

Politechnika Śląska  
Wydział Automatyki, Elektroniki i  
Informatyki

## Programowanie komputerów 3

### Gra karciana 66

**Tomasz Nowok gr. 1 semestr 3**

## 1. Temat zadania

Zadanie polegało na napisaniu gry karcianej 66. Rozgrywka miała toczyć się pomiędzy graczem a komputerem (botem).

## 2. Specyfikacja programu

Program został zrealizowany zgodnie z paradygmatem obiektowym. Logika gry została wyodrębniona do poszczególnych klas, zostały zastosowane mechanizmy dziedziczenia, abstrakcji czy hermetyzacji. Funkcja main w pliku Source.cpp stanowiła punkt wejściowy i wyjściowy programu. W pliku tym również zostały zainicjalizowane instancje odpowiednich klas odpowiadających za przebieg gry.

### Opis zastosowanych struktur danych:

W projekcie została wykorzystana lista jednokierunkowa. Została zaimplementowana jako klasa.

### Opis funkcji main:

Na wejściu włączane są odpowiednie flagi, które sprawdzają wycieki w programie.

```
_CrtSetDbgFlag(_CRTDBG_ALLOC_MEM_DF | _CRTDBG_LEAK_CHECK_DF);  
_CrtSetReportMode(_CRT_WARN, _CRTDBG_MODE_DEBUG);
```

Reszta funkcji dzieje się w bloku try catch (w celu przechwycenia i obsłużenia ewentualnych wyjątków).

Na początku inicjalizowana jest zmienna game, menuUi oraz gameUi.

```
Game* game = new Game();
MenuUIManager* menuUi = new MenuUIManager();
GameUIManager* gameUi = new GameUIManager(game);
```

Kolejne linijki kodu odpowiadają za uruchomienie widoku menu głównego.

```
if (!menuUi->GetGameStarted())
    menuUi->View(ViewName::Home);
```

Kolejnym krokiem jest uruchomienie pętli while w celu zapętlenie rozgrywki (po skończeniu 1 gry, gra się nie wyłącza i powraca do menu głównego).

```
while (!game->GetGameEnded())
{
    if (menuUi->GetIsExit())
        break;

    gameUi->View(ViewName::StartRoundGame);

    if (gameUi->GetRestartGame())
    {
        menuUi->SetGameStarted(false);
        game = new Game();
        gameUi->SetGame(game);
        menuUi->View(ViewName::Home);
    }
}
```

Przedostatnim krokiem jest wyczyszczenie pamięci.

```
delete game;  
delete menuUi;  
delete gameUi;
```

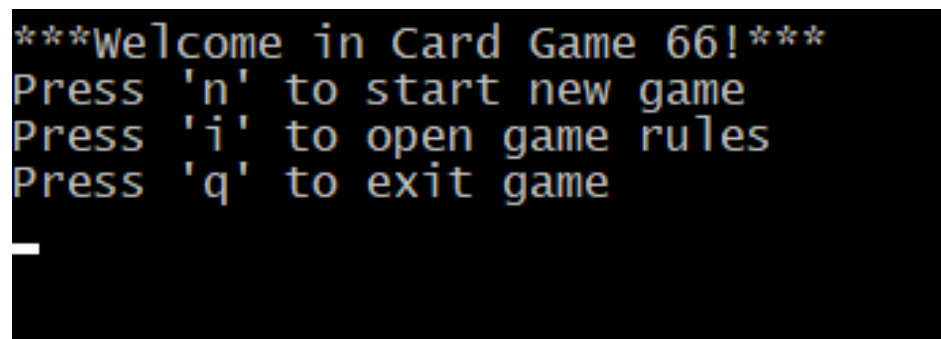
Ostatnim krokiem jest wyświetlenie ewentualnych wycieków w konsoli wyjściowej.

```
_CrtDumpMemoryLeaks();
```

**Szczegółowa dokumentacja i opis poszczególnych klas, funkcji oraz zmiennych znajduje się w kodzie w komentarzach.**

### 3. Opis rozgrywki

Gra rozpoczyna się od ekranu menu głównego.



```
***Welcome in Card Game 66!***  
Press 'n' to start new game  
Press 'i' to open game rules  
Press 'q' to exit game  
_
```

Po wpisaniu odpowiedniej literki program przekieruje nas na inny widok.

Poniżej widzicie widok zasad gry.

```
***Rules***
*Game ends when one of players collects 66 or more points*
1. Deck contains 24 cards
2. Both players got 6 cards and after round they take one card from the deck
3. First card from the top of deck goes to round winner, the second to loser
4. Player can play one card each round with following values:
    AS = 11 points
    Ten = 10 points
    King = 4 points
    Queen = 3 points
    Joker = 2 points
    Nine = 0 points
5. If player won last round and has in their hand king and queen with the same color, they could play report
    trump color = 40 points
    another color = 20 points
6. If deck is empty, players use only their own cards until one of them collects at least 66 points
7. If all cards were used and anyone has 66 points, the man with more points wins
Press 'b' to back to main menu
_
```

Po wciśnięciu klawisza n w menu głównym rozpoczyna się gra.  
Początek rundy wygląda następująco:

```
***Round 1***
Your score: 0
Deck count: 12
Trump color: PIKE
Card from stack: Ten [PIKE]
***Your cards***
*Press NUMBER of card to play it*
Card #1: Nine [PIKE]
Card #2: Joker [HEART]
Card #3: King [HEART]
Card #4: Queen [HEART]
Card #5: Joker [TILE]
Card #6: Queen [TILE]
Press '0' to report!
_
```

Wyświetlane są tutaj wszystkie statystyki gry takie jak ilość punktów, ilość kart w talii, kolor atutowy oraz wszystkie karty posiadane przez gracza w ręce.

Po wpisaniu odpowiedniego numeru karty lub 0 (jako meldunek) turę wykonuje bot oraz wyświetlane jest podsumowanie rundy.

```
5
*Turn result*
You played card: Joker [TILE] - 2 points
Bot played card: King [TILE] - 4 points
You lost this round!
Points scored by bot: 6
*****
```

Automatycznie gra wygeneruje ekran kolejnej rundy:

```
***Round 2***
Your score: 0
Deck count: 10
Trump color: PIKE
Card from stack: Ten [TILE]
***Your cards***
*Press NUMBER of card to play it*
Card #1: Nine [PIKE]
Card #2: Joker [HEART]
Card #3: King [HEART]
Card #4: Queen [HEART]
Card #5: Queen [TILE]
Card #6: AS [PIKE]
Press '0' to report!
```

Jeśli warunek końca gry (jeden z graczy osiągnie 66 punktów lub skończą się wszystkie karty) zostanie spełniony wyświetla się następujący ekran:

```
Points scored by bot: 15
***YOU LOST THE GAME WITH SCORE: 47***
*Bot points: 74*
*Rounds count: 12*
Press any key to continue...
*****
```

Po zamknięciu okna końca gry aplikacja powraca do menu głównego.

## 4. Testowanie

Gra została przetestowana pod kątem różnych możliwych przebiegów.

1. Zagranie asa przeciwko królowej ( $11 > 3$ ) => runda wygrana przez gracza + graczowi została przyznana suma punktów za asa i królową  $11+3=14$ . Liczba kart została zredukowana o 2 (jedna karta przyznana graczowi, druga karta botowi). Graczowi została przyznana pierwsza karta z stosu (Queen [CLOVER]) jako iż wygrał rundę.

```
1
*Turn result*
You played card: AS [PIKE] - 11 points
Bot played card: Queen [HEART] - 3 points
You won this round!
Points scored: 14
*****
***Round 2***
Your score: 14
Deck count: 10
Trump color: CLOVER
Card from stack: King [HEART]
***Your cards***
*Press NUMBER of card to play it*
Card #1: Nine [CLOVER]
Card #2: AS [CLOVER]
Card #3: Ten [CLOVER]
Card #4: AS [TILE]
Card #5: Ten [TILE]
Card #6: Queen [CLOVER]
Press '0' to report!
```



2. Zagranie dziewiątki przeciwko jokerowi ( $0 < 2$ ) => runda przegrana przez gracza + bot otrzymuje 2 punkty, a stos dekrementuje się po raz kolejny o 2. Karta pierwsza ze stosu wędruje do bota.

```
Press '0' to report!  
1  
*Turn result*  
You played card: Nine [CLOVER] - 0 points  
Bot played card: Joker [HEART] - 2 points  
You lost this round!  
Points scored by bot: 2  
*****  
***Round 3***  
Your score: 14  
Deck count: 8  
Trump color: CLOVER  
Card from stack: King [TILE]  
***Your cards***  
*Press NUMBER of card to play it*  
Card #1: AS [CLOVER]  
Card #2: Ten [CLOVER]  
Card #3: AS [TILE]  
Card #4: Ten [TILE]  
Card #5: Queen [CLOVER]  
Card #6: King [CLOVER]  
Press '0' to report!
```

3. Zagranie meldunku królowa + król [CLOVER] => otrzymanie 40 punktów za meldunek o kolorze atutowym

```
Card #3: Ten [TILE]  
Card #4: Queen [CLOVER]  
Card #5: King [CLOVER]  
Card #6: King [TILE]  
Press '0' to report!  
0  
*Turn result*  
*You played report!*  
*Report points: 40*  
*****  
***Round 5***  
Your score: 65  
Deck count: 4  
Trump color: CLOVER  
Card from stack: Nine [PIKE]  
***Your cards***  
*Press NUMBER of card to play it*  
Card #1: Ten [CLOVER]  
Card #2: AS [TILE]  
Card #3: Ten [TILE]  
Card #4: King [TILE]  
Card #5: Nine [HEART]  
Card #6: Ten [HEART]  
Press '0' to report!
```

4. Zagranie dziesiątki przeciwko królowi => wygranie rundy i przekroczenie progu 66 punktów => wygranie gry

```
Press 0 to report:
3
*Turn result*
You played card: Ten [TILE] - 10 points
Bot played card: King [PIKE] - 4 points
You won this round!
Points scored: 14
***YOU WON THE GAME WITH SCORE: 79***
*Bot points: 2*
*Rounds count: 5*
Press any key to continue...
*****
***Game result***
***Welcome in Card Game 66!***
Press 'n' to start new game
Press 'i' to open game rules
Press 'q' to exit game
```

5. Bot zagrywający meldunek otrzymał 40 punktów (meldunek atutowy)

```
Press 0 to report:
1
*Turn result*
*Bot played report!*
*Report points: 40*
*****
```

6. Bot zagrywający meldunek otrzymał 20 punktów i wygrał grę

```
Press 0 to report!  
5  
*Turn result*  
*Bot played report!*  
*Report points: 20*  
***YOU LOST THE GAME WITH SCORE: 12***  
*Bot points: 67*  
*Rounds count: 3*  
Press any key to continue...  
*****
```

## 5. Wnioski

Podjęcie obiektowe bardzo ułatwia pisanie aplikacji. Wygodne i przejrzyste rozdzielenie klas odpowiadających za poszczególne funkcje programu powoduje, że program staje się czytelny i prosty w zarządzaniu. W sytuacji, gdy musiałem rozszerzyć grę o kolejne rzeczy np. meldunki nie było to większym problemem. Ponad to po skończeniu projektu w momencie gdy zlikwidowałem wszystkie wycieki pamięci doszedłem do wniosku, że destruktory są naprawdę pożyteczne i ułatwiają zarządzanie pamięcią.