Задача. Розробити класи для організації тестування знань учня з виконання операцій додавання та множення згідно схеми.

|  |  |
| --- | --- |
| Клас «BaseTester» | |
| Поля | * Ім’я учня * кількість завдань * кількість правильних відповідей * поточний номер тестового завдання * правильна відповідь поточного тестового завдання * тематика тестування |
| Методи | * showTask -- виведення поточного завдання на екран (абстрактний) * createTask -- генерування випадковим чином тестового завдання (абстрактний) * runTest -- проведення тестування кожного разу у користувача запитуємо правильну відповідь і порівнюємо з правильною (не абстрактний) |

|  |  |
| --- | --- |
| Клас «AddTester» (клас нащадок від BaseTester) | |
| Поля | додати мінімальне і максимальне значення діапазону для генерування чисел, які потрібно додати |
| Методи | * реалізувати showTask * реалізувати createTask |

|  |  |
| --- | --- |
| Клас «MultiplicationTester» (клас нащадок від BaseTester) | |
| Поля | перше число (перше число задається, а друге генерується від 0 до 9) |
| Методи | * реалізувати showTask * реалізувати createTask |

Розробити програму клієнт для тестування. У програмі створити функцію, яка через формальний параметр приймає об’єкт одного з класів (runTest) і ініціює проведення тестування (викликає метод).

------------------------ BaseTester.h -----------------

#pragma once

#include <string>

#include <iostream>

using namespace std;

class BaseTester

{

public:

string name;

int testsCount;

int rightAnswersCount;

int currentTestNumber;

int currentRightAnswer;

string subjectOfTesting;

virtual void showTask(string question) = 0;

virtual string createTask() = 0;

void runTest();

BaseTester(string name, int countTest, string subjectOfTesting);

};

------------------ BaseTester.cpp ---------------

#include "stdafx.h"

#include "BaseTester.h"

void BaseTester::runTest()

{

rightAnswersCount = 0;

for (currentTestNumber = 1; currentTestNumber <= testsCount; currentTestNumber++)

{

string question=createTask();

showTask(question);

int answer;

cin >> answer;

if (answer == currentRightAnswer)

{

rightAnswersCount++;

printf("Ok\n");

}

else

printf("Bed answer. Right = %d\n",currentRightAnswer);

}

}

BaseTester::BaseTester(string name, int countTest, string subjectOfTesting)

{

this->name = name;

testsCount = countTest;

this->subjectOfTesting = subjectOfTesting;

}

---------------------- AddTester.h ----------------------

#pragma once

#include "BaseTester.h"

class AddTester :

public BaseTester

{

public:

int minValue;

int maxValue;

AddTester(string name, int countTest, string subjectOfTesting, int minValue, int maxValue);

void showTask(string question);

string createTask();

};

---------------------- AddTester.cpp ----------------------

#include "stdafx.h"

#include "AddTester.h"

AddTester::AddTester(string name, int countTest, string subjectOfTesting, int minValue, int maxValue):

BaseTester(name,countTest,subjectOfTesting)

{

this->minValue = minValue;

this->maxValue = maxValue;

}

void AddTester::showTask(string question)

{

printf("------ Add tester:(current = %d, right aswers = %d ------)\n", currentTestNumber, rightAnswersCount);

printf("%s", question.data());

}

int getRandomNumber(int minV, int maxV)

{

return minV + rand() % (maxV-minV+1);

}

string AddTester::createTask()

{

int number1 = getRandomNumber(minValue, maxValue);

int number2 = getRandomNumber(minValue, maxValue);

currentRightAnswer = number1 + number2;

return to\_string(number1)+ " + "+ to\_string(number2) + " = ";

}

------------------------ MultTester.h ------------------

#pragma once

#include "BaseTester.h"

class MultTester :

public BaseTester

{

public:

int number1;

MultTester(string name, int countTest, string subjectOfTesting, int number1);

void showTask(string question);

string createTask();

};

------------------------ MultTester.cpp ------------------

#include "stdafx.h"

#include "MultTester.h"

MultTester::MultTester(string name, int countTest, string subjectOfTesting, int number1):

BaseTester(name,countTest, subjectOfTesting)

{

this->number1 = number1;

}

void MultTester::showTask(string question)

{

printf("------ Mult tester:(current = %d, right aswers = %d ------)\n", currentTestNumber, rightAnswersCount);

printf("%s", question.data());

}

string MultTester::createTask()

{

int number2 = rand()%10;

currentRightAnswer = number1 \* number2;

return to\_string(number1) + " \* " + to\_string(number2) + " = ";

}

-------------- main ------------------\

// ConsoleApplication8.cpp: определяет точку входа для консольного приложения.

//

#include "stdafx.h"

#include<iostream>

#include"AddTester.h";

#include"MultTester.h";

using namespace std;

void test(BaseTester\* tester)

{

tester->runTest();

printf("You result : %.2f \% \n",tester->rightAnswersCount\*100.0/

tester->testsCount);

}

int main()

{

printf("1. Add\n2.Mult\nYour choise:");

int ans;

cin >> ans;

BaseTester\* tester;

if (ans == 1)

tester = new AddTester("Ivan", 3, "Addd", 1, 9);

else

tester = new MultTester("Ivan", 3, "Multt", 7);

//-------

test(tester);

system("pause");

return 0;

}