endif

**exam.cpp**

#include "Exam.h"

Exam::Exam(int maxScore, std::string name, int studentCount) : Challenge(maxScore, name)

{

\_studentCount = studentCount;

}

void Exam::PrintInfo()

{

std::cout << "Exam " << \_name << " Max score: " << \_maxScore;

std::cout << " Students count " << \_studentCount << std::endl;

}

**finalExam.h**

#ifndef FINAL\_EXAM

#define FINAL\_EXAM

#include "Challenge.h"

class FinalExam : public Challenge

{

public:

FinalExam() : Challenge() { \_maxPassedCount = -1; }

FinalExam(int maxScore, std::string name, int passedCount);

void PrintInfo();

// Setters

void SetPassedCount(int maxPassedCount) { \_maxPassedCount = maxPassedCount; }

// Getters

int GetPassedCount() { return \_maxPassedCount; }

private:

int \_maxPassedCount = -1;

};

#endif

**finalExam.cpp**

#include "FinalExam.h"

FinalExam::FinalExam(int maxScore, std::string name, int passedCount) : Challenge(maxScore, name)

{

\_maxPassedCount = passedCount;

}

void FinalExam::PrintInfo()

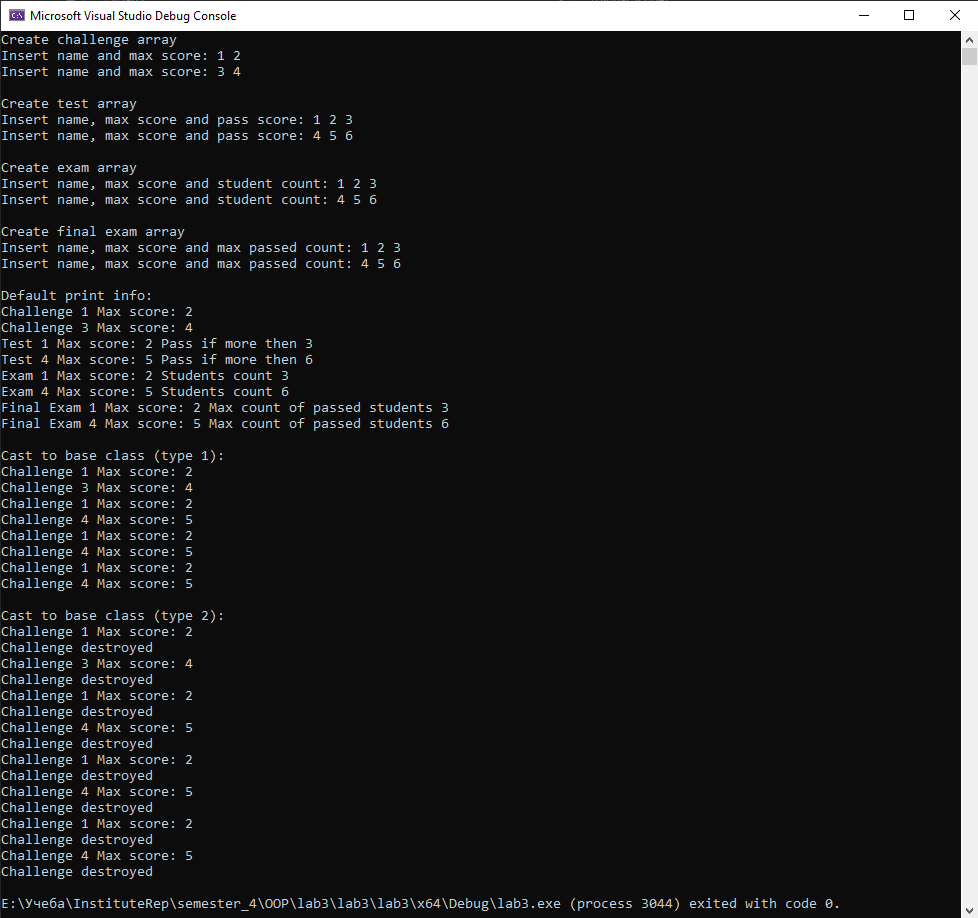
{

std::cout << "Final Exam " << \_name << " Max score: " << \_maxScore;

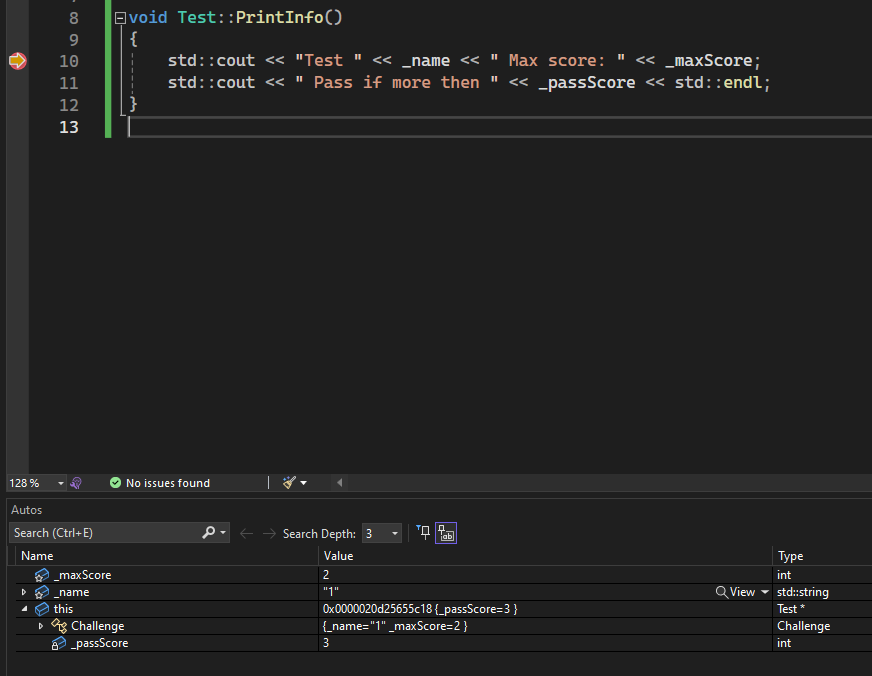
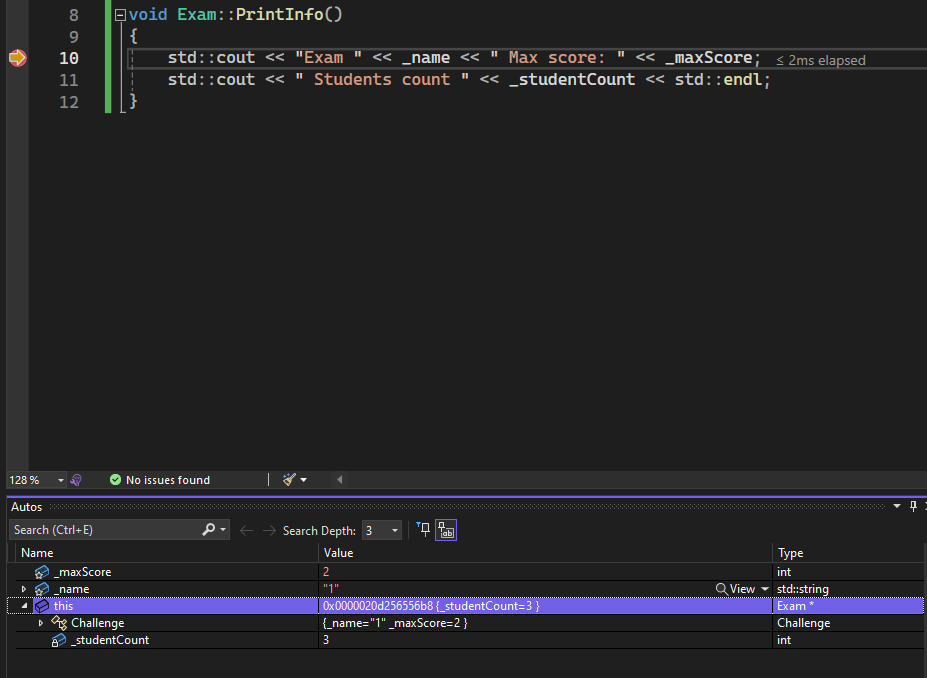
std::cout << " Max count of passed students " << \_maxPassedCount << std::endl;

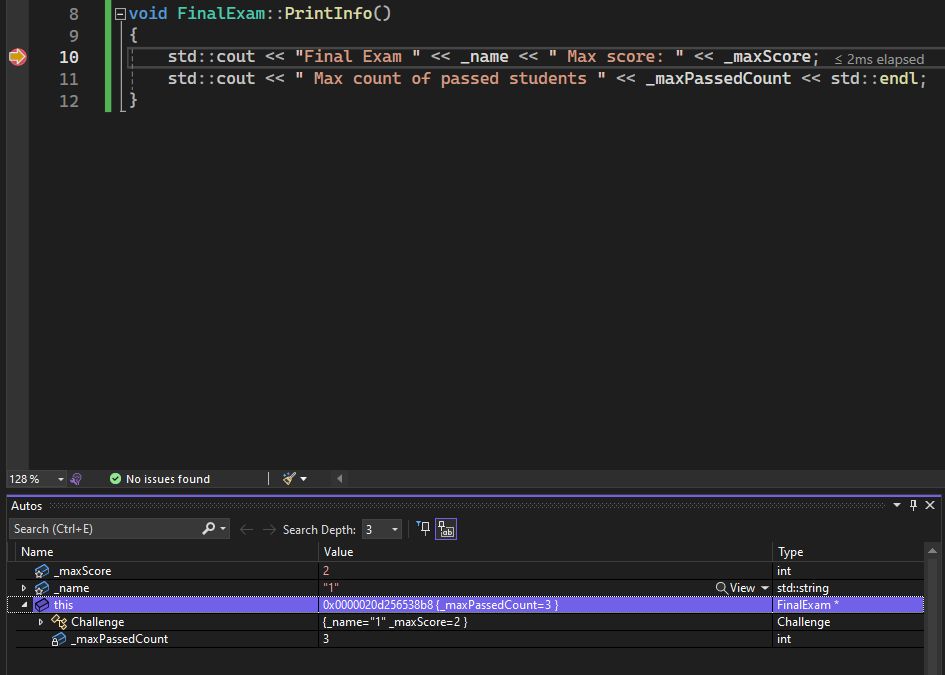
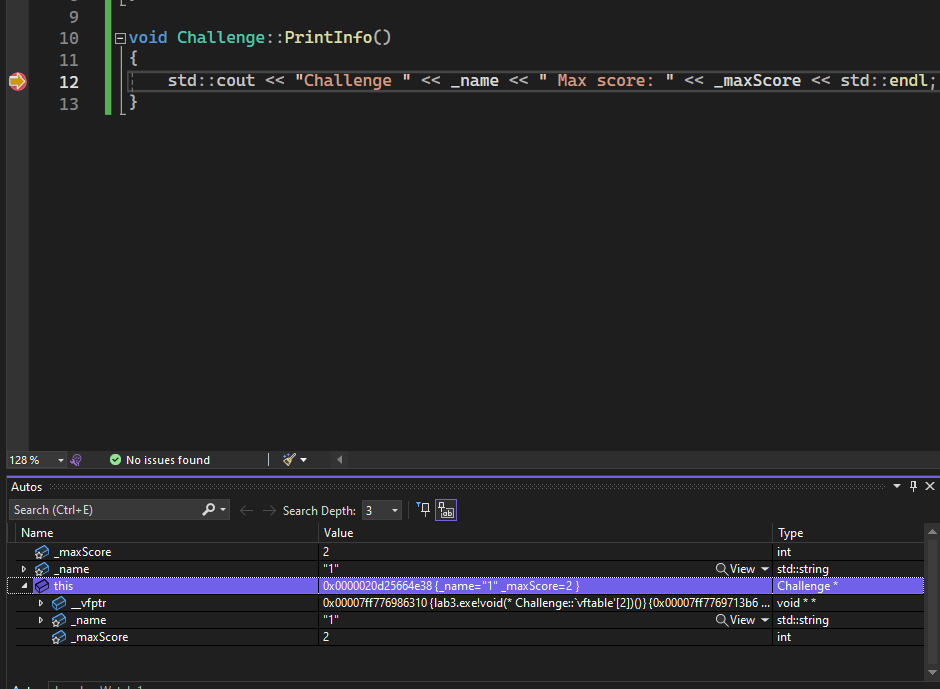
}

**Результат работы программы**



**Протокол трассировки программы**

**Вывод**

Мы изучили механизмы наследования в языке C++