Author: Michael LaPan  
Assignment: Program 6  
Dis: is a quiz program, randomly generates questions, both from files or random numbers. It has two modes, quiz mode- doesn’t show key or teacher mode – shows key while taking the quiz

Table Of contents:  
.h’s  
 MathProblem.h  
 quiz.h  
 quizItem.h  
 triviaQuestion.h  
 wordScramble.h  
.cpp’s  
 MathProblem.cpp  
 quiz.cpp  
 quizItem.cpp  
 triviaQuestion.cpp  
 wordScramble.cpp  
 source.cpp

MathProblem.h  
  
#ifndef MATHPORBLEM\_H

#include <string>

#include "quizItem.h"

using namespace std;

/\*\*

\*this is the math problem class

\*this class is responsable for generating a math question

\*for the quiz. it also has a base class of quizItem.

\*/

class mathProblem : public quizItem

{

private:

double first;//first part of the math eq, (used to calc ans)

double last;//last part of the math eq, (used to calc ans)

char operand;//the randomly chosen operand

string answer;//answer in string form

string firstPart;//first part in string form

string secondPart;//last part in string form

public:

mathProblem();//constructor

void setFirst(double toSet);//sets the first part

void setLast(double toSet);//sets the second part

void setOperand(char toSet);//sets the operand

void setFirstPart(string toSet);//sets the first part as string

void setSecondPart(string toSet);//sets the second part as string

double getFirst( );//returns the first part

double getLast( );//returns the second part

char getOperand( );//returns the operand

string getFirstPart( );//returns the first part as string

string getSecondPart( );//returns the second part as string

void randFirstSecondOp();//randomly chooses the first and second number, and their operand

void GenerateQuestion();//makes the question

};

#define MATHPORBLEM\_H

#endif

Quiz.h  
  
#include <string>

#include "quizItem.h"

#include "wordScramble.h"

#include "MathProblem.h"

#include "triviaQuestion.h"

using namespace std;

/\*\*

\*this is the quiz class

\*the quiz class is what holds all of the quiz items,

\*and calls the correct functions to ask the questions.

\*/

class quiz

{

private:

quizItem \*\*QIArray; //pointer to a soon to be array of quiz items

string \*QIKarray;//pointer to a soon to be array of strings

int size; //size of the array

public:

quiz();//constructor

~quiz();//destructor

void popArray();//populates the array with quiz items

int randArraySize();//randomly sets the array size

string newQuestion();//makes a new questoin of random type

void userIO();//where all user in and out is located.

};

QuizItem.h  
  
#ifndef QUIZITEM\_H

#include <string>

/\*\*

\*this is the quizItem class

\*this is the base class for most all other classes

\*it is responsable for, holding the question's, and

\*answers to the quiz, and also has a virtual function

\*generateQuestoin

\*/

using namespace std;

class quizItem

{

private:

string questionQuiz;//quiz question

string answerQuiz;//quis answer

string answerScrambled;//answer to scrambled question

public:

//GenerateQuestion is the virtual function that all classes have

//this will generate the question for any of its class, that

//inherates quizItem

virtual void GenerateQuestion();

void buildQ();// outputs the question

void setQuestion(string toSet);//set the question

void setAnswer(string toSet);// set the answer

void setAnswerScrambled(string toSet);//set the scrambled words answer

string getQuestion();//get the question

string getAnswer();//get the answer

string getAnswerScrambled();//gets the scrambled words answer.

};

#define QUIZITEM\_H

#endif

triviaQuestion.h  
  
#ifndef TRIVIAQUESTION\_H

#include <string>

#include "quizItem.h"

/\*\*

\*this is the triviaQuestion class

\*this class is responsable for, generating a triva

\*question, reading all questions and answers from a file

\*providing both the question and answer. this class has

\*a base class of quizItem

\*/

using namespace std;

class triviaQuestion : public quizItem

{

private:

string \*qArray;//pointer to a string, to be turned into an dynamicly sized array

string \*aArray;//pointer to a string, to be turned into an dynamicly sized array

int size;//size of the array

public:

triviaQuestion();//constructor

~triviaQuestion();//descructor

void GenerateQuestion();//makes the question

void popArray();//reades the file into the array

int getFileSize();//gets the size of the file, and then sets array size to it.

};

#define TRIVIAQUESTION\_H

#endif

wordScremble.h  
  
#ifndef WORDSCRAMBLE\_H

#include <string>

#include "quizItem.h"

using namespace std;

/\*\*

\*this is the word scramble class

\*this class is responsable for, reading all the words

\*from thee comWords.txt, jumbleing them up, and making

\*the question. this class has a base class of quizItem

\*/

class wordScramble : public quizItem

{

private:

string \*ansArray; //pointer to a string, to be turned into an dynamicly sized array

string \*ansArrayScram; //pointer to a string, to be turned into an dynamicly sized array

int size; //the size of the array

public:

wordScramble();//constructor

~wordScramble();//destructor

void GenerateQuestion();//generates the question to ask

void popArray();//populates the array

int getFileSize();//gets the size to use for the array

void jumble();//jumbles up the word

};

#define WORDSCRAMBLE\_H

#endif

mathProblem.cpp  
  
#include <string>

#include <iostream>

#include <stdlib.h>

#include "MathProblem.h"

using namespace std;

void mathProblem::setFirst(double toSet)

{

first = toSet;

}

void mathProblem::setLast(double toSet)

{

last = toSet;

}

void mathProblem::setOperand(char toSet)

{

operand = toSet;

}

void mathProblem::setFirstPart(string toSet)

{

firstPart = toSet;

}

void mathProblem::setSecondPart(string toSet)

{

secondPart = toSet;

}

double mathProblem::getFirst()

{

return first;

}

double mathProblem::getLast()

{

return last;

}

char mathProblem::getOperand()

{

return operand;

}

string mathProblem::getFirstPart()

{

return firstPart;

}

string mathProblem::getSecondPart()

{

return secondPart;

}

/\*

\*randomly generates, the first and second number.

\*then generates a number from 0-4, and that is put in

\*to the switch. depending on what the number was,

\*itll assign and preform calc with that operand

\*/

void mathProblem::randFirstSecondOp()

{

first = rand() % 999 + 1;

last = rand() % 999 + 1;

firstPart = to\_string((int)first);

secondPart = to\_string((int)last);

int temp = rand() % 5;

switch (temp)

{

case 0:

operand = '+';

answer = to\_string((int)(first + last));

break;

case 1:

operand = '-';

answer = to\_string((int)(first - last));

break;

case 2:

operand = 'X';

answer = to\_string((int)(first \* last));

break;

case 3:

operand = '/';

answer = to\_string((first / last)) ;

break;

case 4:

operand = '%';//wouldnt be programming w/o mod!

answer = to\_string(((int)first % (int)last));

break;

}

}

/\*

\*sets te question and answer

\*/

void mathProblem::GenerateQuestion()

{

setQuestion("Calculate " + firstPart + " " + operand + " " + secondPart + "\nfor division must include leading 0,\n need to calculate to sith decimal (0.######)");

setAnswer(answer);

}

/\*

\*!!constructor!!

\*calls the method to randomly make first second number and operator.

\*/

mathProblem::mathProblem()

{

randFirstSecondOp();

}

Quiz.cpp  
  
#include <string>

#include <iostream>

#include "quiz.h"

#include <time.h>

using namespace std;

/\*

\*!!constructor!!

\*this is where i get the arrays size,

\*and create the quiz array, and the quiz key array

\*and then i populate both arrays

\*/

quiz::quiz()

{

srand(time(NULL));//sets the seed for a little more random

size = randArraySize();

QIArray = new quizItem \*[size];

QIKarray = new string[size];

popArray();

}

/\*

\*HOUSEKEEPING!

\*deletes the arrays made with new

\*/

quiz::~quiz()

{

delete[] QIArray;

delete[] QIKarray;

}

/\*

\*this is where the questions are asked and the user

\*has a chance to get it right. if he doesnt get it

\*it will ask if they want to try again. also has option

\*to show keey while taking the quiz

\*/

void quiz::userIO()

{

string sToC;

int i = 0, again = 0, tp = 0;

cout << "see key with test? (enter teacher pass)" << endl << endl;

cin >> tp;

do

{

if (tp==42)//answer to everything

{

//clear the cin buffer. getline was getting skipped

//due to random return in the buffer, have to clear

//for getline to be encountered

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

QIArray[i]->buildQ();

cout << " Answer: " << QIKarray[i] << endl;;

getline(cin, sToC);

if (sToC == QIKarray[i])

{

cout << "good job!! you got the question right!\n" << endl;

i++;

}

else

{

cout << "your answer is wrong, try again? (1 for yes, 0 for no)\n" << endl;

cin >> again;

if (again == 0)

{

i++;

}

}

}

else

{

//clear the cin buffer. getline was getting skipped

//due to random return in the buffer, have to clear

//for getline to be encountered

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

QIArray[i]->buildQ();

getline(cin, sToC);

if (sToC == QIKarray[i])

{

cout << "good job!! you got the question right!\n" << endl;

i++;

}

else

{

cout << "your answer is wrong, try again? (1 for yes, 0 for no)\n" << endl;

cin >> again;

if (again == 0)

{

i++;

}

}

}

} while (i != size);

}

/\*

\*populates the array with random questions

\*/

void quiz::popArray()

{

string randAns;

for (size\_t i = 0; i < size; i++)

{

randAns = newQuestion();

if (randAns == "w")

{

QIArray[i] = new wordScramble();

QIArray[i]->GenerateQuestion();

QIKarray[i] = QIArray[i]->getAnswer();

}

else if (randAns == "m")

{

QIArray[i] = new mathProblem();

QIArray[i]->GenerateQuestion();

QIKarray[i] = QIArray[i]->getAnswer();

}

else if (randAns == "t")

{

QIArray[i] = new triviaQuestion();

QIArray[i]->GenerateQuestion();

QIKarray[i] = QIArray[i]->getAnswer();

}

}

userIO();

}

//randomly makes the array size

int quiz::randArraySize()

{

return rand() % 20 + 10;

}

/\*

\*randomly picks the type of question

\*returns what the random type should be

\*/

string quiz::newQuestion()

{

int randQ = rand() % 3;

switch (randQ)

{

case 0:

return "m";

break;

case 1:

return "t";

break;

case 2:

return "w";

break;

}

}

quizItem.cpp

#include <string>

#include <time.h>

#include <iostream>

#include "quizItem.h"

using namespace std;

/\*

\*does nothing

\*/

void quizItem::GenerateQuestion()

{

cout << "in the gen questoin" << endl;

}

void quizItem::setQuestion(string toSet)

{

questionQuiz = toSet;

}

void quizItem::setAnswer(string toSet)

{

answerQuiz = toSet;

}

void quizItem::setAnswerScrambled(string toSet)

{

answerScrambled = toSet;

}

string quizItem::getQuestion()

{

return questionQuiz;

}

string quizItem::getAnswer()

{

return answerQuiz;

}

string quizItem::getAnswerScrambled()

{

return answerScrambled;

}

/\*

\*outputs the question

\*/

void quizItem::buildQ()

{

cout << questionQuiz << " " << answerScrambled << endl;

}

triviaQuestion.cpp

#include <iostream>

#include <fstream>

#include <stdlib.h>

#include <time.h>

#include "triviaQuestion.h"

using namespace std;

/\*

\*!!constructor!!

\*i get the file size, and the create my two arrays

\*with that file size. size/2 is becuase every other

\*item in the file is the questoin. i them populate

\*the array.

\*/

triviaQuestion::triviaQuestion()

{

size = getFileSize();

qArray = new string[size / 2 + 1];

aArray = new string[size / 2 + 1];

popArray();

}

/\*

\*HOUSE KEEPING!

\*delete the arrays i made with new

\*/

triviaQuestion::~triviaQuestion()

{

delete[] qArray;

delete[] aArray;

}

/\*

\*reads from the file placeing itmes into the array

\*in the patteren of every other. first item in file

\*is the question, second is answer.

\*/

void triviaQuestion::popArray()

{

int i = 0;

ifstream inFile("trivia.txt");

string inputLine;

if (inFile.fail()) // Test for file existence

{

cout << "Problem opening file";

system("pause");

exit(-1);

}

while (!inFile.eof())

{

getline(inFile, inputLine); //continuation read

qArray[i] = inputLine;

getline(inFile, inputLine); //continuation read

aArray[i] = inputLine;

i++;

}

}

/\*

\*randomly picks a question and answer from the

\*question and answer array, thens sets them.

\*/

void triviaQuestion::GenerateQuestion()

{

int randAns = rand() % 36;

setAnswer(aArray[randAns]);

setQuestion(qArray[randAns]);

}

/\*

\*loops through the file getting the length of the file

\*used in the creation of the arrays

\*/

int triviaQuestion::getFileSize()

{

int i = 0;

ifstream inFile("trivia.txt");

string inputLine;

if (inFile.fail()) // Test for file existence

{

cout << "Problem opening file";

system("pause");

exit(-1);

}

//priming read

while (!inFile.eof())

{

getline(inFile, inputLine); //continuation read

i++;

}

return i;

}

wordScramble.cpp  
  
#include "wordScramble.h"

#include <stdlib.h>

#include <fstream>

#include <iostream>

#include <string>

#include <algorithm>

using namespace std;

/\*

\*reads from the compWords file and places the words

\*into an array

\*/

void wordScramble::popArray()

{

int i = 0;

ifstream inFile("compWords.txt");

string inputLine;

if (inFile.fail()) // Test for file existence

{

cout << "Problem opening file";

system("pause");

exit(-1);

}

while (!inFile.eof())

{

getline(inFile, inputLine); //continuation read

ansArray[i] = inputLine;

i++;

}

}

/\*

\*gets the size of the file, so i can set the size of the array

\*/

int wordScramble::getFileSize()

{

int i = 0;

ifstream inFile("compWords.txt");

string inputLine;

if (inFile.fail()) // Test for file existence

{

cout << "Problem opening file";

system("pause");

exit(-1);

}

//priming read

while (!inFile.eof())

{

getline(inFile, inputLine); //continuation read

i++;

}

return i;

}

/\*

\*generates the question, mostly just setting the

\*question and answer. it randomly picks a question

\*and answer.

\*/

void wordScramble::GenerateQuestion()

{

int num = rand() % size;

setQuestion("Unscramble this word");

setAnswer(ansArray[num]);

setAnswerScrambled(ansArrayScram[num]);

}

/\*

\*this function jumbles up a string in the ansArray

\*and places it in to the scrammbled array.

\*i jumble the string using random\_shuffle from

\*algorithms

\*/

void wordScramble::jumble()

{

string temp ;

for (size\_t i = 0; i < size - 1; i++)

{

temp = ansArray[i];

random\_shuffle(temp.begin(), temp.end());

ansArrayScram[i] = temp;

}

}

/\*

\*!!constructor!!

\*this is where both arrays are given their sizes

\*also all calls are made from here so everything

\*is ready, and no other unnesscary calls need to

\*be made

\*/

wordScramble::wordScramble()

{

size = getFileSize();

ansArray = new string[size + 1];

ansArrayScram = new string[size + 1];

popArray();

jumble();

}

/\*

\*HOUSE KEEPING!

\*delete the arrays i made with new to free up ram

\*/

wordScramble::~wordScramble()

{

delete[] ansArray;

delete[] ansArrayScram;

}

Source.cpp

#include "quiz.h"

/\*

\*driver class, calls the quiz

\*/

int main() {

quiz quizlet;

quizlet.userIO();

system("pause");

return 0;

}

