
سلسلة الخوارزميات وحل المشاكل - المستوى الثاني
PROGRAMMING ADVICES
LEARN THE RIGHT WAY
26+ Years of Experience
Mohammed Abu-Hadhoud
MBA, PMOC, PgMP®, PMP®, PMI-RMP®, CM, ITILF, MCPD, MCSD
حقوق النشر محفوظة، أسعار الكورسات في المنصة هي أسعار رمزيه جدا، ارجو عدم نشر هذه الوثيقة لان نشرها سيمنعنا من الاستمرار في تقديم العلم للآخرين
ارجو عدم استخدام هذه الوثيقة من غير وجه حق لأنك ستحرم الاف الناس من التعلم
ProgrammingAdvices.com

```cpp
#include<iostream>

using namespace std;

enum enQuestionsLevel { EasyLevel = 1, MedLevel = 2, HardLevel =
3, Mix = 4 };

enum enOperationType { Add = 1, Sub = 2, Mult = 3, Div = 4, MixOp
= 5 };

string GetOpTypeSymbol(enOperationType OpType)
{
	switch (OpType)
	{
	case enOperationType::Add:
		return "+";
	case enOperationType::Sub:
		return "-";
	case enOperationType::Mult:
		return "x";
	case enOperationType::Div:
		return "/";
	default:
		return "Mix";
	}
}

string GetQuestionLevelText(enQuestionsLevel QuestionLevel)
{
	string arrQuestionLevelText[4] = { "Easy","Med","Hard","Mix"
};
	return arrQuestionLevelText[QuestionLevel - 1];
}

int RandomNumber(int From, int To)
{
	//Function to generate a random number
	int randNum = rand() % (To - From + 1) + From;
	return randNum;
}
```

```
void SetScreenColor(bool Right)
{
	if (Right)
		system("color 2F"); //turn screen to Green
	else
	{
		system("color 4F"); //turn screen to Red
		cout << "\a";
	}
}

short ReadHowManyQuestions()
{
	short NumberOfQuestions;
	do
	{
		cout << "How Many Questions do you want to answer ? ";
		cin >> NumberOfQuestions;
	} while (NumberOfQuestions < 1 || NumberOfQuestions >10);

	return NumberOfQuestions;
}

enQuestionsLevel ReadQuestionsLevel()
{
	short QuestionLevel = 0;
	do
	{
		cout << "Enter Questions Level [1] Easy, [2] Med, [3]
Hard, [4] Mix ? ";
		cin >> QuestionLevel;
	} while (QuestionLevel < 1 || QuestionLevel >4);

	return (enQuestionsLevel) QuestionLevel;
}

enOperationType ReadOpType()
{
	short OpType;
	do
	{
		cout << "Enter Operation Type [1] Add, [2] Sub, [3] Mul,
[4] Div, [5] Mix ? ";
		cin >> OpType;
	} while (OpType < 1 || OpType >5);

	return (enOperationType) OpType;
}
```

```
struct stQuestion
{
	int Number1 = 0;
	int Number2 = 0;
	enOperationType OperationType;
	enQuestionsLevel QuestionLevel;
	int CorrectAnswer = 0;
	int PlayerAnswer = 0;
	bool AnswerResult = false;
};

struct stQuizz
{
	stQuestion QuestionList[100];
	short NumberOfQuestions;
	enQuestionsLevel QuestionsLevel;
	enOperationType OpType;
	short NumberOfWrongAnswers = 0;
	short NumberOfRightAnswers = 0;
	bool isPass = false;
};

int SimpleCalculator(int Number1, int Number2, enOperationType
OpType)
{
	switch (OpType)
	{
	case enOperationType::Add:
		return Number1 + Number2;
	case enOperationType::Sub:
		return Number1 - Number2;
	case enOperationType::Mult:
		return Number1 * Number2;
	case enOperationType::Div:
		return Number1 / Number2;
	default:
		return Number1 + Number2;
	}
}

enOperationType GetRandomOperationType()
{
	int Op = RandomNumber(1, 4);
	return (enOperationType)Op;
}
```

```
stQuestion GenerateQuestion(enQuestionsLevel QuestionLevel,
enOperationType OpType)
{
      stQuestion Question;

      if (QuestionLevel == enQuestionsLevel::Mix)
      {
            QuestionLevel = (enQuestionsLevel) RandomNumber(1, 3);
      }

      if (OpType == enOperationType::MixOp)
      {
            OpType = GetRandomOperationType();
      }

      Question.OperationType = OpType;

      switch (QuestionLevel)
      {
      case enQuestionsLevel::EasyLevel:
            Question.Number1 = RandomNumber(1, 10);
            Question.Number2 = RandomNumber(1, 10);

            Question.CorrectAnswer =
            SimpleCalculator(Question.Number1, Question.Number2,
            Question.OperationType);

            Question.QuestionLevel = QuestionLevel;
            return Question;

      case enQuestionsLevel::MedLevel:
            Question.Number1 = RandomNumber(10, 50);
            Question.Number2 = RandomNumber(10, 50);

            Question.CorrectAnswer =
            SimpleCalculator(Question.Number1, Question.Number2,
            Question.OperationType);

            Question.QuestionLevel = QuestionLevel;
            return Question;
```

```
        case enQuestionsLevel::HardLevel:
            Question.Number1 = RandomNumber(50, 100);
            Question.Number2 = RandomNumber(50, 100);

            Question.CorrectAnswer =
            SimpleCalculator(Question.Number1, Question.Number2,
            Question.OperationType);

            Question.QuestionLevel = QuestionLevel;
            return Question;
        }

        return Question;
}

void GenerateQuizzQuestions(stQuizz& Quizz)
{
        for (short Question = 0; Question < Quizz.NumberOfQuestions;
Question++)
        {
            Quizz.QuestionList[Question] =
GenerateQuestion(Quizz.QuestionsLevel, Quizz.OpType);
        }

}

int ReadQuestionAnswer()
{
        int Answer = 0;
        cin >> Answer;
        return Answer;
}

void PrintTheQuestion(stQuizz& Quizz, short QuestionNumber)
{
        cout << "\n";
        cout << "Question [" << QuestionNumber + 1 << "/" <<
Quizz.NumberOfQuestions << "] \n\n";
        cout << Quizz.QuestionList[QuestionNumber].Number1 << endl;
        cout << Quizz.QuestionList[QuestionNumber].Number2 << " ";
        cout <<
GetOpTypeSymbol(Quizz.QuestionList[QuestionNumber].OperationType);
        cout << "\n_________" << endl;
}
```

```
void CorrectTheQuestionAnswer(stQuizz& Quizz, short
QuestionNumber)

{
     if (Quizz.QuestionList[QuestionNumber].PlayerAnswer !=
     Quizz.QuestionList[QuestionNumber].CorrectAnswer)
     {
          Quizz.QuestionList[QuestionNumber].AnswerResult = false;
          Quizz.NumberOfWrongAnswers++;

          cout << "Worng Answer :-( \n";
          cout << "The right answer is: ";
          cout <<Quizz.QuestionList[QuestionNumber].CorrectAnswer;
          cout << "\n";
     }
     else
     {
          Quizz.QuestionList[QuestionNumber].AnswerResult = true;
          Quizz.NumberOfRightAnswers++;

          cout << "Right Answer :-) \n";
     }
     cout << endl;

SetScreenColor(Quizz.QuestionList[QuestionNumber].AnswerResult);

}

void AskAndCorrectQuestionListAnswers(stQuizz& Quizz)
{

     for (short QuestionNumber = 0; QuestionNumber <
Quizz.NumberOfQuestions; QuestionNumber++)
     {

          PrintTheQuestion(Quizz, QuestionNumber);

          Quizz.QuestionList[QuestionNumber].PlayerAnswer =
          ReadQuestionAnswer();

          CorrectTheQuestionAnswer(Quizz, QuestionNumber);

     }

     Quizz.isPass = (Quizz.NumberOfRightAnswers >=
     Quizz.NumberOfWrongAnswers);

}
```

```
string GetFinalResultsText(bool Pass)
{
     if (Pass)
          return "PASS :-)";
     else
          return "Fail :-(";
}

void PrintQuizzResults(stQuizz Quizz)
{
     cout << "\n";
     cout << "_____________________________\n\n";

cout << " Final Resutls is " <<
GetFinalResultsText(Quizz.isPass);
     cout << "\n_____________________________\n\n";

cout << "Number of Questions: " << Quizz.NumberOfQuestions <<
endl;
     cout << "Questions Level    : " <<
GetQuestionLevelText(Quizz.QuestionsLevel) << endl;

     cout << "OpType             : " <<
GetOpTypeSymbol(Quizz.OpType) << endl;

     cout << "Number of Right Answers: " <<
Quizz.NumberOfRightAnswers << endl;

     cout << "Number of Wrong Answers: " <<
Quizz.NumberOfWrongAnswers << endl;

     cout << "_____________________________\n";
}

void PlayMathGame()
{
     stQuizz Quizz;

     Quizz.NumberOfQuestions = ReadHowManyQuestions();
     Quizz.QuestionsLevel = ReadQuestionsLevel();
     Quizz.OpType = ReadOpType();

     GenerateQuizzQuestions(Quizz);
     AskAndCorrectQuestionListAnswers(Quizz);
     PrintQuizzResults(Quizz);
}
```

```
void ResetScreen()
{
	system("cls");
	system("color 0F");
}

void StartGame()
{
	char PlayAgain = 'Y';
	do
	{
		ResetScreen();
		PlayMathGame();

		cout << endl << "Do you want to play again? Y/N? ";
		cin >> PlayAgain;

	} while (PlayAgain == 'Y' || PlayAgain == 'y');
}

int main()
{
	//Seeds the random number generator in C++, called only once
	srand((unsigned)time(NULL));

	StartGame();

	return 0;
}
```