Задача. Відгадай слово

|  |  |
| --- | --- |
| Відгадай слово |  |
| Поля(властивості) | Поле, у якому зберігається слово  Масив, у якому зберігаються букви, які вже вгадав користувач |
| Методи (дії) | Перевірка чи є вказана буква у слові  Підрахунок кількості вказаної букви у слові  Метод перевірки і фіксації вгадування букви  Метод для перевірки того, чи є ще невгадані букви  Виведення ігрового поля (букви, які вгадано відображаються, а які не вгадані – відображаються зірочкою) |

На основі цього класу зробити клас, у якому додатково додано поле «підказка» та метод виведення підказки.

#pragma once

#include<string>

using namespace std;

class GuessWord

{

private:

string word;

void set\_Word(string value);

string guessedCharacters="";

int guessedLettersCount = 0;

public:

string get\_Word();

bool containsLetter(char letter);

int getLetterOcurenceCount(char letter);

int checkLetter(char letter);

void pringField();

bool isNotGuessedLetter();

GuessWord(string word);

};

#include "stdafx.h"

#include "GuessWord.h"

bool isAllCharsAlpha(string value)

{

//value.length -кількість символів у рядку

for (int i = 0; i < value.length(); i++)

{

if ( ! isalpha(value[i])) //Якщо не буква

{

return false;

}

}

return true;

}

void GuessWord::set\_Word(string value)

{

if (isAllCharsAlpha(value))

{

word = value;

}

else

throw "All symbol must be character";

}

string GuessWord::get\_Word()

{

return word;

}

int indexOf(string word, char letter)

{

int wordLength = word.length();

for (int i = 0; i < wordLength; i++)

{

if (word[i]==letter)

{

return i;

}

}

return -1;

}

bool GuessWord::containsLetter(char letter)

{

//return word.find(letter) != std::string::npos;

return indexOf(word, letter) != -1;

}

int GuessWord::getLetterOcurenceCount(char letter)

{

int count = 0;

int wordLength = word.length();

for (int i = 0; i < wordLength; i++)

{

if (word[i] == letter)

{

count++;

}

}

return count;

}

int GuessWord::checkLetter(char letter)

{

if (containsLetter(letter) && indexOf(guessedCharacters,letter)==-1 )

{

guessedCharacters += letter;

int ocurenceCount= getLetterOcurenceCount(letter);

guessedLettersCount += ocurenceCount;

return ocurenceCount;

}

return 0;

}

void GuessWord::pringField()

{

int wordLength = word.length();

printf("--------------------\n|");

for (int i = 0; i < wordLength; i++)

{

if (indexOf(guessedCharacters, word[i]) == -1)

{

printf(" \* |");

}

else

printf(" %c |", word[i]);

}

printf("\n--------------------\n");

}

bool GuessWord::isNotGuessedLetter()

{

return word.length()>guessedLettersCount;

}

GuessWord::GuessWord(string word)

{

set\_Word(word);

}

#pragma once

#include "GuessWord.h"

class GussWordWithTip :

public GuessWord

{

private:

string tip;

public:

string get\_Tip();

GussWordWithTip(string word , string tip);

};

#include "stdafx.h"

#include "GussWordWithTip.h"

string GussWordWithTip::get\_Tip()

{

return tip;

}

GussWordWithTip::GussWordWithTip(string word, string tip):GuessWord(word)

{

this->tip = tip;

}

// ConsoleApplication32.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include <iostream>

#include"GussWordWithTip.h"

using namespace std;

int main()

{

GussWordWithTip\* word = new GussWordWithTip("HELLO","Formal/informal greating");

do

{

char letter;

printf("Tip: %s\n",word->get\_Tip().data());

printf("Your letter :");

cin >> letter;

int guessedCount = word->checkLetter(letter);

if (guessedCount > 0)

{

printf("Guessed letters count =%d\n", guessedCount);

}

else

printf("There are no such letters in the word! \n");

word->pringField();

} while (word->isNotGuessedLetter());

printf("Finally done!");

system("pause");

return 0;

}