Politechnika Śląska
Wydział Matematyki Stosowanej
Kierunek Informatyka

Gliwice, 02.02.2022

Programowanie I

projekt zaliczeniowy

"Dokumenty, proszę!"

Kinga Grabarczyk gr. lab. 2/3

1. Opis projektu.

Stworzony przeze mnie projekt jest grą uruchamianą w konsoli pt. „Dokumenty, proszę!”. Inspiracją do jej stworzenia była istniejąca już gra komputerowa „Papers, please!”.

W grze wcielamy się w inspektora, który musi sprawdzać dokumenty ludzi przechodzących przez granicę, zgodnie z panującymi w danym dniu przepisami. Gra zawiera pięć pełnych dni pracy i zakończenie.

2. Wymagania

Aplikacja spełnia poniższe wymagania:

1. Zawiera menu z tytułem gry oraz opisem jak grać,
2. Losowo generuje dokumenty typu paszport, dowód osobisty itp.,
3. Losowo tworzy błędy w zgodności ww. dokumentów,
4. Zawiera mechanikę wskazywania niezgodności w dokumentach,
5. Zawiera mechanikę podejmowania decyzji o zezwoleniu lub odmówieniu petentowi wejścia na teren kraju,
6. Ma ograniczenie czasowe jednego dnia pracy (różne w zależności od dnia),
7. Ma możliwość zapisu gry i wznowienia jej po ponownym uruchomieniu oraz stworzenia nowej gry,
8. Są w niej krótkie dialogi,
9. Zawiera podsumowanie dnia pracy gracza,
10. Zawiera koniec gry zależny od podjętych przez gracza decyzji,
11. Zawiera napisy w różnych kolorach i polskie znaki,
12. Zapewnia oryginalną ścieżkę dźwiękową,
13. Wyświetla tekst gry jak maszyna do pisania tzn. „litera po literze”.

3. Przebieg realizacji

Tworzenie projektu rozpoczęłam od przygotowania planu działania oraz rozbicia poszczególnych funkcjonalności na mniejsze zadania z pomocą aplikacji Trello (podział na TO DO, IN PROGRESS i DONE), co pozwoliło mi w łatwy sposób kontrolować przebieg realizacji. Po stworzeniu każdej funkcjonalności gra była testowana.

Na samym początku zajęłam się stworzeniem menu głównego i menu wyboru zapisu gry. Następnie stworzyłam funkcję, która wypisuje tekst na ekranie znak po znaku, a dalej, aby ułatwić czytanie kodu – wypisywanie dłuższych tekstów z pliku .txt.

Następnym krokiem było stworzenie obiektu, który będzie przechowywał różne dane – w tym celu stworzyłam strukturę, która przechowuje losowo generowane dane. Dalej – należało na podstawie tych danych wylosować, w której z nich ma pojawić się błąd – do tego ponownie użyłam struktury tego samego typu oraz stworzyłam funkcję, która tworzy różnego rodzaju błędy np. zamienia litery (odpowiednio małe i duże) oraz cyfry.

Najważniejsza mechanika gry, czyli wybieranie, które dane nie zgadzają się z obecnymi przepisami dnia, sprawiła najwięcej trudności.

Początkowo wybieranie błędu miało być realizowane za pomocą wpisywania numeru błędu z listy, jednak lepszym pomysłem było stworzenie menu z „przeskakiwaniem" po kolejnych opcjach.

Program sprawdza, w zależności od dnia, czy dane są poprawne, jest to najobszerniejszy kawałek kodu, ponieważ składa się z wielu połączonych ze sobą funkcji.

```
bool CheckIfMistakeIsHere(int choice, PASSPORT fake_passport, ID_CARD fake_id_card)
{
    switch (choice)
    {
    case 1:
        if (fake_passport.name == fake_id_card.name)
            return false;
        break;
    case 2:
        if (fake_passport.surname == fake_id_card.surname)
            return false;
        break;
    case 3:
        if ((DayNumber == 3) && (!(IsAgeOK("56", fake_passport.birth_date, 18, 65)) || !(IsAgeOK("56", fake_id_card.birth_date, 18, 65))))
            return true;
        if (fake_passport.birth_date == fake_id_card.birth_date)
            return false;
        break;
    case 4:
    {
        if ((DayNumber == 1) && fake_passport.country != "Kavertia")
            return true;
        if (fake_passport.country == fake_id_card.country)
            return false;
        break;
    }
    case 5:
        if (fake_passport.code == fake_id_card.code)
```

```
do
{
    choice = SelectMistake(choice, how_many_data+1);
    gotoxy(60, 21);
    cout << "                                    ";
    cout << "\r";

    if (choice == how_many_data+1)
    {
        gotoxy(60, 21);
        cout << "W dokumentach nie ma bledu.";
        Sleep(1000);
        mistake_is_here = true;
    }
    else
    {
        mistake_is_here = CheckIfMistakeIsHere(choice, fake_passport, fake_id_card);
        gotoxy(60, 21);
        if (mistake_is_here)
        {
            Beep(600, 200);
            cout << "Wykryto niezgodnosc.";
            gotoxy(60, 23);
            Typewriter("> Te dane sie nie zgadzaja.", 9);
        }
        else
        {
            Beep(120, 50);
```

W zależności od tego, czy gracz wybierze opcję „ODMOWA” lub „UZNANO” – dostaje punkty, które będą skutkowały jednym z czterech zakończeń (brak pieniędzy na utrzymanie, neutralne, złe i dobre), od tego też zależy, ile pieniędzy otrzyma na koniec dnia.

Wykorzystane biblioteki:

1. *<fstream> - biblioteka do odczytu i zapisu plików*
2. *<ctime> - biblioteka do pobierania czasu*
3. *<random> - biblioteka do losowego generowania*
4. *<stdlib.h> - biblioteka do poleceń systemowych*
5. *<windows.h> - biblioteka zawierająca funkcję Sleep()*
6. *<iomanip> - biblioteka do manipulacji wyglądu wyjścia*
7. *<string> - biblioteka pozwalająca na użycie typu string*
8. *<conio.h> - biblioteka zawierająca funkcję _getch()*
9. *<locale.h> - biblioteka pozwalająca na dodanie polskich znaków*

Program składa się z:

1. *Main.cpp – plik główny*
2. *Functions.cpp – plik ze wszystkimi funkcjami*
3. *Nagłówek.h – plik z nazwami funkcji*
4. *data.txt – plik do zapisu gry*
5. *DayNews.txt – plik z „wiadomościami codziennymi”*
6. *DayRules.txt – plik z „przepisami prawnymi”*
7. *SoundTrackTwo.wav – plik dźwiękowy*
8. *Innych plików pomocniczych*

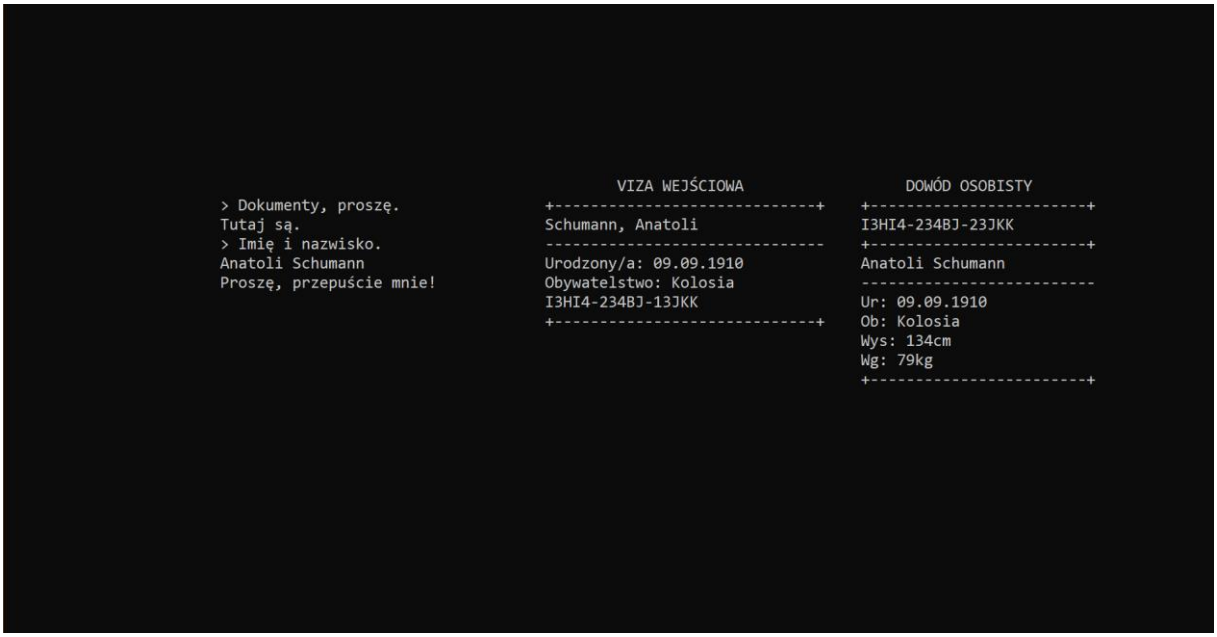
4. Instrukcja użytkownika

Po uruchomieniu programu użytkownik ma dostęp do menu głównego. Z tego poziomu może wyjść z gry, sprawdzić jak grać lub rozpocząć grę. Nawigacja po menu odbywa się za pomocą strzałek „góra”, „dół”. Klawiszem powrotu jest strzałka w lewo. Zatwierdza się klawiszem „Enter”.

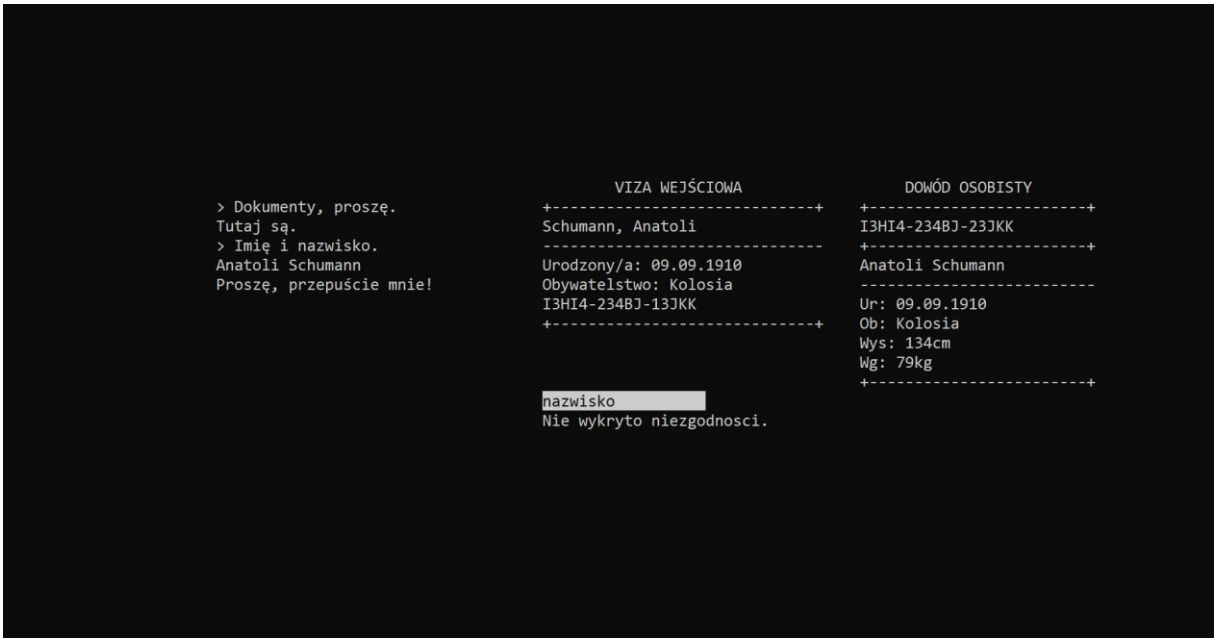
Po wybraniu opcji „GRAJ” użytkownik może wybrać, czy chce rozpocząć nową grę czy kontynuować zapisaną (wczytać zapisane dane z pliku).

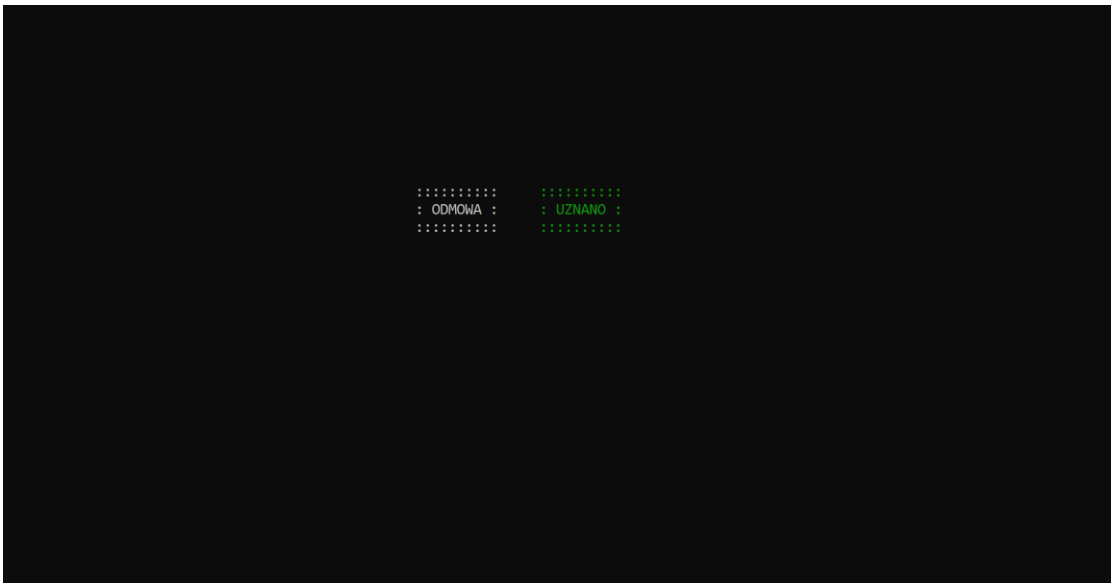
Po pojawieniu się zdania „Rozpocznij dzień” należy użyć klawisza „Enter”.

W głównej grze posługujemy się strzałkami „góra”, „dół” do wybrania z menu błędów, interesującej nas opcji (zatwierdzenie klawiszem „Enter”). W zależności, jakiego wyboru dokonaliśmy, pokaże się odpowiedni komunikat.



Jeśli wybierzemy poprawnie i wyświetli się „Wykryto niezgodność” lub wybraliśmy „Brak” – program przekieruje nas do ekranu (w którym poruszamy się strzałkami „prawo”, „lewo”), gdzie możemy dokonać wyboru i zatwierdzić go „Enterem”. Po nim pokaże się odpowiedni komunikat.





Po upływie określonego czasu nastąpi „Koniec dnia” i gracz otrzyma podsumowanie zarobionych pieniędzy w danym dniu.

Aby kontynuować grę, należy użyć strzałki w „prawo”.

Po przejściu pięciu dni nastąpi jedno z możliwych zakończeń.

5. Podsumowanie i wnioski.

Udało mi się zrealizować wszystkie założenia, oprócz „fabuły i decyzji, które wpłyną na świat gry”. Nie udało mi się użyć biblioteki PDcurses do stworzenia lepszego interfejsu gry. Pod koniec starałam się natomiast dodać polskie znaki do całej gry, co było dosyć trudnym zadaniem.

Planuję dodać do gry więcej „dni pracy”, więcej przepisów, które należałoby sprawdzać, więcej dokumentów, lepszą fabułę i wybory wpływające na grę (jednak nawet bez tego gra jest całkiem grywalna).

Mimo to, uważam, że założenia zostały w znacznej mierze spełnione. Dodatkowym atutem projektu jest to, że mechanikę i sposób wykonania musiałam od zera wymyślić i stworzyć sama, bazując jedynie na przyglądaniu się, jak działa gra Papers, please.