Занятие 10

12.04.2021

* Задание на две недели. Кроме интерфейса, здесь есть возможность продемонстрировать свои знания по объектно-ориентированному программированию.

Test вопрос ответ тестирующая программа

* Что вы понимаете под «Приложение на базе диалога»?

**WinAPI: разработка приложений с графическим интерфейсом**

Разработка тестирующей программы

***Требования к интерфейсу***

Приложение на базе диалога.

Тип области знаний для вопросов – любая. Необходимые данные считываются из файла.

Вопросы двух типов:

* альтернативный выбор, на базе Radio;
* множественный выбор, на базе Check.

Количество вариантов ответов – от 2 до 5.

Обеспечить:

* загрузку файла с вопросами, использовать стандартный диалог;
* листание вопросов: к следующему/к предыдущему;
* сохранение ответов пользователя с возможностью изменить ответ;
* оценка результата по завершению тестирования, отчет сохраняется в текстовом файле.

Question.h

#pragma once

#include<iostream>

#include<vector>

class Question {

public:

Question() = default;

Question(int id, std::string type, std::string question\_text, std::vector<std::string> answer\_options, std::vector<int> correct\_answers)

: type\_(type), answer\_options\_(answer\_options), question\_text\_(question\_text), correct\_answers\_(correct\_answers) { }

~Question() = default;

Question& operator=(const Question& q) {

type\_ = q.type\_;

question\_text\_ = q.question\_text\_;

for (int i = 0; i < answer\_options\_.size(); i++) {

answer\_options\_[i] = q.answer\_options\_[i];

}

for (int i = 0; i < correct\_answers\_.size(); i++) {

correct\_answers\_[i] = q.correct\_answers\_[i];

}

return \*this;

}

std::string GetQuestionText() const { return question\_text\_; }

std::vector<int> GetCorrectAnswers()const { return correct\_answers\_; }

std::string GetType()const { return type\_; }

std::vector<std::string> GetAnswersOptions()const { return answer\_options\_; }

private:

std::string type\_;

std::string question\_text\_;

std::vector<std::string> answer\_options\_;

std::vector<int> correct\_answers\_;

};

Answer.h

#include <iostream>

#include <vector>

class Answer {

public:

Answer() { answers\_.push\_back(0); };

Answer(std::vector<int> answer) :answers\_(answer) {}

~Answer() = default;

void ChangeAnswer(std::vector<int> new\_) { answers\_ = new\_; }

std::vector<int> GetAnswers() { return answers\_; }

private:

std::vector<int> answers\_;

};

TestProgram.h

#include <fstream>

#include <sstream>

#include <algorithm>

#include"Question.h"

#include"Answer.h"

class TestProgram {

public:

TestProgram() = default;

~TestProgram() = default;

void UploadQuestions(std::string file\_name) {

std::ifstream in\_file(file\_name);

if (!in\_file.is\_open()) {

std::cout << "Error file open\n";

return;

};

while (!in\_file.eof()) {

std::string question\_str;

std::getline(in\_file, question\_str,'\n');

std::stringstream in(question\_str);

int question\_id=1;

std::string question\_text;

std::getline(in, question\_text, ';');

std::string questions;

std::getline(in,questions,';');

std::stringstream qs(questions);

std::vector<std::string> answer\_options;

std::string answer\_option;

while (std::getline(qs, answer\_option, ',')) {

answer\_options.push\_back(answer\_option);

}

int correct\_answer;

std::vector<int> correct\_answers;

while (!in.eof()) {

in >> correct\_answer;

correct\_answers.push\_back(correct\_answer);

}

if (correct\_answers.size() == 1) {

Question question (question\_id, "radio", question\_text, answer\_options, correct\_answers);

questions\_.push\_back(question);

Answer answer;

answers\_.push\_back(answer);

}

else {

Question question(question\_id, "check", question\_text, answer\_options, correct\_answers);

questions\_.push\_back(question);

Answer answer;

answers\_.push\_back(answer);

}

question\_id++;

}

}

void ChangeAnswer(int question\_id, std::vector<int> new\_answer) {

answers\_[question\_id - 1].ChangeAnswer(new\_answer);

}

std::vector <bool> GetResult() {

std::vector<bool> result; //элемент, имеющий индекс i, отражает првильность ответа на вопрос i

for (int i = 0; i < questions\_.size(); ++i) {

Question question = questions\_[i];

if (question.GetCorrectAnswers() == answers\_[i].GetAnswers()) {

result.push\_back(true);

}

else {

result.push\_back(false);

}

}

return result;

};

std::vector<Question> GetQuestions()const { return questions\_; }

std::vector<Answer> GetAnswers()const { return answers\_; }

private:

std::vector<Question> questions\_;//индекс Question в векторе соответствует номеру вопроса

std::vector<Answer> answers\_;// индекс Answer в векторе соответсвует номеру вопроса, которому принадлежит данный ответ

};

Source.cpp

#include<windows.h>

#include<string>

#include<cstdio>

#include"TestProgram.h"

#include"resource.h"

INT\_PTR CALLBACK DlgProc(HWND, UINT, WPARAM, LPARAM);

void ShowOptions(HWND hDlg,Question question) {

if (question.GetType() == "radio") {

for (int i = 0; i < question.GetAnswersOptions().size(); ++i) {

ShowWindow(GetDlgItem(hDlg, IDC\_RADIO1 + i), SW\_SHOWNORMAL);

SetDlgItemText(hDlg, IDC\_RADIO1 + i, question.GetAnswersOptions()[i].c\_str());

}

for (int i = question.GetAnswersOptions().size(); i < 3; i++) {

ShowWindow(GetDlgItem(hDlg, IDC\_RADIO1 + i), SW\_HIDE);

}

}

if (question.GetType() == "check") {

for (int i = 0; i < 5; ++i) {

ShowWindow(GetDlgItem(hDlg, IDC\_CHECK1 + i), SW\_SHOWNORMAL);

SetDlgItemText(hDlg, IDC\_CHECK1 + i, question.GetAnswersOptions()[i].c\_str());

}

for (int i = question.GetAnswersOptions().size(); i < 5; i++) {

ShowWindow(GetDlgItem(hDlg, IDC\_CHECK1 + i), SW\_HIDE);

}

}

}

std::vector<int> ReadAnswers(HWND hDlg, Question question) {

std::vector<int> answers;

int answer\_id=0;// answer\_id = i

if (question.GetType() == "radio") {

for (int i = 0; i < question.GetAnswersOptions().size(); ++i) {

if (IsDlgButtonChecked(hDlg, IDC\_RADIO1 + i)== BST\_CHECKED) {

answer\_id++;

answers.push\_back(answer\_id);

}

}

}

if (question.GetType() == "check") {

for (int i = 0; i < question.GetAnswersOptions().size(); ++i) {

if (IsDlgButtonChecked(hDlg, IDC\_CHECK1 + i) == BST\_CHECKED) {

answer\_id++;

answers.push\_back(answer\_id);

}

}

}

return answers;

}

int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPWSTR lpCmdLine,

\_In\_ int nCmdShow)

{

DialogBox(hInstance, MAKEINTRESOURCE(IDD\_DIALOG1), NULL, DlgProc);

return 0;

}

INT\_PTR CALLBACK DlgProc(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)

{

TestProgram program;

CHAR file\_name[150] = { '\0' };

static int current\_question\_id = 0;

std::vector<Question> questions;

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_INITDIALOG:

for (int i = 0; i < 3; ++i) {

ShowWindow(GetDlgItem(hDlg, IDC\_RADIO1 + i), SW\_HIDE);

}

for (int i = 0; i < 5; ++i) {

ShowWindow(GetDlgItem(hDlg, IDC\_CHECK1 + i), SW\_HIDE);

}

ShowWindow(GetDlgItem(hDlg, IDC\_BUTTON2), SW\_HIDE);

ShowWindow(GetDlgItem(hDlg, IDC\_BUTTON3), SW\_HIDE);

case WM\_COMMAND:

if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)

{

EndDialog(hDlg, LOWORD(wParam));

return (INT\_PTR)TRUE;

}

switch (LOWORD(wParam))

{

case IDC\_BUTTON1://file open

OPENFILENAME ofn;

ZeroMemory(&ofn, sizeof(ofn));

ofn.lStructSize = sizeof(ofn);

ofn.hwndOwner = NULL;

ofn.lpstrFile = file\_name;

ofn.nMaxFile = sizeof(file\_name);

ofn.lpstrFilter = "All Files\0\*.\*\0\0";

ofn.nFilterIndex = 1;

ofn.lpstrFileTitle = NULL;

ofn.nMaxFileTitle = 0;

ofn.Flags = OFN\_FILEMUSTEXIST | OFN\_PATHMUSTEXIST;

if (GetOpenFileName(&ofn)) {

program.UploadQuestions(std::string(file\_name));

ShowWindow(GetDlgItem(hDlg, IDC\_BUTTON2),SW\_SHOWNORMAL);

ShowWindow(GetDlgItem(hDlg, IDC\_BUTTON3), SW\_SHOWNORMAL);

questions = program.GetQuestions();

current\_question\_id++;

Question question = questions[current\_question\_id-1];

SetDlgItemText(hDlg, IDC\_EDIT1, question.GetQuestionText().c\_str());

ShowOptions(hDlg, question);

}

break;

case IDC\_BUTTON3://next

{

Question question = questions[current\_question\_id - 1];

std::vector<int> current\_answers = ReadAnswers(hDlg, question);

program.ChangeAnswer(current\_question\_id, current\_answers);

current\_question\_id++;

question = questions[current\_question\_id - 1];

SetDlgItemText(hDlg, IDC\_EDIT1, question.GetQuestionText().c\_str());

ShowOptions(hDlg, question);

break;

}

case IDC\_BUTTON2://prev

{

Question question = questions[current\_question\_id - 1];

std::vector<int> current\_answers = ReadAnswers(hDlg, question);

program.ChangeAnswer(current\_question\_id, current\_answers);

current\_question\_id--;

question = questions[current\_question\_id - 1];

SetDlgItemText(hDlg, IDC\_EDIT1, question.GetQuestionText().c\_str());

ShowOptions(hDlg, question);

break;

}

case IDC\_BUTTON4:

{

std::vector<bool> result = program.GetResult();

std::ofstream out("Result.txt");

out << "Wrong answers: ";

for (int i = 0; i < result.size(); i++) {

if (result[i] == false) {

out << i+1;

}

}

EndDialog(hDlg, LOWORD(wParam));

}

default:

break;

}

return (INT\_PTR)TRUE;

default:

return (INT\_PTR)FALSE;

break;

}

return (INT\_PTR)TRUE;

}