Politechnika Śląska w Gliwicach Wydział Automatyki, Elektroniki i Informatyki.

Programowanie Komputerów 4

Biblioteka Gier

Autor	Karolina Kachelska
Prowadzący	mgr inż. Grzegorz Kwiatkowski
Kierunek	informatyka
Rodzaj studiów	SSI
Semestr	4
Termin laboratorium	wtorek 10.30-12.00
Grupa	3
Data oddania sprawozdania	05.06.2020r.

https://github.com/KarolinaKachelska/Projekt

1. Temat

Zadanie polegało na napisaniu programu w języku C++ z wykorzystaniem zagadnień poznanych na laboratorium. Wybrany przeze mnie program jest biblioteką gier. Dane przechowywane są w plikach txt.

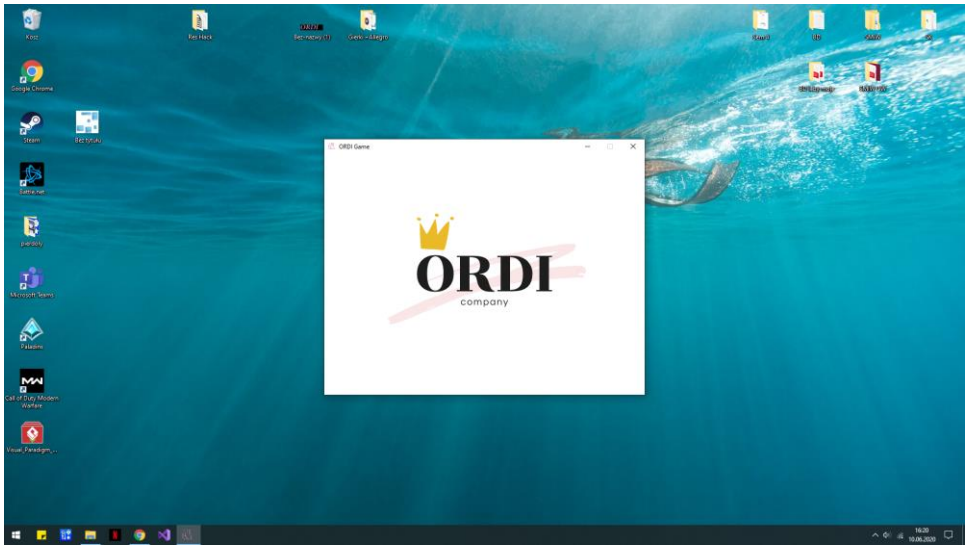
2. Analiza tematu

Przystępując do analizy zadania starałam się brać pod uwagę funkcjonalność i optymalność programu, jak i jak najlepsze doświadczenie dla użytkownika. W związku z tym logika programu jest podatna na rozbudowy i był przyjemny w odbiorze dla użytkownika biblioteki. Program opiera się na grze użytkownika. Biblioteka Gier pomaga mu grać w różne gry używając jednego konta i wchodząc w jedną aplikacje, co znacznie ułatwia rozgrywkę. Przykładem na to są o dużo bardziej rozbudowane biblioteki gier takie jak np. Steam.

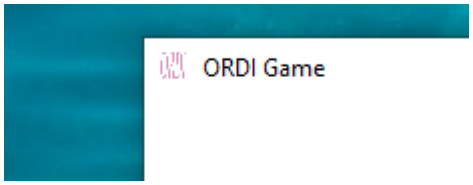
Program jest również podatny na rozbudowę, czego sama doświadczyłam, gdyż początkowo został napisany cały interfejs, a kolejno gry –jedna po drugiej. Istnieje dużo funkcji wirtualnych, które są używane na swój sposób w każdej z gier, jak również zwyczajne funkcje min. Funkcja “End” kończąca rozgrywkę. Ułatwia to dopisane kolejnych gier, tym samym rozbudowę programu.

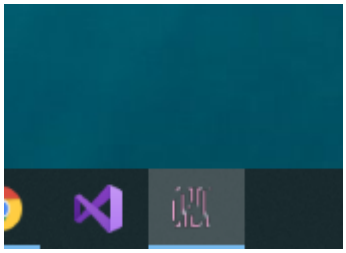
Funkcje początkowe programu są bardzo dopracowane. Wyznaje zasadę, iż pierwsze wrażenie jest najważniejsze. Liczę, iż taki zabieg mógłby zdobyć dobrą opinie u użytkowników. Co możemy zaobserwować :

1) Program rozpoczyna się 4 sekundowym intro (w tle grana jest przyjemna muzyka)



2) Ikona Aplikacji, by program ładniej wyglądał podczas użytkowania.





3. Specyfikacja zewnętrzna

Program korzysta z interfejsu graficznego stworzonego przy pomocy Allegro 5.

Początkowo możemy wejść w menu ogólne, gdzie mamy do wyboru zalogowanie się na konto, bądź rejestracje nowego użytkownika, grę bez konta i wyjście z programu.

- Zaloguj się

Gdy wybierzemy tą opcje dostajemy możliwość zalogowania się na utworzone już konto. Program sprawdza czy podany nick istnieje w bazie danych o nazwie “lista”. Jeżeli istnieje takowa nazwa użytkownika to przekierowuje go do User Menu.

- Zarejestruj się

Gdy wybierzemy tą opcje dostajemy możliwość zarejestrowania się do systemu programu. Daje to użytkownikowi możliwość do zapisywania punktów uzyskanych podczas gry, jak również do pojawienia się w końcowym rankingu, gdzie pokazywane jest 5 najlepszych wyników zdobytych przez użytkowników w danej grze.

Podczas rejestracji nowego użytkownika program sprawdza czy dany nick już istnieje w bazie danych o nazwie “lista”, jeżeli nie, to zapisuje daną nazwę i przekierowuje użytkownika do User Menu.

- Graj jako gość

Program przekierowuje użytkownika do User Menu. Punkty z gier nie będą zapisywane do bazy danych programu.

- User menu

To główne menu, dzięki któremu użytkownik może przejść do wybranej gry. Do wyboru ma:

- Graj w Państwa-Miasta

Program czyta dwa pliki tekstowe. Jeden z nazwami Państw, a drugi z nazwami Stolic. Następnie zapisuje je odpowiednio do list. Jako, iż nazwy są już w pliku posortowane nie widziałam potrzeby tworzenia drzewa binarnego. Użytkownik wpisuje najpierw nazwę Państwa na wylosowaną losowo literę. Jeżeli się pomyli to gra się zakończy i zostaną mu przydzielone uzyskane punkty. Natomiast jeżeli wpisana nazwa będzie poprawna to program poprosi o wpisanie Stolicy na daną literę. Tutaj program działa tak samo w razie pomyłki. W razie odpowiedzi poprawnej możemy wybrać, czy chcemy grać dalej (program wylosuje inną literę i powtórzy prośby o wpisanie kolejno Państwa i Stolicy), czy chcemy zakończyć grę z uzyskanym wynikiem.

Następnie pokazuje się ranking, który jest wyświetlany przez 2 sekundy i program pyta nas czy chcemy wyjść z gry, czy przejść do User Menu (Ten etap powtarza się po zakończeniu każdej gry).

Program jest również zabezpieczony pod względem błędnego wpisania odpowiedzi. Gdy użytkownik stwierdzi, że zrobił literówkę, bądź chce po prostu wpisać coś innego może wtedy użyć klawisza Backspace, który usuwa wcześniej wpisane znaki.

- Graj w Zgadnij Kto To?

Program czyta szesnaście plików tekstowych (każdy do innej postaci). Zawarte są w nim imiona smerfów i pytania, które może zadać użytkownik razem z odpowiedziami, jakie udzieli program. Losowany jest najpierw jeden plik, którego postać będzie "dobrą odpowiedzią" lub inaczej postacią, którą mamy zgadnąć. Następnie losowane są dwa dodatkowe pliki z innymi możliwymi odpowiedziami. Program również losuje kolejność wyświetlania odpowiedzi (a,b i c, gdzie jedna z nich jest "dobrą"). Zostało to użyte by nie było wiadomo na którym miejscu pojawi się prawidłowa odpowiedź, co też utrudnia zgadnięcie postaci i urozmaica grę. Gdy już możliwe odpowiedzi i pytania się pojawią, to użytkownik może wpisać dowolny numer pytania, które chce zadać. Program odpowiada - wyświetlając "tak" lub "nie" przy zadanym pytaniu. Użytkownik może zadać dowolną liczbę pytań i w dowolnym momencie może udzielić odpowiedzi wpisując a,b lub c i zatwierdzając to Enter'em.

Program jest również zabezpieczony pod względem błędnego wpisania pytania. Mianowicie, gdy wpisana liczba nie jest równa liczbą od 1 do 16 to program wypisuję, iż wpisano złą liczbę.

Po udzieleniu odpowiedzi program kończy się tak samo jak zostało opisane w grze Państwa-Miasta.

- Graj w Familiadę

Gra ta przedstawia ostatnią część w popularnym teleturnieju "Familiada".

Program czyta szesnaście plików tekstowych (każdy do innego pytania). Zawarte są tam pytanie i pięć poprawnych odpowiedzi, z czego każde kolejne gorzej jest punktowane.

Wybierane jest pięć losowych plików z pytaniami. Następnie do obiektów klasy F zapisywane są poprawne odpowiedz i pytanie, które zostaje wyświetlone. Gra zaczyna się, gdy program wyświetli pierwsze pytanie. Zadaniem użytkownika jest wpisanie odpowiedzi na pytanie. Tutaj również program został zabezpieczony tak samo jak zostało opisane w grze Państwa-Miasta. Po dokonaniu odpowiedzi na pytanie program sprawdza czy jest one zgodne z zapisanymi. Jeśli tak to przydziela odpowiednio punkty, jeśli nie to liczba punktów wynosi 0. Każdą odpowiedź należy zatwierdzić Enter'em. Koniec gry następuje po udzieleniu odpowiedzi na ostatnie, piąte pytanie.

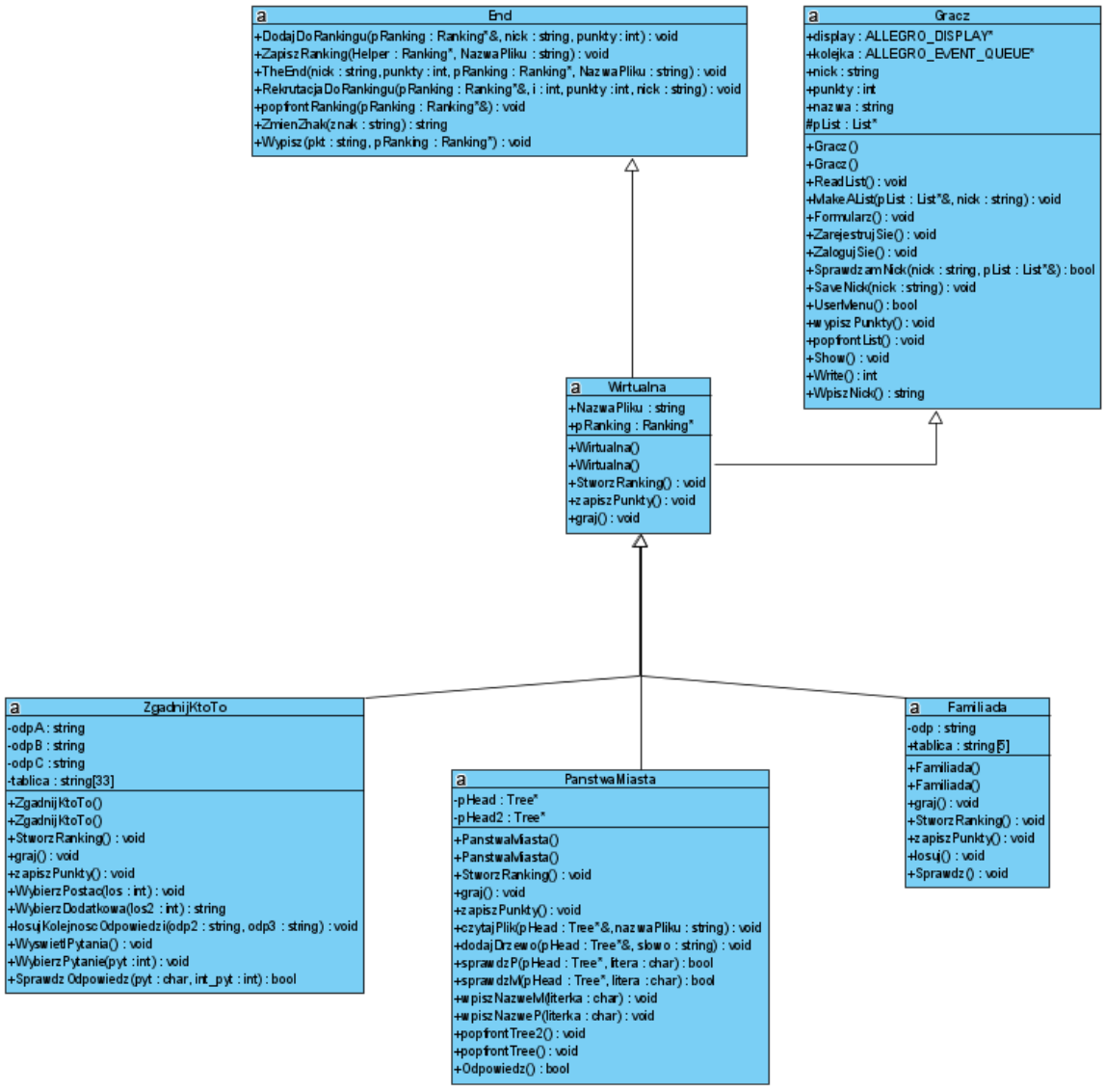
Program kończy się tak samo jak zostało opisane w grze Państwa-Miasta.

- Wyświetl Historie Gier

Program wyświetla punkty za każdą grę zalogowanego użytkownika. Widoczne 2 sekundy, potem program wraca do User Menu.

4. Specyfikacja wewnętrzna

Diagram Klas:



Głównymi funkcjami programu są:

- Void Main() - tworzy główny interfejs gry, jak również znajduje się tu główne menu gry.
- Void Graj() - funkcja wirtualna , wykorzystywana w każdej z gier na swój sposób. Tutaj toczy się rozgrywka.
- bool End() - funkcja , wykorzystywana w każdej z gier(tak samo). Podsumowywuje koniec gry.
- Void UserMenu() - funkcja, w której znajduje się główne menu użytkownika. Tutaj wybierana jest czynność jaką chce wykonać użytkownik.

Klasy wykorzystywane w programie :

- Gracz – klasa główna opisująca interfejs gry, jak również gracza (nick i punkty). Zawiera ona wskaźnik na listę, która prezentuje listę nick'ów w bazie danych.
- Wirtualna – klasa wirtualna, która jest używana przez klasy gier. Prezentuje ona wspólne obiekty takie jak nazwa pliku tekstowego, wskaźnik na listę rankingową czy funkcje graj(). Dziedziczy ona publicznie po klasie End i Gracz.
- PM- klasa opisująca gre Państwa-Miasta. Posiada dwa wskaźniki na listę Tree (lista państw i lista miast). Dziedziczy ona publicznie po klasie Wirtualna.
- ZKT- klasa opisująca gre Zgadnij Kto To. Posiada tablice, w której zapisuje pytania i odpowiedzi. Dziedziczy ona publicznie po klasie Wirtualna.
- F - klasa opisująca gre Familiada. Posiada tablice, w której zapisuje możliwe odpowiedzi do pytań. Dziedziczy ona publicznie po klasie Wirtualna.
- END – klasa posiada funkcje, które są używane po zakończeniu gry, jako podsumowanie.

Zawarte są również w oddzielnym pliku .h struktury wykorzystywane w programie:

- List – lista używana do zapisu i odczytu nick'ów
- Tree – lista używana do odczytu nazw Państw i Miast.
- Ranking - lista używana do zapisu i odczytu rankingu gry.

Przykłady elementów omawianych na labaratorium znajdujących się w projekcie:

- RTTI

```
vector< Wirtualna*> Wir;

//Wirtualna *Wir = new Wirtualna;

PM* pm1 = new PM;
Wir.push_back(pm1);
ZKT* zkt1 = new ZKT;
Wir.push_back(zkt1);
F* f1 = new F;
Wir.push_back(f1);

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

    Wir[i]->StworzRanking();
    Wir[i]->nick = this->nick;

}
```

- Wyjątki

```
plik.open(nazwaPliku, std::ios::in);
try {
if (plik.good() == true)
{


    while (!plik.eof())
    {

        getline(plik, linia);

        istringstream iss(linia);
        iss >> linia;
        NickView = linia;
        iss >> linia;
        PointView = atoi(linia.c_str());
        DodajDoRankingu ( pRanking , NickView, PointView);


    }
    plik.close();
}

else
{
    throw ss;
}
}
catch (string)
{
    cout << "Blad odczytu pliku" << endl;
    Sleep(1000);
    system("cls");
    exit(5);
}
```

- Smart Pointers

```
if (!pList) pList = new List{ nick,nullptr };
else
    return MakeAList(pList->pNext, nick);
```

- kontenery

```
vector< Wirtualna*> Wir;

//Wirtualna *Wir = new Wirtualna;

PM* pm1 = new PM;
Wir.push_back(pm1);
ZKT* zkt1 = new ZKT;
Wir.push_back(zkt1);
F* f1 = new F;
Wir.push_back(f1);

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

    Wir[i]->StworzRanking();
    Wir[i]->nick = this->nick;

}
```

5. Testowanie i uruchamianie

Program działa poprawnie, nie zostały wykryte żadne błędy. Program można uruchomić z pliku .exe lub normalnie po kompilacji w visualu.

6. Wnioski

Zrealizowanie tego projektu umożliwiło mi skonstruowanie programu z interfejsem graficznym wykonanym w Allegro 5, pozwalającym na wygodną interakcję użytkownika z programem. Problem oddzielenia interfejsu od logiki oraz optymalizacja silnika gry zdecydowanie rozwinął moją umiejętność tworzenia oprogramowania. Skorzystanie z zagadnień omawianych na laboratorium pozwoliło mi dzięki użyciu wyjątków na wygodne przechwytywanie błędów w programie. Użyte zostały także smartpointery, RTTI, oraz kontenery. Myślę, że udało mi się zapewnić przyjemną obsługę programu oraz bardzo wygodny, podatny na rozbudowę silnik.