Politechnika Śląska
Wydział Matematyk Stosowanej
Kierunek Informatyka
Gliwice, 20.06.2024

Programowanie II

projekt zaliczeniowy

"Milionerzy"

Michał Tarnawa gr. lab. 4/7

Wojciech Grzywocz gr. lab. 4/8

MILIONERZY

INSTRUKCJA

Użytkownik ma do wyboru kilka opcji do wpisania w konsoli:

1. GRAJ- rozpoczyna grę
2. DODAJ PYTANIE – dodaje nowe pytanie do bazy
3. USUN PYTANIE – usuwa pytanie z bazy danych, po podaniu ID rekordu

Gameplay:

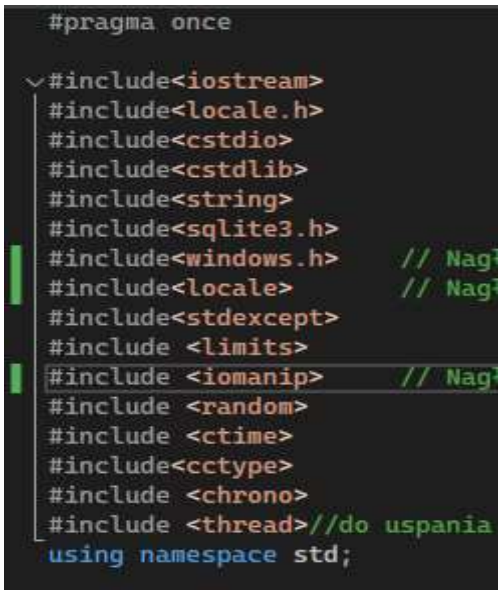
1. Gracz otrzymuje pytanie i musi odpowiedzieć na nie wpisując do konsoli A,B,C lub D. Kiedy gracz wpisze znak niedozwolony do użycia gra zada pytanie na nowo.
2. Dostępne są 3 koła ratunkowe:
    a. Telefon do przyjaciela(po wpisaniu T w konsoli)
        i. Ma 90% szans za zwrócenie poprawnej odpowiedzi
    b. Pytanie do eksperta(E)
        i. Wyklucza 2 błędne odpowiedzi
    c. Pytanie do publiczności(P)
        i. Pokazuje jak ”głosowała” publiczność. Poprawna odpowiedź jest faworyzowana
3. Istnieje możliwość wycofania się z gry po wpisaniu K.
    a. Zgarniamy nagrodę za poprzednio odpowiedziane pytanie.
4. Jeśli odpowiedź jest zła, to gra się kończy i otrzymujemy pieniądze z progu gwarantowanego.

DZIAŁANIE PROGRAMU

Program składa się z kilku programów:

- Header.h - plik nagłówkowy, zawiera
- Dane.cpp - zawiera definicje zmiennych wykorzystywanych w innych plikach
- DB CREATION.cpp – odpowiada za stworzenie bazy danych za pomocą SQLite
- Funckje.cpp - zawiera funkcje, których nie udało się posegregować w innych plikach
- funkcje gry.cpp - zawiera funkcje, które są wykorzystywane podczas gry.
- InsertQuestion.cpp - odpowiada za dodanie pytania w bazie danych
- kola.cpp- zawiera ciała funkcji związanymi z kołami ratunkowymi
- Losowanie.cpp - odpowiada za losowanie pytań
- Read_Question.cpp- odczytuje pytania z bazy danych
- stawki,cpp – zawiera funkcję wypisującą progi gwarantowane
- main.cpp - odpowiada za działanie całego programu

1. Header.cpp
    1.1.Zawiera wszystkie potrzebne nagłówki



```
#pragma once

#include<iostream>
#include<locale.h>
#include<cstdio>
#include<cstdlib>
#include<string>
#include<sqlite3.h>
#include<windows.h>     // Nag
#include<locale>        // Nag
#include<stdexcept>
#include <limits>
#include <iomanip>      // Nag
#include <random>
#include <ctime>
#include<cctype>
#include <chrono>
#include <thread>//do uspania
using namespace std;
```

    1.2.

    Deklarację klasy Questions, która zawiera atrybuty dotyczące treści pytania, odpowiedzi oraz poprawne odpowiedzi. Oprócz tego zawiera więzy przyjaźni do innych funkcji oraz definicje przyjaźni

```
class Questions
{
public:
    string question;
    string answer1;
    string answer2;
    string answer3;
    string answer4;
    string correctAnswer;

    Questions(){}
    Questions(const string& q, const string& a1, const string& a2, const string& a3, const string& a4, const string& ca)
    {
        question = q;
        answer1 = a1;
        answer2 = a2;
        answer3 = a3;
        answer4 = a4;
        correctAnswer = ca;
    }
    friend Questions readQuestionByID(sqlite3* db, int id, Questions& row);
    friend void poprawnosc(Questions tab[]);
    void wyswietl(int &);
    void przyjaciel(bool warunek);
    void ekspert(bool warunek);
    void pub(bool warunek);

    virtual ~Questions(){}
};
```

    Definicje funkcji oraz wsztstkich zmiennych
        1.2.1. extern int maks;
        1.2.2. extern int numer_pytania;
        1.2.3. extern int poprawne_odp;
        1.2.4. extern Questions pomocnicze;
        1.2.5. extern char odp;
        1.2.6. extern string ans;
        1.2.7. extern bool war1;

1.2.8. extern bool kolo1;
1.2.9. extern bool kolo2;
1.2.10. extern bool kolo3;
1.2.11. string litera(int a);
1.2.12. extern WORD backgroundColor;
1.2.13. extern WORD textColor;
1.2.14. void createTable(sqlite3* db);
1.2.15. void check(int result, sqlite3* db);
1.2.16. void InsertQuestion(sqlite3* db);
1.2.17. int countQuestions(sqlite3* db);
1.2.18. void drawing(int n, bool logic[], int wylosowane[]);
1.2.19. void stawki();
1.2.20. void kola(bool kolo1, bool kolo2, bool kolo3);
1.2.21. void przeliczanie();
1.2.22. void wygrana(bool war1, int b);
1.2.23. void beginning();
1.2.24. void DeleteQuestion(sqlite3* db, int id);

2. DB creation.cpp
   2.1. Funckja void createTable(sqlite3 *db) tworzy tabelę jeśli ona nie istnieje

```
void createTable(sqlite3* db)
{
    const char* sqlCreateTable =                    //tworzenie tabeli w bazie
        "CREATE TABLE IF NOT EXISTS PYTANIA ("
        "ID INTEGER PRIMARY KEY AUTOINCREMENT, "
        "QUESTION TEXT NOT NULL, "
        "ANSWER1 TEXT NOT NULL, "
        "ANSWER2 TEXT NOT NULL, "
        "ANSWER3 TEXT NOT NULL, "
        "ANSWER4 TEXT NOT NULL, "
        "CORRECT_ANSWER TEXT NOT NULL);";

    int creation_result = sqlite3_exec(db, sqlCreateTable, nullptr, nullptr, nullptr);        //wykonanie zapytania
    check(creation_result, db);                                                  //sprawdzenie czy wykonano poprawnie
}
```

   Tabela ta zawiera pola takie jak ID(numer pytania), Pole Question(pytanie), answer1/2/3/4 jako odpowiedzi oraz CORRECT_ANSWER jako poprawna odpowiedź

   2.2. Funkcja void check(int result, sqlite3*db) sprawdza czy nie doszło do żadnej stuacji wyjątkowej. Jeśli tak to zwraca błąd.

```
void check(int result, sqlite3* db)          //Funkcja sprawdzajaca poprawnosc działania zapytania do bazy danych
{
    if (result != SQLITE_OK)
    {
        cerr << "Wystąpił błąd " << sqlite3_errmsg(db) << endl;
        sqlite3_close(db);
        exit(1);
    }
}
```

3. Insert_Question.cpp

3.1.- dodaje pytanie do bazy danych

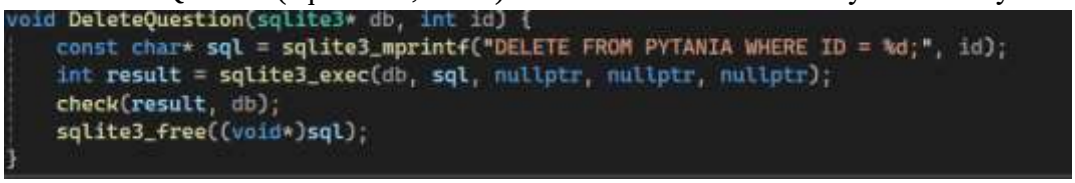
```
void InsertQuestion(sqlite3* db)
{
    int number_of_q;
    cout << "Ile pytan chcesz dodac? " << endl;
    cin >> number_of_q;
    for (int i = 1; i <= number_of_q; i++)
    {
        string k;
        string question, answer1, answer2, answer3, answer4, correctAnswer;
        cout << "Podaj pytanie: " << endl;
        cin.ignore();
        getline(cin,question);
        cout << "Podaj odpowiedz a: " << endl;
        getline(cin,answer1);
        cout << "Podaj odpowiedz b: " << endl;
        getline(cin,answer2);
        cout << "Podaj odpowiedz c: " << endl;
        getline(cin,answer3);
        cout << "Podaj odpowiedz d: " << endl;
        getline(cin,answer4);
        while (k != "A" && k != "B" && k != "C" && k != "D") {
            cout << "Podaj poprawną odpowiedz: ";
            getline(cin, k);
        }
        correctAnswer = k;

        const char* sql = sqlite3_mprintf("INSERT INTO PYTANIA (QUESTION, ANSWER1, ANSWER2, ANSWER3, ANSWER4, CORRECT_ANSWER)" "VAL
        int result = sqlite3_exec(db, sql, nullptr, nullptr, nullptr);
        check(result, db);
    }

}
```

Można dodać więcej pytań. Check sprawdza czy poprawnie dodano pytania. Sprawdza czy poprawna odpowiedź to A/B/C/D, jeśli nie to ponawia zapytanie

3.1.1. Linijka która się nie zmieściła

3.1.2. const char* sql = sqlite3_mprintf("INSERT INTO PYTANIA (QUESTION, ANSWER1, ANSWER2, ANSWER3, ANSWER4, CORRECT_ANSWER)" "VALUES ('%q', '%q', '%q', '%q', '%q', '%q');", question.c_str(), answer1.c_str(), answer2.c_str(), answer3.c_str(), answer4.c_str(), correctAnswer.c_str());

3.2. Void DeleteQuestion(sqlite* db, int id) - usuwa rekord o określonym id z bazy



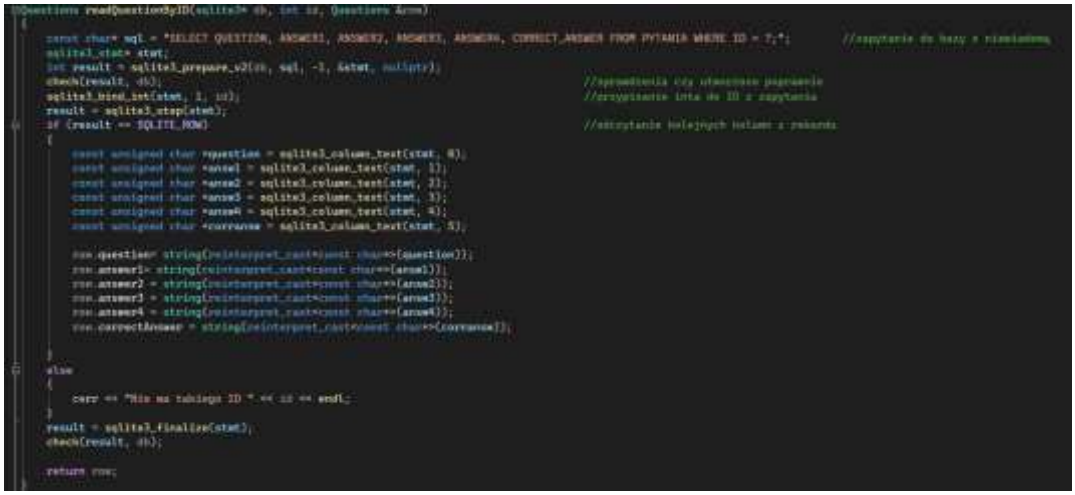
```
void DeleteQuestion(sqlite3* db, int id) {
    const char* sql = sqlite3_mprintf("DELETE FROM PYTANIA WHERE ID = %d;", id);
    int result = sqlite3_exec(db, sql, nullptr, nullptr, nullptr);
    check(result, db);
    sqlite3_free((void*)sql);
}
```

check sprawdza poprawność

4. Read_Question.cpp

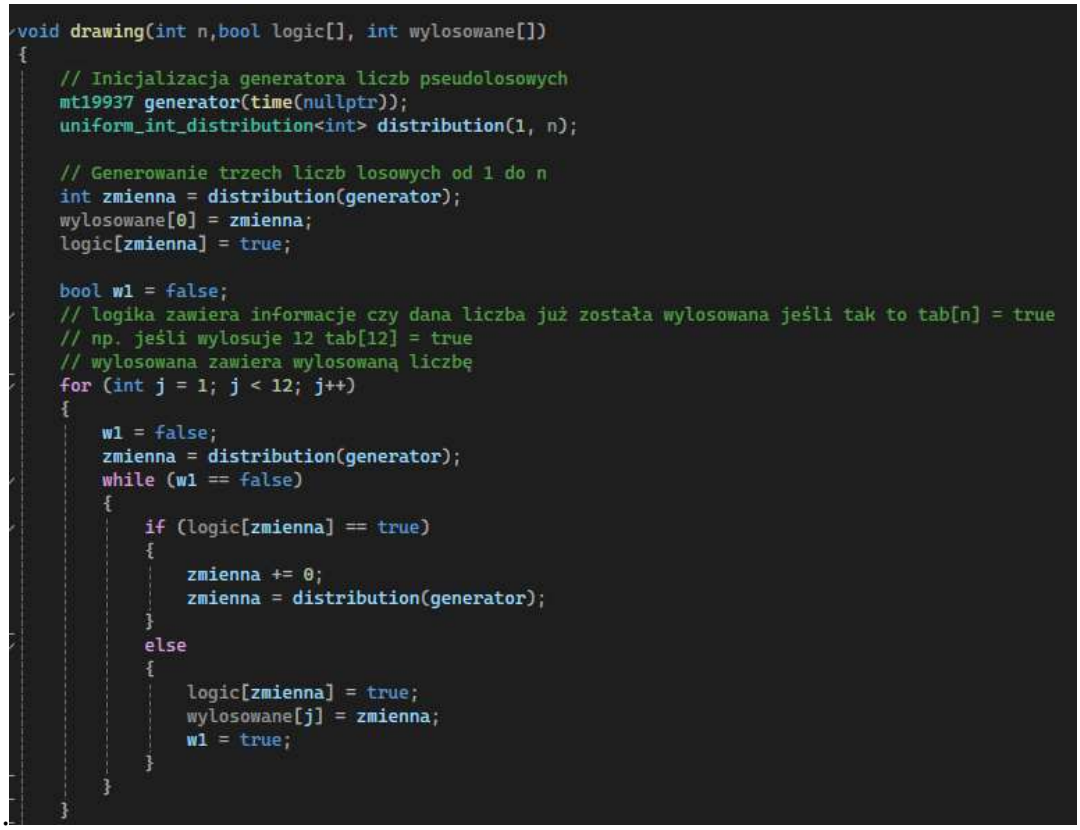
4.1. Jest to funkcja zaprzyjaźniona z klasa Questions. Odczytuje ona rekord o określonym ID, a następnie przypisuje informacje zawarte w tym rekordzie do tożsamych

atrybutów obiektu



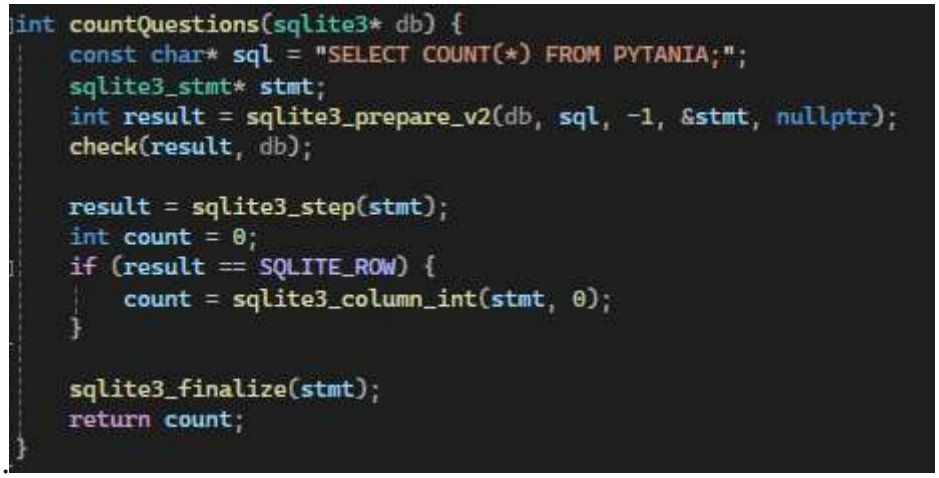
5. Losowanie.cpp

5.1.



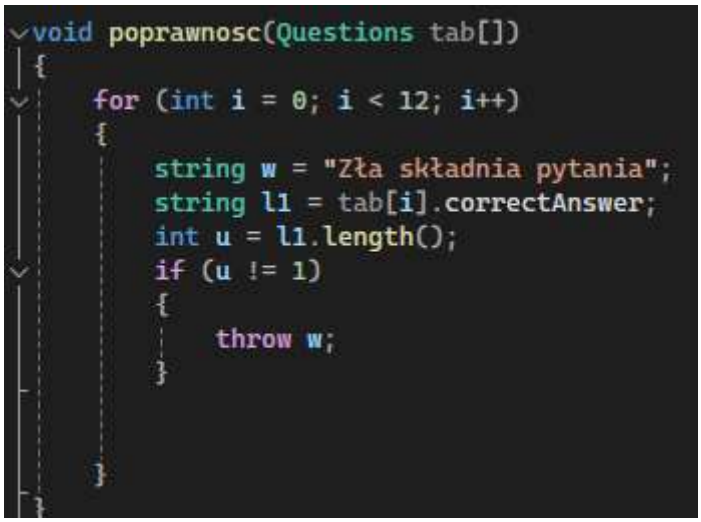
Funkcja ta na podstawie ilości pytań(n) losuje kolejność pytań i zapisuje wylosowane liczby do tabeli losowe. Tabela bool logic zawiera informacje czy dana liczba została już wylosowana. Jeśli tak to zmienia wartość na true. Np. Jeśli wylosuje 6, to logic[6] = True. Jeśli jakiś indeks na już wartość true, ponownie losuje nową liczbę do skutku.

6. Funckje.cpp

6.1.


Funkcja countQuestions ma za zadanie zliczenie ile jest pytań w bazie danych. Check(results,db) sprawdza poprawność odczytania. Funkcja zwraca int count, czyli ilość pytań.

6.2. Sprawdza czy długość pola odpowiedzi wynosi 1. Jeśli tak co nic nie zwraca. Jeśli nie to wykorzysta polecenie throw jako obsługę wyjątku



6.3. Funkcja ta jest wykorzystywane do zamiany odpowiedniej liczby na odpowiednią literę.

```
string litera(int a) // zamienia cyfrę od 1 do 4 na A DO D
{
    string w;
    if (a == 1) {
        w = "A";
    }
    else if (a == 2) {
        w = "B";
    }
    else if (a == 3) {
        w = "C";
    }
    else if (a == 4) {
        w = "D";
    }
    return w;

}
```

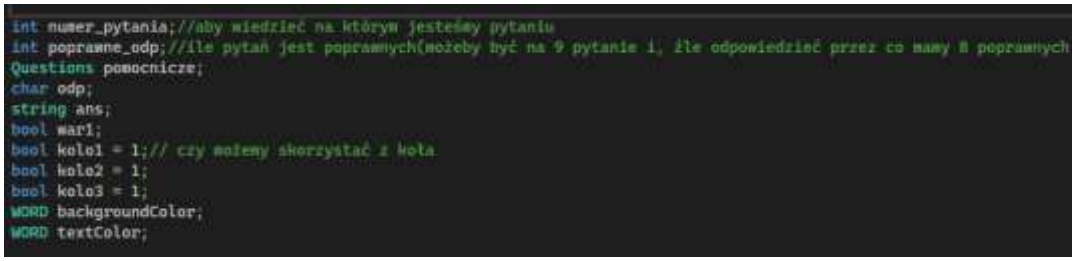
6.4. Dwie funkcje służące odpowiednio do ustalenia rozmiaru konsoli i później na podstawie tej wartości tworzony jest wyśrodkowany wypis w konsoli

```
int getConsoleWidth()
{
    CONSOLE_SCREEN_BUFFER_INFO csbi;
    int columns;
    GetConsoleScreenBufferInfo(GetStdHandle(STD_OUTPUT_HANDLE), &csbi);
    columns = csbi.srWindow.Right - csbi.srWindow.Left + 1;
    return columns;
}

void coutCentered(const string& text)
{
    int consoleWidth = getConsoleWidth();
    int textLength = text.length();
    int spaces = (consoleWidth - textLength) / 2;
    for (int i = 0; i < spaces; ++i)
    {
        cout << " ";
    }
    cout << text << endl;

}
```

6.5. Funkcja służąca do zmiany koloru i tła w konsoli

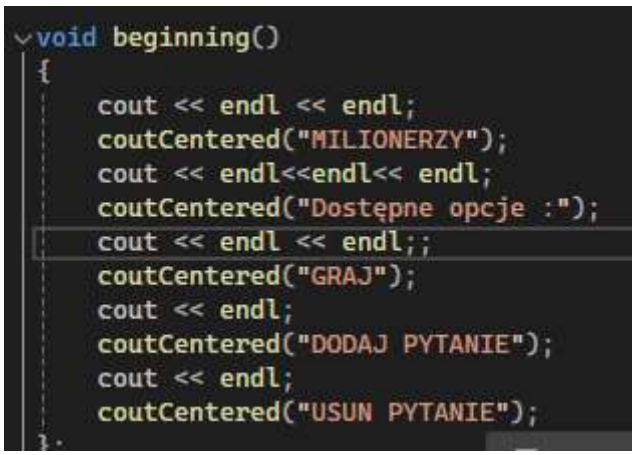
```
void setConsoleColor(WORD attributes)
{
    HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(hConsole, attributes);
}
```

7. Dane.cpp
    7.1. Zawiera deklaracje wszystkich potrzebnych zmiennych



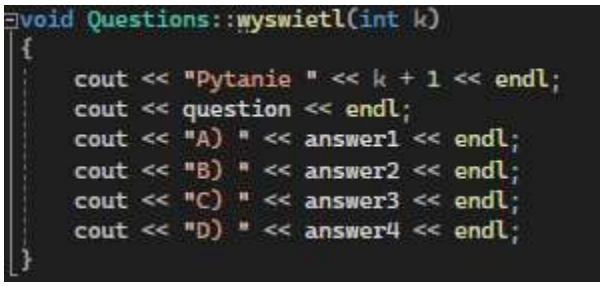
```
int numer_pytania;//aby wiedzieć na którym jesteśmy pytaniu
int poprawne_odp;//ile pytań jest poprawnych(możeby być na 9 pytanie 1, źle odpowiedzieć przez co mamy 8 poprawnych
Questions pomocnicze;
char odp;
string ans;
bool war1;
bool kolo1 = 1;// czy możemy skorzystać z koła
bool kolo2 = 1;
bool kolo3 = 1;
WORD backgroundColor;
WORD textColor;
```

8. funkcje gry.cpp
    8.1. Rozpoczyna grę.



```
void beginning()
{
    cout << endl << endl;
    coutCentered("MILIONERZY");
    cout << endl<<endl<< endl;
    coutCentered("Dostępne opcje :");
    cout << endl << endl;;
    coutCentered("GRAJ");
    cout << endl;
    coutCentered("DODAJ PYTANIE");
    cout << endl;
    coutCentered("USUN PYTANIE");
}
```

    8.2. Jest to metoda z ciała klasy. Wypisuje pytanie oraz odpowiedzi. Wywoływana za pomocą: obiekt.wyswietl(numer_pytania)



```
void Questions::wyswietl(int k)
{
    cout << "Pytanie " << k + 1 << endl;
    cout << question << endl;
    cout << "A) " << answer1 << endl;
    cout << "B) " << answer2 << endl;
    cout << "C) " << answer3 << endl;
    cout << "D) " << answer4 << endl;
}
```

    8.3. Wykorzystuje blblioteki thread oraz chrono w celu odliczania, podczas oczekiwania na wynik(ma za zadanie budować napięcie).

```
void przeliczanie()
{
    for (int i = 0; i < 5; i++)
    {
        backgroundColor = BACKGROUND_BLUE;
        textColor = FOREGROUND_RED | FOREGROUND_GREEN | FOREGROUND_BLUE;
        setConsoleColor(backgroundColor | textColor);
        system("cls");

        cout << "Przeliczanie wyniku proszę czekać...." << 5 - i<< endl;
        this_thread::sleep_for(chrono::seconds(1));
        system("cls");
        backgroundColor = BACKGROUND_BLUE;
        setConsoleColor(backgroundColor | textColor);
    }

}
```

8.4. Funcka void wygrana(bool war1, int b) zwraca informacje o tym ile wygraliśmy. War1 jest warunkiem, który sprawdza czy wycowaliśmysię z gry(true), czy przegraliśmy(false). B to ilość poprawnych pytań.

```
void wygrana(bool war1, int b)// Podaje nagrodę
{
    if (war1)// jesli sie wycofaliśmy lub wygraliśmy
    {
        switch (b)
        {
        case 12:
            cout << "Gratulacje! Wygrałeś 1.000.000!" << endl;
            break;
        case 11:
            cout << "Wygrałeś 500.000!" << endl;
            break;
        case 10:
            cout << "Wygrałeś 250.000!" << endl;
            break;
        case 9:
            cout << "Wygrałeś 125.000!" << endl;
            break;
        case 8:
            cout << "Wygrałeś 75.000!" << endl;
            break;
        case 7:
            cout << "Wygrałeś 40.000!" << endl;
            break;
        case 6:
            cout << "Wygrałeś 20.000!" << endl;
            break;
        case 5:
            cout << "Wygrałeś 10.000!" << endl;
            break;
        case 4:
            cout << "Wygrałeś 5.000!" << endl;
            break;
        case 3:
            cout << "Wygrałeś 2.000!" << endl;
            break;
        case 2:
            cout << "Wygrałeś 1.000!" << endl;
            break;
        case 1:
            cout << "Wygrałeś 500!" << endl;
            break;
        default:
            cout << "Niestety, nie wygrałeś żadnej nagrody." << endl;
            break;
        }
    }
    else { // jeśli źle odpowiedzieliśmy
        if ((b) >= 2 && b < 7)
        {
            cout << "Wygrałeś 1.000!" << endl;

        }
        if ((b) >= 7 && b < 11)
        {
            cout << "Wygrałeś 40.000!" << endl;

        }
        if (b < 2)
        {
            cout << "Niestety nic nie wygrałeś, nie martw się, dasz radę następnym razem";
        }
    }
}
```

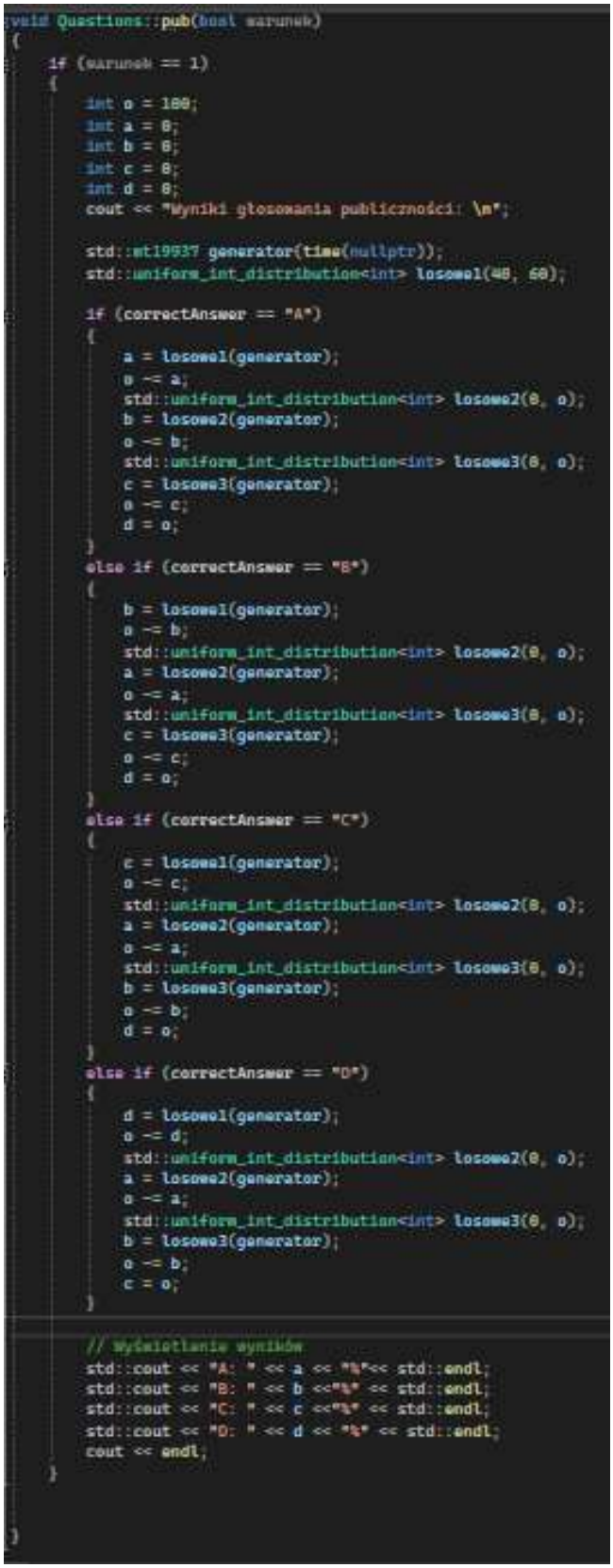
9. Kola.cpp

9.1. Metoda klasy void ekspert(bool warunek) zwraca informację jakie dwa pytania z czterech są niepoprawne

```
void Questions::ekspert(bool warunek)
{
    if (warunek == 1)
    {
        mt19937 generator(time(nullptr));
        uniform_int_distribution<int> losowe(1, 4);
        int a = losowe(generator);
        int b = losowe(generator);
        // w tej pętli sprawdzimy, czy odrzucane odpowiedzi nie są poprawną i czy są różne od siebie
        while (litera(a) == correctAnswer || litera(b) == correctAnswer || litera(b) == litera(a))
        {
            //cout << '1';
            a = losowe(generator);
            b = losowe(generator);
        }

        cout << "Po przeanalizowaniu pytania należy wykluczyć odpowiedź ";
        backgroundColor = BACKGROUND_BLUE;
        textColor = FOREGROUND_RED;
        setConsoleColor(backgroundColor | textColor);
        cout << litera(a);
        textColor = FOREGROUND_RED | FOREGROUND_GREEN | FOREGROUND_BLUE;
        setConsoleColor(backgroundColor | textColor);
        cout << " i ";
        textColor = FOREGROUND_RED;
        setConsoleColor(backgroundColor | textColor);
        cout << litera(b);
        textColor = FOREGROUND_RED | FOREGROUND_GREEN | FOREGROUND_BLUE;
        setConsoleColor(backgroundColor | textColor);
        cout << endl;
    }
}
```

Gdzie bool warunek informuje nas czy dane koło jest dostępne(true) czy nie. Pytanie do wykluczenia są losowane i czy są inne niż poprawne czy różne od siebie.

9.2. Metoda void pub(int warunek) pokazuje jakie pytanie wybrała publiczność.



```
void Questions::pub(bool warunek)
{
    if (warunek == 1)
    {
        int o = 100;
        int a = 0;
        int b = 0;
        int c = 0;
        int d = 0;
        cout << "Wyniki głosowania publiczności: \n";

        std::mt19937 generator(time(nullptr));
        std::uniform_int_distribution<int> losowa1(40, 60);

        if (correctAnswer == "A")
        {
            a = losowa1(generator);
            o -= a;
            std::uniform_int_distribution<int> losowa2(0, o);
            b = losowa2(generator);
            o -= b;
            std::uniform_int_distribution<int> losowa3(0, o);
            c = losowa3(generator);
            o -= c;
            d = o;
        }
        else if (correctAnswer == "B")
        {
            b = losowa1(generator);
            o -= b;
            std::uniform_int_distribution<int> losowa2(0, o);
            a = losowa2(generator);
            o -= a;
            std::uniform_int_distribution<int> losowa3(0, o);
            c = losowa3(generator);
            o -= c;
            d = o;
        }
        else if (correctAnswer == "C")
        {
            c = losowa1(generator);
            o -= c;
            std::uniform_int_distribution<int> losowa2(0, o);
            a = losowa2(generator);
            o -= a;
            std::uniform_int_distribution<int> losowa3(0, o);
            b = losowa3(generator);
            o -= b;
            d = o;
        }
        else if (correctAnswer == "D")
        {
            d = losowa1(generator);
            o -= d;
            std::uniform_int_distribution<int> losowa2(0, o);
            a = losowa2(generator);
            o -= a;
            std::uniform_int_distribution<int> losowa3(0, o);
            b = losowa3(generator);
            o -= b;
            c = o;
        }

        // Wyświetlanie wyników
        std::cout << "A: " << a << "%"<< std::endl;
        std::cout << "B: " << b <<"%" << std::endl;
        std::cout << "C: " << c <<"%" << std::endl;
        std::cout << "D: " << d << "%" << std::endl;
        cout << endl;
    }


}
```

Dla poprawnej odpowiedzi przyjmowana jest wartość od 40 do 60. Następnie dla kolejne odpowiedzi losuje się wartość od 0 do o, gdzie o wynosi 100 – poprzednia wartość. Dla trzeciej opcji losuje się od 0 do o, gdzie o = 0 – poprzednia wartość itd. Potem wyświetlane są wszelkie wartości. Metoda ta faworyzje poprawną odpowiedź

9.3. Metoda void przyjaciel(bool warunek) ma 90% szans za zwrot poprawnej odpowiedzi. Prawdopodobieństwo jest pseudolosowe. Gdzie bool warunek informuje nas czy dane koło jest dostępne(true) czy nie.

```cpp
void Questions::przyjaciel(bool warunek)
{
    if (warunek == 1)// Czy nadal możemy skorzystać z koła
    {
        mt19937 generator(time(nullptr));
        uniform_int_distribution<int> distribution(1, 100); //szansa na to, że przyjaciel zna odpowiedź
        uniform_int_distribution<int> unlucky(1, 4); // jeśli nie to losujemy inną odpowiedź
        int chance = distribution(generator);
        if (chance <= 90) // mamy 90% szans na to, że otrzymamy poprawną odpowiedź
        {
            cout << " Słuchaj stary wydaje mi się, że poprawna odpowiedź to: " << correctAnswer << endl;
        }
        else
        {
            int pech = unlucky(generator);
            while (litera(pech) == correctAnswer)
            {
                int pech = unlucky(generator);
            }
            cout << " Słuchaj stary wydaje mi się, że poprawna odpowiedź to: " << litera(pech) << endl;
        }

    }
}
```

10. Stawki.cpp

Void stawki() wypisuje stawki oraz zaznacza na czerwono progi gwarantowane

```cpp
void stawki() // wypisuje stawki wygranej
{

    cout << " Tak oto wyglądają nagrody za poszczególne pytania" << endl;
    cout << setw(2) << "12" << setw(15) << fixed << setprecision(2) << "1.000.000" << endl;
    cout << setw(2) << "11" << setw(15) << fixed << setprecision(2) << "500.000" << endl;
    cout << setw(2) << "10" << setw(15) << fixed << setprecision(2) << "250.000" << endl;
    cout << setw(2) << "9" << setw(15) << fixed << setprecision(2) << "125.000" << endl;
    cout << setw(2) << "8" << setw(15) << fixed << setprecision(2) << "75.000" << endl;
    backgroundColor = BACKGROUND_BLUE;
    textColor = FOREGROUND_RED;
    setConsoleColor(backgroundColor | textColor);
    cout << setw(2) << "7" << setw(15) << fixed << setprecision(2) << "40.000" << endl;
    textColor = FOREGROUND_RED | FOREGROUND_GREEN | FOREGROUND_BLUE;
    setConsoleColor(backgroundColor | textColor);
    cout << setw(2) << "6" << setw(15) << fixed << setprecision(2) << "20.000" << endl;
    cout << setw(2) << "5" << setw(15) << fixed << setprecision(2) << "10.000" << endl;
    cout << setw(2) << "4" << setw(15) << fixed << setprecision(2) << "5.000" << endl;
    cout << setw(2) << "3" << setw(15) << fixed << setprecision(2) << "2.000" << endl;
    textColor = FOREGROUND_RED;
    setConsoleColor(backgroundColor | textColor);
    cout << setw(2) << "2" << setw(15) << fixed << setprecision(2) << "1.000" << endl;
    textColor = FOREGROUND_RED | FOREGROUND_GREEN | FOREGROUND_BLUE;
    setConsoleColor(backgroundColor | textColor);
    cout << setw(2) << "1" << setw(15) << fixed << setprecision(2) << "500" << endl;
    cout << endl << "Na ";
    textColor = FOREGROUND_RED;
    setConsoleColor(backgroundColor | textColor);
    cout << "czerwono ";
    textColor = FOREGROUND_RED | FOREGROUND_GREEN | FOREGROUND_BLUE;
    setConsoleColor(backgroundColor | textColor);
    cout << "zaznaczono progi gwarantowane [ENTER]";

}
```

11. Main.cpp

11.1. Ostatni i najważniejszy plik, to on składa wszystko, tak aby działało.

11.2. Na początku ustawia parametry wstępne konsoli takie jak np kolor tła i tekstu, a następnie otwiera bazę danych i zwraca informację jeśli to się nie uda(check)

```
int main()
{


    SetConsoleCP(CP_UTF8);
    SetConsoleOutputCP(CP_UTF8);
    setlocale(LC_ALL, "pl_PL");  // ustawienie polskich znaków
    backgroundColor = BACKGROUND_BLUE;
    textColor = FOREGROUND_RED | FOREGROUND_GREEN | FOREGROUND_BLUE;
    setConsoleColor(backgroundColor | textColor);
    sqlite3* db;
    int result;



    result = sqlite3_open("Milionerzy-pytania Baza danych", &db); // otwieranie bazy danych lub utworzenie
    check(result, db);
    sqlite3_exec(db, "PRAGMA encoding = 'UTF-8';", nullptr, nullptr, nullptr);
    cout << "Połączono z bazą danych pytań!" << endl;

    createTable(db); // tworzenie tabeli w bazie danych jeśli nie istnieje
```

11.3. Następnie otwiera się menu główne oraz kilka opcji

```
while (ans != "GRAJ" && ans!= "graj")
{
    system("cls");
    cout << "Połączono z bazą danych pytań!" << endl;
    beginning();
    getline(cin, ans);
    if (ans == "DODAJ PYTANIE" || ans== "dodaj pytanie")
    {
        InsertQuestion(db);
        cout << "Dodano pytanie/a " << endl;
    }
    if (ans == "USUN PYTANIE" || ans== "usun pytanie")
    {
        int id = 0;
        cout << "Podaj numer rekordu do usuniecia";
        cin >> id;
        DeleteQuestion(db, id);
        cout << "Usunięto pytanie/a " << endl;
    }
}
system("cls");
```

11.4. Kolejnym krokiem jest warunek_początkowy - czy na pewno jest przynajmniej 12 pytań w bazie danych. Jeśli tak gra się zaczyna getline(cin,ans) pozwala na przejście dalej jeśli gracz naciśnie ENTER

```
system("cls");
int war_początkowy = countQuestions(db);
if (war_początkowy >= 12)
{
    cout << "Witaj w grze Milionerzy" << endl
        << "Przed tobą 12 pytań, i 4 odpowiedzi dla każdego, lecz tylko jedna odpowiedź jest poprawna" << endl
        << "Naciśnij ENTER by kontynuować";


    getline(cin, ans); // Ignoruj wszystko do nowej linii
    system("cls");
```

11.5. Następnie losowane są pytania

```
bool* logic = new bool[war_początkowy+1];
for (int i = 0; i < war_początkowy+1; ++i) {
    logic[i] = false;
}

int wylosowane[12]{};

drawing(war_początkowy,logic,wylosowane);
```

A wartość wylosowanych pytań zapisywana jest do tablicy wylosowane.

11.6. Następna część main'a tworzy 12 obiektów typu Questions oraz tablicę z tymi 12 pytaniami.

```
Questions p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12;

readQuestionByID(db, wylosowane[0], p1);
readQuestionByID(db, wylosowane[1], p2);
readQuestionByID(db, wylosowane[2], p3);
readQuestionByID(db, wylosowane[3], p4);
readQuestionByID(db, wylosowane[4], p5);
readQuestionByID(db, wylosowane[5], p6);
readQuestionByID(db, wylosowane[6], p7);
readQuestionByID(db, wylosowane[7], p8);
readQuestionByID(db, wylosowane[8], p9);
readQuestionByID(db, wylosowane[9], p10);
readQuestionByID(db, wylosowane[10], p11);
readQuestionByID(db, wylosowane[11], p12);

Questions tab[12] = { p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12 };
string Exception;
```

11.7. Następnie sprawdzana jest poprawność bazy danych. Czy pole z poprawną odpowiedzią na pewno ma długość 1.

```
string Exception;

try {
    poprawnosc(tab);
}
catch (string w)
{
    cout << "Niepoprawna składnia odpowiedzi na pytanie w bazie danych!!! ";
    this_thread::sleep_for(chrono::seconds(15));
    numer_pytania = 234;
}
```

Jeśli nie zwróci wyjątek i uniemożliwi dalsze wykonywanie programu

11.8. W dalszej części rozpoczyna się gra.

```
while (numer_pytania < 12)
{

    //cout << ans;

    tab[numer_pytania].wyswietl(numer_pytania);
    kola(kolo1, kolo2,kolo3);
    getline(cin, ans);
```

11.8.1. Polecenie wyświetl realizuje kod w swoich ciele, a funckja kola() wyświetla ile mamy kół ratunkowych.

11.9. Użytkownik podaje odpowiedź:

```
system("cls");
ans = ans[0];
char ansAlternative = ans[0];
ansAlternative = toupper(ansAlternative);
char correctAlternative = tab[numer_pytania].correctAnswer[0];

if (tab[numer_pytania].correctAnswer == ans || correctAlternative == ansAlternative)
{
    numer_pytania += 1;
    poprawne_odp += 1;
    cout << "To poprawna odpowiedź" << endl;
    cout << "Przechodzimy do kolejnej rundy" << endl;
    //this_thread::sleep_for(chrono::seconds(5));
    system("cls");
    war1 = 1;
}
```

A program akceptuje 1 podany znak. Jeśli jest poprawny to przechodzimy dalej.

11.10. 

```
else if (ans != "A" && ans != "B" && ans != "C" && ans != "D" && ans != "N" && ans != "P" && ans != "E" && ans != "T")
{
    system("cls");
}
```

Warunek ten sprawdza czy podana przez nas odpowiedź jest poza zakresem akceptowalnych odpowiedzi. Jeśli tak to ponownie wyświetli pytanie

11.11. Pozostałe warunki kończą grę lub aktywują koła ratunkowe

```
}
else if (ans == "K" || ans=="k")// zakończenie gry
{
    cout << "Dziękuję za grę, zgarniasz kwotę dla poprzedniego pytania" << endl;
    this_thread::sleep_for(chrono::seconds(5));
    war1 = 1;
    break;
}
else if (ans == "P" || ans=="p") { //publiczność

    system("cls");
    tab[numer_pytania].pub(kolo3);
    kolo3 = 0;


}
else if (ans == "T" || ans=="t") {//telefon do przyjaciela

    system("cls");
    tab[numer_pytania].przyjaciel(kolo1);
    kolo1 = 0;
```

11.12. Jeśli źle odpowiemy gra się zakończy:

```
    else {
        backgroundColor = BACKGROUND_BLUE;
        textColor = FOREGROUND_RED;
        setConsoleColor(backgroundColor | textColor);
        cout << endl << "To niestety zła odpowiedź. Poprawna to: " << tab[numer_pytania].correctAnswer << endl;
        setConsoleColor(FOREGROUND_RED|FOREGROUND_GREEN | FOREGROUND_BLUE);
        war1 = 0;// war1 =0 oznacza, że źle odpowiedziałeś, więc funkcja wygrana zwróci(do nam próg gwarantowany, a nie z ostatniego pytania)
        this_thread::sleep_for(chrono::seconds(3));
        break;
    }
}


przeliczanie();
wygrana(war1, poprawne_odp);
```

.przeliczanie() odlicza do podania odpowiedzi. A wygrana zwraca wygraną przez nas sume.

11.13. Jeśli pytań jest za mało wyświetli się komunikat:

```
else {
    cout << "Za mało pytań w bazie. Sprawdź pliki źródłowe gry";
}
```

11.14. Pod koniec zamykana jest baza danych.

CIEKAKAWOSTKI

1. Podczas tworzenia pytania do przyjaciela kiedy testowane było działanie koła te dwa razy podrząd zwróciło błędną odpowiedź. Na początku myśleliśmy, że to błąd, ale wszystko działało zgodnie z kodem. Prawdopodobieństwo takiego zajścia wynosi 1%
2. Mimo, iż projekt wydaje się łatwy, to tak na prawdę jest skomplikowane. Głownie przez bazę danych oraz skomplikowane mechanizmy kół ratunkowych i uwzględnianie prawdopodobieństwa, biorąc oczywiście pod uwagę pewne ograniczenia które wynikają ze specyfiki działania sprzętu komputerowego.
3. Waga folderu .vs to ponad 1GB

4. Projekt kompiluje się kilka sekund

Wnioski:

Mimo, iż sam gameplay wydaje się prosty, to stworzenie tego projektu zajęło wiele długich godzin (wraz z przygotowaniem).Wszystkie postawione sobie cele udało się zrealizować dużym nakładem pracy i kilkoma nieprzespanymi nocami. Udało się zrealizować najważniejsze i najtrudniejsze(ok 40% czasu projektu) podpięcie bazy danych oraz jej poprawne funkcjonowanie. Dzięki temu projektowi zrozumieliśmy ile czasu zajmuje tworzenie gier. Nauczyliśmy się podpinać i obsługiwać bazy danych. Poznaliśmy składnię języka SQL oraz wykorzystaliśmy w praktyce wiedzę zdobytą na temat programowania obiektowego.

Dodatkowe informacje

Program można rozbudować miedzy innymi o tzw. Endlessmode – gra trwa do pierwszego błędu

Możliwość wzbogacenia o interfejs graficzny

Można rozbudować program o dodatkowe koła ratunkowe i pytania