**Uniwersytet Przyrodniczo-Humanistyczny**

**w Siedlcach**

Kierunek Informatyka

Igor Kucyk

**Temat zadania:**

Projekt i implementacja gry „Kości” dla dwóch graczy z systemem historii gier i krótkimi wprowadzającymi zasadami.

Dokumentacja

Siedlce, 2020

**Spis treści:**

1. **Sformułowanie zadania 2**
2. **Funkcjonalność 3**
3. **Schematy blokowe wybranych funkcji 4**

* **int main() 4**
* **void Gra() 5**
* **void TKosci::Zapisz() 6**

1. **Instrukcja dla użytkownika 7**
2. **Opis rozwiązania 7**
3. **Opis wybranych funkcji 8**
4. **Kod programu 9**

* **main.cpp 9**
* **menu.h 9**
* **menu.cpp 9**
* **operacje\_na\_consoli.h 12**
* **operacje\_na\_consoli.cpp 12**
* **znaki\_specjalne.h 12**
* **znaki\_specjalne.cpp 13**
* **kosci.h 13**
* **kosci.cpp 14**

1. **Przykłady działania programu 18**
2. **Sformułowanie zadania**

Napisać grę w kości dla dwóch graczy. Zaimplementować opis zasad gry, system zapisywania historii meczy oraz niezbędne do poprawnej rozgrywki funkcje, takie jak zatrzymywanie kości, wybór kategorii itd.

Zasady, które będą panować w grze:

Gra polega na rzucaniu kośćmi, aby uzyskać określone układy oczek, za które otrzymuje się punkty. W każdej z trzynastu rund gracze mają do dyspozycji trzy rzuty. Pierwszy z nich odbywa się zawsze wszystkimi pięcioma kostkami, a w drugim i trzecim, które nie są obowiązkowe, wybrane kostki mogą zostać zatrzymane – rzut odbywa się wtedy tylko niezatrzymanymi.

Po wykonaniu rzutów, układ oczek uzyskany na kostkach musi zostać zapisany przez gracza w tabeli punktacji do jednej z trzynastu kategorii. Raz wybrana kategoria nie może być później ponownie wybrana.

Kategorie lewej części tabeli:

* Jedynki, dwójki, trójki, czwórki, piątki, szóstki – suma oczek liczb z wybranej kategorii (np. 4, 3, 4, 4, 1 – gracz wskazuje „czwórki”, więc zyskuje 12 pkt.) Jeśli w lewych kategoriach tabeli uzyska się razem 63 pkt. lub więcej, to dodatkowo otrzymuje się premię w wysokości 35 pkt.

Kategorie prawej części tabeli:

* 3 jednakowe – suma oczek wszystkich pięciu kostek, jeśli występują co najmniej trzy jednakowe liczby oczek
* 4 jednakowe – suma oczek wszystkich pięciu kostek, jeśli występują co najmniej cztery jednakowe liczby oczek
* Full – 25 pkt., jeśli w układzie występują trzy jednakowe liczby oczek oraz dwie inne jednakowe
* Mały strit – 30 pkt., jeśli przynajmniej cztery kości mają kolejne liczby oczek
* Duży strit – 40 pkt., jeśli wszystkie kości mają kolejne liczby oczek
* Generał – 50 pkt., jeśli liczba oczek na wszystkich kostkach jest jednakowa
* Szansa – niezależnie od układu kostek uzyskuje się sumę oczek z wszystkich kostek

Wyboru każdej z wolnych kategorii można dokonać niezależnie od posiadanego układu, przy czym jeśli nie spełnia on określonych wymagań, to otrzymuje się 0 pkt. (taki wybór za 0 pkt. jest czasami konieczny, kiedy układ nie pasuje do żadnej z wolnych kategorii).

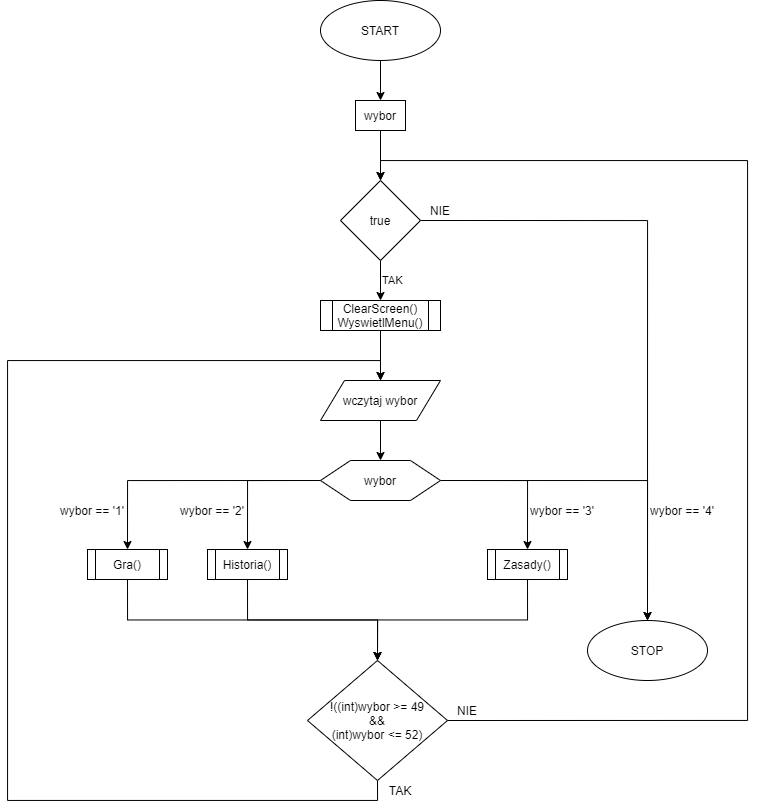
Drugi, trzeci i każdy kolejny wyrzucony generał może zostać użyty jako joker, jeśli kategoria „generał” i kategoria lewej części tabeli odpowiadająca danemu generałowi są już wykorzystane. Jokera zapisuje się w prawych kategoriach tabeli punktacji, otrzymując stosowną liczbę punktów (np. 25 pkt. za „full” itd.), chyba, że wszystkie prawe kategorie są już wykorzystane, wtedy trzeba go zapisać do jednej z lewych za 0 pkt. Jeśli joker został użyty co najmniej raz, to otrzymuje się 100 pkt. premii.

1. **Funkcjonalność**

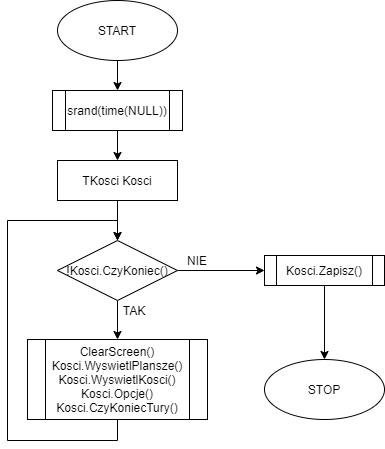
* Obsługa dwóch graczy
* Automatycznie aktualizująca się tabela punktów
* Losowanie niezatrzymanych kości
* Zmienianie stanu kości (zatrzymane/niezatrzymane)
* Nieobowiązkowe dodatkowe dwa rzuty (razem trzy rzuty)
* Wybór dostępnych kategorii w tabeli (po trzech rzutach lub po wybraniu odpowiedniej opcji)
* Podpowiedzi odnośnie ilości punktów dla dostępnych kategorii w zależności od układu kości
* Opis zasad gry
* System zapisywania historii meczu (po zakończonej rozgrywce program poprosi graczy o podanie swoich imion, które zostaną następnie razem z pełną tabelą punktów zapisane do plików i będą one dostępne do obejrzenia w specjalnie na to przeznaczonym miejscu)

1. **Schematy blokowe wybranych funkcji**

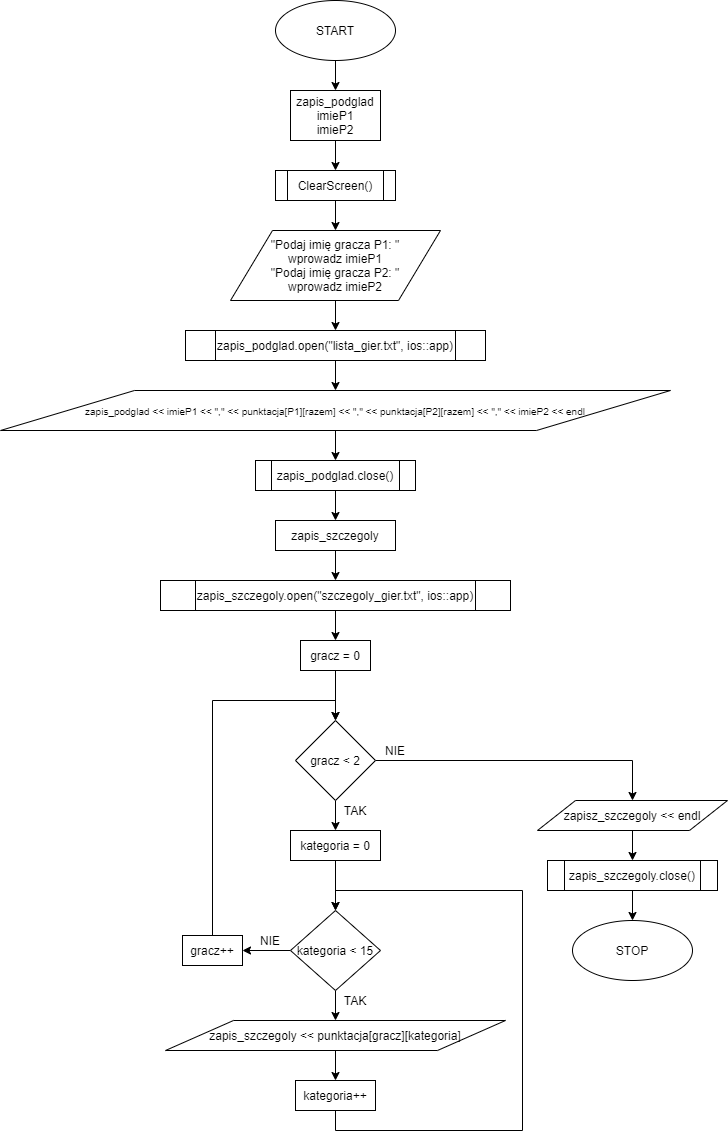
**int main()**

****

**void Gra()**

****

**void TKosci::Zapisz()**

****

1. **Instrukcja dla użytkownika**
2. Wejdź w katalog, w którym znajduje się program
3. Nie modyfikuj plików „lista\_gier.txt” oraz „szczegoly\_gier.txt”. Jeżeli brakuje choć jednego z nich, należy usunąć pozostały plik. Jeżeli oba pliki nie istnieją, to zostaną utworzone po zakończeniu meczu i wpisaniu imion graczy przynajmniej raz
4. Uruchom znajdujący się w otwartym wcześniej katalogu program „Kosci.exe” klikając na niego dwukrotnie
5. Programem steruje się naciskając klawisze odpowiadające danej opcji, np.

chcąc wybrać opcję „[1] Rozpocznij grę”, należy wcisnąć klawisz „1”

UWAGA! Wyjątkiem jest historia meczy, gdzie należy najpierw wpisać opcję, a następnie zatwierdzić ją enterem

1. Jeżeli włączasz program po raz pierwszy, zaleca się przeczytanie zasad gry(opcja „3” w głównym menu)
2. Będąc w głównym menu, wciśnij przycisk „1” aby rozpocząć grę.
3. Przyciski od „1” do „5” służą do zmiany stanu kości. Przycisk „A” służy do ponownego rzutu, zaś przycisk „B” przekierowuje do wyboru kategorii w tabeli
4. W wyborze kategorii należy wcisnąć przycisk odpowiadający wybranej przez użytkownika kategorii (np. „A” dla „JEDYNKI”)
5. Po zakończeniu rozgrywki należy wpisać imię pierwszego gracza i zatwierdzić enterem, a następnie powtórzyć czynność dla drugiego gracza
6. Po zakończonej rozgrywce i wpisaniu imion, historia meczu zostanie zapisana do odpowiednich plików. Z poziomu głównego menu, wchodząc w „Historia meczy”, można obejrzeć listę rozegranych wcześniej gier oraz, po wybraniu odpowiedniej opcji, sprawdzić tabelę punktów danego meczu
7. **Opis rozwiązania**

Pierwsza wersja programu została napisana z użyciem funkcji. Z powodu licznych zmiennych, funkcje te miały ogromną ilość argumentów, przez co kod był nieczytelny, a naprawianie błędów było znacząco utrudnione. Postanowiłem napisać program od nowa, lecz tym razem z użyciem klasy oraz wykorzystując wiedzę zdobytą z książki „Opus Magnum C++11” autorstwa Jerzego Grębosza.

Klasa „TKosci” kryje w sobie całą mechanikę gry (np. losowanie kości, wyświetlanie planszy, interakcja z użytkownikiem, zapisywanie historii meczu itd.). Metody płynnie wykorzystują inne metody oraz operują na licznych zmiennych i stałych.

System historii gier opiera się na zapisywaniu i odczytywaniu danych z dwóch plików – „lista\_gier.txt” oraz „szczegoly\_gier.txt”. Zdecydowanie najtrudniejszym zadaniem była implementacja poprawnego obliczania ilości punktów dla różnych kategorii (których jest razem 14) w zależności od układu wylosowanych kości oraz późniejsze wykorzystanie potrzebnych wartości w innych miejscach programu.

1. **Opis wybranych funkcji**

**void WyswietlMenu()** – wyświetla główne menu, które zawiera następujące opcje: „Rozpocznij grę”, „Historia meczy”, „Zasady”, „Wyjście”

**void Historia()** – pokazuje listę rozegranych gier w formacie:

ImieP1 IlośćPunktówP1 : IlośćPunktówP2 ImieP2

po wybraniu opcji (od „1” do N, gdzie N równa się ilości rozegranych gier), program ukazuje użytkownikowi podgląd tabeli punktów danego meczu. Opcja „0” przenosi do menu głównego

**short TKosci::IloscPunktow(const short kategoria, bool joker, bool czy\_podpowiedz)** – zwraca ilość punktów wybranej kategorii

**void TKosci::Opcje()** – wyświetla dostępne opcje w trakcie gry. Są dwa warianty opcji: pierwszy umożliwia zmiany stanu kości, ponowny rzut lub natychmiastowe przejście do wyboru kategorii; drugi pokazuje dostępne (niewybrane) kategorie oraz ilość punktów w zależności od układu kości.

**void TKosci::Zapisz()** – pobiera od użytkownika imiona graczy, a następnie zapisuje dane do dwóch plików: „lista\_gier.txt”, w którym znajdują się imiona oraz ilość punktów uzyskanych w grze; „szczegoly\_gier.txt”, w którym znajduje się punktacja wykorzystywana do przedstawienia użytkownikowi podglądu tabeli

1. **Kod programu**

**main.cpp**

#include <iostream>

#include <windows.h>

#include <conio.h>

#include "menu.h"

#include "operacje\_na\_consoli.h"

#include "kosci.h"

using namespace std;

int main()

{

char wybor;

while(true)

{

ClearScreen();

WyswietlMenu();

do

{

wybor = getch();

switch(wybor)

{

case '1':

Gra();

break;

case '2':

Historia();

break;

case '3':

Zasady();

break;

case '4':

return 0;

}

} while(!((int)wybor >= 49 && (int)wybor <= 52));

}

return 0;

}

**menu.h**

#ifndef MENU\_H\_INCLUDED

#define MENU\_H\_INCLUDED

void Zasady();

void Historia();

void WyswietlMenu();

#endif // MENU\_H\_INCLUDED

**menu.cpp**

#include <iostream>

#include <windows.h>

#include <conio.h>

#include <iomanip>

#include <fstream>

#include <sstream>

#include "menu.h"

#include "znaki\_specjalne.h"

#include "operacje\_na\_consoli.h"

using namespace std;

void Zasady()

{

ClearScreen();

cout << "Gra polega na rzucaniu ko\x98\x86mi, aby uzyska\x86 okre\x98lone uk\x88\aady oczek, za kt\xA2re otrzymuje si\xA9 punkty.\n"

<< "\n"

<< "W ka\xBE\dej z trzynastu rund gracze maj\xA5 do dyspozycji trzy rzuty. Pierwszy z nich odbywa si\xA9 zawsze wszystkimi pi\xA9\cioma\n"

<< "kostkami, a w drugim i trzecim, kt\xA2re nie s\xA5 obowi\xA5zkowe, wybrane kostki mog\xA5 zosta\x86 zatrzymane - rzut odbywa si\xA9 wtedy\n"

<< "tylko niezatrzymanymi.\n"

<< "\n"

<< "Po wykonaniu rzut\xA2w, uk\x88\aad oczek uzyskany na kostkach musi zosta\x86 zapisany przez gracza w tabeli punktacji do jednej\n"

<< "z trzynastu kategorii. Raz wybrana kategoria nie mo\xBE\x65 by\x86 p\xA2\xABniej ponownie wybrana.\n"

<< "\n"

<< "\n"

<< "\n"

<< "Strona 1/3 - naci\x98nij enter aby przej\x98\x86 dalej";

cin.clear(); cin.sync(); cin.get();

ClearScreen();

cout << C9; w(CD, 10); cout << CB << CD << "P1" << CD << CB << CD << "P2" << CD << BB << "\t" << C9; w(CD, 10); cout << CB << CD << "P1" << CD << CB << CD << "P2" << CD << BB << "\n"

<< BA << " JEDYNKI " << BA << " " << BA << " " << BA << "\t" << BA << " 3 JEDN. " << BA << " " << BA << " " << BA << "\t" << "Kategorie lewej cz\xA9\x98\ci tabeli\n"

<< BA << " DW\xE0JKI " << BA << " " << BA << " " << BA << "\t" << BA << " 4 JEDN. " << BA << " " << BA << " " << BA << "\n"

<< BA << " TR\xE0JKI " << BA << " " << BA << " " << BA << "\t" << BA << " FULL " << BA << " " << BA << " " << BA << "\t" << "JEDYNKI, DW\xE0JKI, ..., SZ\xE0STKI - suma oczek liczb z wybranej kategorii\n"

<< BA << " CZW\xE0RKI " << BA << " " << BA << " " << BA << "\t" << BA << "MA\x9DY STRIT" << BA << " " << BA << " " << BA << "\t" << "(np. 4, 3, 4, 4, 1 - gracz wskazuje \"CZW\xE0RKI\", wi\xA9\c zyskuje 12 pkt.).\n"

<< BA << " PI\xA4TKI " << BA << " " << BA << " " << BA << "\t" << BA << "DU\xBDY STRIT" << BA << " " << BA << " " << BA << "\t" << "Je\x98li w lewych kategoriach tabeli uzyska si\xA9 razem 63 pkt. lub wi\xA9\cej,\n"

<< BA << " SZ\xE0STKI " << BA << " " << BA << " " << BA << "\t" << BA << " GENERA\x9D " << BA << " " << BA << " " << BA << "\t" << "to dodatkowo otrzymuje si\xA9 premi\xA9 w wysoko\x98\ci 35pkt.\n"

<< BA << " PREMIA " << BA << " " << BA << " " << BA << "\t" << BA << " SZANSA " << BA << " " << BA << " " << BA << "\n"

<< C8; w(CD, 10); cout << CA; w(CD, 4); cout << CA; w(CD, 4); cout << BC << "\t" << C8; w(CD, 10); cout << CA; w(CD, 4); cout << CA; w(CD, 4); cout << BC << "\n"

<< "\n"

<< C9; w(CD, 10); cout << CB << CD << "P1" << CD << CB << CD << "P2" << CD << BB << "\n"

<< BA << " RAZEM " << BA << " " << BA << " " << BA << "\n"

<< C8; w(CD,10); cout << CA; w(CD, 4); cout << CA; w(CD, 4); cout << BC << "\n"

<< "\n"

<< "Kategorie prawej cz\xA9\x98\ci tabeli\n"

<< "\n"

<< "3 JEDN. - suma oczek wszystkich pi\xA9\ciu kostek, je\x98li wyst\xA9puj\xA5 co najmniej trzy jednakowe liczby oczek\n"

<< "4 JEDN. - suma oczek wszystkich pi\xA9\ciu kostek, je\x98li wyst\xA9puj\xA5 co najmniej cztery jednakowe liczby oczek\n"

<< "FULL - 25 pkt., je\x98li w uk\x88\x61\dzie wyst\xA9puj\xA5 trzy jednakowe liczby oczek oraz dwie inne jednakowe\n"

<< "MA\x9DY STRIT - 30 pkt., je\x98li przynajmniej cztery ko\x98\ci maj\xA5 kolejne liczby oczek\n"

<< "DU\x8DY STRIT - 40 pkt., je\x98li wszystkie ko\x98\ci maj\xA5 kolejne liczby oczek\n"

<< "GENERA\x9D - 50 pkt., je\x98li liczba oczek na wszystkich kostkach jest jednakowa\n"

<< "SZANSA - niezale\xBEnie od uk\x88\x61\du kostek uzyskuje si\xA9 sum\xA9 oczek z wszystkich kostek\n"

<< "\n"

<< "\n"

<< "\n"

<< "Strona 2/3 - naci\x98nij enter aby przej\x98\x86 dalej";

cin.clear(); cin.sync(); cin.get();

ClearScreen();

cout << "Wyboru ka\xBE\dej z wolnych kategorii mo\xBEna dokona\x86 niezale\xBEnie od posiadanego uk\x88\x61\du, przy czym je\x98li nie spe\x88nia on\n"

<< "okre\x98lonych wymaga\xE4, to otrzymuje si\xA9 0 pkt. (taki wyb\xA2r za 0 pkt. jest czasami konieczny, kiedy uk\x88\x61\d nie pasuje\n"

<< "do \xBE\x61\dnej z wolnych kategorii).\n"

<< "\n"

<< "Drugi, trzeci i ka\xBE\dy kolejny wyrzucony genera\x88 mo\xBE\x65 zosta\x86 u\xBEyty jako joker, je\x98li kategoria \"GENERA\x9D\" i kategoria\n"

<< "lewej cz\xA9\x98\ci tabeli odpowiadaj\xA5\ca danemu genera\x88owi s\xA5 ju\xBE wykorzystane. Jokera zapisuje si\xA9 w prawych kategoriach\n"

<< "tabeli punktacji, otrzymuj\xA5\c stosown\xA5 liczb\xA9 punkt\xA2w (np. 25 pkt. za \"FULL\" itd.), chyba \xBE\x65 wszystkie prawe kategorie\n"

<< "s\xA5 ju\xBE wykorzystane, wtedy trzeba go zapisa\x86 do jednej z lewych za 0 pkt. Je\xBE\x65li joker zosta\x88 u\xBEyty co najmniej raz, to\n"

<< "otrzymuje si\xA9 100 pkt. premii.\n"

<< "\n"

<< "\n"

<< "Strona 3/3 - naci\x98nij enter aby wr\xA2\ci\x86 do menu";

cin.clear(); cin.sync(); cin.get();

}

void Historia()

{

static constexpr short P1 = 0,

P2 = 1,

jedynki = 0, dwojki = 1, trojki = 2, czworki = 3, piatki = 4, szostki = 5,

trzy\_jedn = 6, cztery\_jedn = 7, full = 8, maly\_strit = 9, duzy\_strit = 10, general = 11, szansa = 12,

premia = 13, razem = 14;

short odczyt\_pkt[2][15];

ifstream odczyt\_podglad;

odczyt\_podglad.open("lista\_gier.txt", ios::in);

if(odczyt\_podglad.good())

{

string linia, token;

int opcja, index = 0, i = 0;

ClearScreen();

while(!odczyt\_podglad.eof())

{

getline(odczyt\_podglad, linia);

istringstream iss(linia);

while(getline(iss, token, ','))

{

switch(i)

{

case 0:

cout << "[" << index+1 << "]\t"

<< setw(10) << token << " ";

break;

case 1:

cout << setw(3) << token << "pkt. : ";

break;

case 2:

cout << setw(3) << token << "pkt.";

break;

case 3:

cout << " " << token << " ";

break;

}

i++;

}

cout << "\n";

i = 0;

index++;

}

// WYBOR OPCJI

cout << "[0] Wyj\x98\cie\n";

cout << "Wpisz numer gry i zatwierd\xAB enterem: ";

cin >> opcja;

while (cin.fail() || opcja < 0 || opcja > index-1)

{

cin.clear();

cin.ignore(INT\_MAX, '\n');

cout << "B\x88\x61\d!" << endl;

cout << "Wpisz numer gry i zatwierd\xAB enterem: ";

cin >> opcja;

}

if(opcja == 0) return;

// ------------

ifstream odczyt\_szczegoly;

odczyt\_szczegoly.open("szczegoly\_gier.txt", ios:: in);

if(odczyt\_szczegoly.good())

{

string linia;

ClearScreen();

for(int i = 0; i < opcja; i++) getline(odczyt\_szczegoly, linia);

istringstream iss(linia);

int gracz = 0, kategoria = 0;

while(getline(iss, token, ','))

{

if(kategoria < 15)

{

odczyt\_pkt[gracz][kategoria] = stoi(token);

kategoria++;

if(kategoria >= 15)

{

kategoria = 0;

gracz++;

}

}

}

cout << C9; w(CD, 10); cout << CB << CD << "P1" << CD << CB << CD << "P2" << CD << BB << "\t" << C9; w(CD, 10); cout << CB << CD << "P1" << CD << CB << CD << "P2" << CD << BB << "\n"

<< BA << " JEDYNKI " << BA << setw(3) << odczyt\_pkt[P1][jedynki] << " " << BA << setw(3) << odczyt\_pkt[P2][jedynki] << " " << BA << "\t" << BA << " 3 JEDN. " << BA << setw(3) << odczyt\_pkt[P1][trzy\_jedn] << " " << BA << setw(3) << odczyt\_pkt[P2][trzy\_jedn] << " " << BA << "\n"

<< BA << " DW\xE0JKI " << BA << setw(3) << odczyt\_pkt[P1][dwojki] << " " << BA << setw(3) << odczyt\_pkt[P2][dwojki] << " " << BA << "\t" << BA << " 4 JEDN. " << BA << setw(3) << odczyt\_pkt[P1][cztery\_jedn] << " " << BA << setw(3) << odczyt\_pkt[P2][cztery\_jedn] << " " << BA << "\n"

<< BA << " TR\xE0JKI " << BA << setw(3) << odczyt\_pkt[P1][trojki] << " " << BA << setw(3) << odczyt\_pkt[P2][trojki] << " " << BA << "\t" << BA << " FULL " << BA << setw(3) << odczyt\_pkt[P1][full] << " " << BA << setw(3) << odczyt\_pkt[P2][full] << " " << BA << "\n"

<< BA << " CZW\xE0RKI " << BA << setw(3) << odczyt\_pkt[P1][czworki] << " " << BA << setw(3) << odczyt\_pkt[P2][czworki] << " " << BA << "\t" << BA << "MA\x9DY STRIT" << BA << setw(3) << odczyt\_pkt[P1][maly\_strit] << " " << BA << setw(3) << odczyt\_pkt[P2][maly\_strit] << " " << BA << "\n"

<< BA << " PI\xA4TKI " << BA << setw(3) << odczyt\_pkt[P1][piatki] << " " << BA << setw(3) << odczyt\_pkt[P2][piatki] << " " << BA << "\t" << BA << "DU\xBDY STRIT" << BA << setw(3) << odczyt\_pkt[P1][duzy\_strit] << " " << BA << setw(3) << odczyt\_pkt[P2][duzy\_strit] << " " << BA << "\n"

<< BA << " SZ\xE0STKI " << BA << setw(3) << odczyt\_pkt[P1][szostki] << " " << BA << setw(3) << odczyt\_pkt[P2][szostki] << " " << BA << "\t" << BA << " GENERA\x9D " << BA << setw(3) << odczyt\_pkt[P1][general] << " " << BA << setw(3) << odczyt\_pkt[P2][general] << " " << BA << "\n"

<< BA << " PREMIA " << BA << setw(3) << odczyt\_pkt[P1][premia] << " " << BA << setw(3) << odczyt\_pkt[P2][premia] << " " << BA << "\t" << BA << " SZANSA " << BA << setw(3) << odczyt\_pkt[P1][szansa] << " " << BA << setw(3) << odczyt\_pkt[P2][szansa] << " " << BA << "\n"

<< C8; w(CD, 10); cout << CA; w(CD, 4); cout << CA; w(CD, 4); cout << BC << "\t" << C8; w(CD, 10); cout << CA; w(CD, 4); cout << CA; w(CD, 4); cout << BC << "\n"

<< "\n"

<< C9; w(CD, 10); cout << CB << CD << "P1" << CD << CB << CD << "P2" << CD << BB << "\n"

<< BA << " RAZEM " << BA << setw(3) << odczyt\_pkt[P1][razem] << " " << BA << setw(3) << odczyt\_pkt[P2][razem] << " " << BA << "\n"

<< C8; w(CD,10); cout << CA; w(CD, 4); cout << CA; w(CD, 4); cout << BC << "\n"

<< "Naci\x98nij enter aby wr\xA2\ci\x86 do menu" << endl;

cin.clear(); cin.sync(); cin.get();

odczyt\_szczegoly.close();

}

else

{

cout << "\n\n\n\n\n"

<< "B\x88\xA5\d!\n"

<< "Plik \"szczegoly\_gier.txt\" nie istnieje lub jest pusty. Je\xBE\x65li nie posiadasz kopii tego pliku,\n"

<< "zaleca si\xA9 usuni\xA9\cie \"lista\_gier.txt\" oraz \"szczegoly\_gier.txt\" aby naprawi\x86 problem.\n"

<< "UWAGA - utracisz dotychczasow\xA5 histori\xA9 gier!\n"

<< "Naci\x98nij enter aby wr\xA2\ci\x86 do menu" << endl;

cin.clear(); cin.sync(); cin.get();

}

odczyt\_podglad.close();

}

else

{

cout << "\n\n\n\n\nBrak historii rozegranych gier.\n"

<< "Naci\x98nij enter aby kontynuowa\x86" << endl;

cin.clear(); cin.sync(); cin.get();

}

}

void WyswietlMenu()

{

cout << C9; w(CD, 21); cout << BB << "\n"

<< BA << " Ko\x98\ci " << BA << "\n"

<< CC; w(CD, 21); cout << B9 << "\n"

<< BA << " [1] Rozpocznij gr\xA9 " << BA << "\n"

<< BA << " [2] Historia meczy " << BA << "\n"

<< BA << " [3] Zasady " << BA << "\n"

<< BA << " [4] Wyj\x98\cie " << BA << "\n"

<< CC; w(CD, 21); cout << B9 << "\n"

<< BA << " Wykona\x88 Igor Kucyk " << BA << "\n"

<< C8; w(CD, 21); cout << BC << "\n";

}

**operacje\_na\_consoli.h**

#ifndef OPERACJE\_NA\_CONSOLI\_H\_INCLUDED

#define OPERACJE\_NA\_CONSOLI\_H\_INCLUDED

// Funkcja zapozyczona z internetu, gdyz system() jest powszechnie uznawany za zly

// Wyjasnienie: http://www.cplusplus.com/articles/j3wTURfi/

// Zrodlo funkcji: https://stackoverflow.com/questions/6486289/how-can-i-clear-console

void ClearScreen();

#endif // OPERACJE\_NA\_CONSOLI\_H\_INCLUDED

**operacje\_na\_consoli.cpp**

#include <iostream>

#include <windows.h>

#include "operacje\_na\_consoli.h"

using namespace std;

void ClearScreen()

{

HANDLE hStdOut;

CONSOLE\_SCREEN\_BUFFER\_INFO csbi;

DWORD count;

DWORD cellCount;

COORD homeCoords = { 0, 0 };

hStdOut = GetStdHandle( STD\_OUTPUT\_HANDLE );

if (hStdOut == INVALID\_HANDLE\_VALUE) return;

/\* Get the number of cells in the current buffer \*/

if (!GetConsoleScreenBufferInfo( hStdOut, &csbi )) return;

cellCount = csbi.dwSize.X \*csbi.dwSize.Y;

/\* Fill the entire buffer with spaces \*/

if (!FillConsoleOutputCharacter(

hStdOut,

(TCHAR) ' ',

cellCount,

homeCoords,

&count

)) return;

/\* Fill the entire buffer with the current colors and attributes \*/

if (!FillConsoleOutputAttribute(

hStdOut,

csbi.wAttributes,

cellCount,

homeCoords,

&count

)) return;

/\* Move the cursor home \*/

SetConsoleCursorPosition( hStdOut, homeCoords );

}

**znaki\_specjalne.h**

#ifndef ZNAKI\_SPECJALNE\_H\_INCLUDED

#define ZNAKI\_SPECJALNE\_H\_INCLUDED

// Dwuliniowe (do menu)

extern const std::string B9, // BOK z lewa odnoga

BA, // BOK

BB, // prawy gorny ROG

BC, // prawy dolny ROG

C8, // lewy dolny ROG

C9, // lewy gorny ROG

CA, // DOL z odnoga

CB, // GORA z odnoga

CC, // BOK z prawa odnoga

CD, // GORA/DOL

CE, // KRZYZ

// Jednoliniowe (do kosci)

B3, // BOK

BF, // prawy gorny ROG

C0, // lewy dolny ROG

C4, // GORA/DOL

D9, // prawy dolny rog

DA; // lewy gorny rog

void w(std::string, short);

#endif // ZNAKI\_SPECJALNE\_H\_INCLUDED

**znaki\_specjalne.cpp**

#include <iostream>

#include "znaki\_specjalne.h"

using namespace std;

// Dwukreskowe (do menu)

const string B9 = "\xB9", // BOK z lewa odnoga

BA = "\xBA", // BOK

BB = "\xBB", // prawy gorny ROG

BC = "\xBC", // prawy dolny ROG

C8 = "\xC8", // lewy dolny ROG

C9 = "\xC9", // lewy gorny ROG

CA = "\xCA", // DOL z odnoga

CB = "\xCB", // GORA z odnoga

CC = "\xCC", // BOK z prawa odnoga

CD = "\xCD", // GORA/DOL

CE = "\xCE", // KRZYZ

// Jednokreskowe (do kosci)

B3 = "\xB3", // BOK

BF = "\xBF", // prawy gorny ROG

C0 = "\xC0", // lewy dolny ROG

C4 = "\xC4", // GORA/DOL

D9 = "\xD9", // prawy dolny ROG

DA = "\xDA"; // lewy gorny ROG

void w(string znak, short il = 1)

{

for(int i = 0; i < il; i++)

cout << znak;

}

**kosci.h**

#ifndef KOSCI\_H\_INCLUDED

#define KOSCI\_H\_INCLUDED

void Gra();

class TKosci

{

private:

char wybor;

static constexpr short P1 = 0,

P2 = 1,

jedynki = 0, dwojki = 1, trojki = 2, czworki = 3, piatki = 4, szostki = 5,

trzy\_jedn = 6, cztery\_jedn = 7, full = 8, maly\_strit = 9, duzy\_strit = 10, general = 11, szansa = 12,

premia = 13, razem = 14;

short tura = P1,

ilosc\_rzutow = 1,

kosci[5],

punktacja[2][15] = {},

ktora = 1; // Ktora runda, sluzy jako index do 'runda'

bool czy\_zakonczyc\_ture = false,

czy\_podpowiedz = false,

czy\_zatrzymane[5] = {},

czy\_wybrane[2][14] = {},

czy\_joker = false,

czy\_uzyto\_jokera[2] = {},

czy\_przyznano\_premie\_jokera[2] = {};

std::string runda[13] = {"I","II","III","IV","V","VI","VII","VIII","IX","X","XI","XII","XIII"};

std::string DrukujKosc(short, short);

std::string P(short gracz, short kategoria){ // Funkcja wyswietlajaca zawartosc zmiennej 'punktacja' dla konkretnego gracza i kategorii

if(kategoria == razem)

return std::to\_string(punktacja[gracz][razem]) + " ";

if(czy\_wybrane[gracz][kategoria])

return std::to\_string(punktacja[gracz][kategoria]) + " ";

return "";

}

void LosujKosci(){

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

if(!czy\_zatrzymane[i])

kosci[i] = rand() % 6 + 1;

}

short IloscPunktow(const short, bool = false, bool = true);

void CzyJoker(){

if(czy\_wybrane[tura][general] && punktacja[tura][general] != 0)

{

short ilosc\_oczek = kosci[0];

for(int i = 1; i < 5; i++)

if(kosci[i] != (int)ilosc\_oczek)

{

czy\_joker = false;

return;

}

czy\_joker = true;

}

else

czy\_joker = false;

}

public:

TKosci(){LosujKosci();}

bool CzyKoniec() {return ktora > 13;}

void WyswietlPlansze();

void WyswietlKosci();

void Opcje();

void CzyKoniecTury(){

if(czy\_zakonczyc\_ture)

{

if(tura == P1) tura = P2;

else{tura = P1; ktora++;}

ilosc\_rzutow = 1;

for(int i = 0; i < 5; i++) czy\_zatrzymane[i] = false;

LosujKosci();

wybor = 0;

czy\_zakonczyc\_ture = false;

}

}

void Zapisz();

};

#endif // KOSCI\_H\_INCLUDED

**kosci.cpp**

#include <iostream>

#include <conio.h>

#include <time.h>

#include <stdlib.h>

#include <algorithm>

#include <iomanip>

#include <unordered\_set>

#include <fstream>

#include "kosci.h"

#include "operacje\_na\_consoli.h"

#include "znaki\_specjalne.h"

using namespace std;

void Gra()

{

srand(time(NULL));

TKosci Kosci;

while(!Kosci.CzyKoniec())

{

ClearScreen();

Kosci.WyswietlPlansze();

Kosci.WyswietlKosci();

Kosci.Opcje();

Kosci.CzyKoniecTury();

}

Kosci.Zapisz();

}

void TKosci::WyswietlPlansze()

{

cout << C9; w(CD, 10); cout << CB << CD << "P1" << CD << CB << CD << "P2" << CD << BB << "\t" << C9; w(CD, 10); cout << CB << CD << "P1" << CD << CB << CD << "P2" << CD << BB << "\n"

<< BA << " JEDYNKI " << BA << setw(4) << P(P1,jedynki) << BA << setw(4) << P(P2,jedynki) << BA << "\t" << BA << " 3 JEDN. " << BA << setw(4) << P(P1,trzy\_jedn) << BA << setw(4) << P(P2,trzy\_jedn) << BA << "\n"

<< BA << " DW\xE0JKI " << BA << setw(4) << P(P1,dwojki) << BA << setw(4) << P(P2,dwojki) << BA << "\t" << BA << " 4 JEDN. " << BA << setw(4) << P(P1,cztery\_jedn) << BA << setw(4) << P(P2,cztery\_jedn) << BA << "\n"

<< BA << " TR\xE0JKI " << BA << setw(4) << P(P1,trojki) << BA << setw(4) << P(P2,trojki) << BA << "\t" << BA << " FULL " << BA << setw(4) << P(P1,full) << BA << setw(4) << P(P2,full) << BA << "\n"

<< BA << " CZW\xE0RKI " << BA << setw(4) << P(P1,czworki) << BA << setw(4) << P(P2,czworki) << BA << "\t" << BA << "MA\x9DY STRIT" << BA << setw(4) << P(P1,maly\_strit) << BA << setw(4) << P(P2,maly\_strit) << BA << "\n"

<< BA << " PI\xA4TKI " << BA << setw(4) << P(P1,piatki) << BA << setw(4) << P(P2,piatki) << BA << "\t" << BA << "DU\xBDY STRIT" << BA << setw(4) << P(P1,duzy\_strit) << BA << setw(4) << P(P2,duzy\_strit) << BA << "\n"

<< BA << " SZ\xE0STKI " << BA << setw(4) << P(P1,szostki) << BA << setw(4) << P(P2,szostki) << BA << "\t" << BA << " GENERA\x9D " << BA << setw(4) << P(P1,general) << BA << setw(4) << P(P2,general) << BA << "\n"

<< BA << " PREMIA " << BA << setw(4) << P(P1,premia) << BA << setw(4) << P(P2,premia) << BA << "\t" << BA << " SZANSA " << BA << setw(4) << P(P1,szansa) << BA << setw(4) << P(P2,szansa) << BA << "\n"

<< C8; w(CD, 10); cout << CA; w(CD, 4); cout << CA; w(CD, 4); cout << BC << "\t" << C8; w(CD, 10); cout << CA; w(CD, 4); cout << CA; w(CD, 4); cout << BC << "\n"

<< "\n"

<< C9; w(CD, 10); cout << CB << CD << "P1" << CD << CB << CD << "P2" << CD << BB << "\t" << "\xAF Runda" << setw(6) << runda[ktora-1] << " \xAE\n"

<< BA << " RAZEM " << BA << setw(4) << P(P1,razem) << BA << setw(4) << P(P2,razem) << BA << "\t" << "\xAF Tura" << setw(6) << "P" << tura+1 << " \xAE\n"

<< C8; w(CD,10); cout << CA; w(CD, 4); cout << CA; w(CD, 4); cout << BC << "\t" << "\xAF Rzut" << setw(7) << ilosc\_rzutow << " \xAE\n";

}

string TKosci::DrukujKosc(short kosc, short wiersz)

{

switch(kosc)

{

case 1:

{

string jeden[5] = {DA + C4 + C4 + C4 + C4 + C4 + BF,

B3 + " " + B3,

B3 + " o " + B3,

B3 + " " + B3,

C0 + C4 + C4 + C4 + C4 + C4 + D9};

return jeden[wiersz];

}

case 2:

{

string dwa[5] = {DA + C4 + C4 + C4 + C4 + C4 + BF,

B3 + " o " + B3,

B3 + " " + B3,

B3 + " o " + B3,

C0 + C4 + C4 + C4 + C4 + C4 + D9};

return dwa[wiersz];

}

case 3:

{

string trzy[5] = {DA + C4 + C4 + C4 + C4 + C4 + BF,

B3 + " o " + B3,

B3 + " o " + B3,

B3 + " o " + B3,

C0 + C4 + C4 + C4 + C4 + C4 + D9};

return trzy[wiersz];

}

case 4:

{

string cztery[5] = {DA + C4 + C4 + C4 + C4 + C4 + BF,

B3 + " o o " + B3,

B3 + " " + B3,

B3 + " o o " + B3,

C0 + C4 + C4 + C4 + C4 + C4 + D9};

return cztery[wiersz];

}

case 5:

{

string piec[5] = {DA + C4 + C4 + C4 + C4 + C4 + BF,

B3 + " o o " + B3,

B3 + " o " + B3,

B3 + " o o " + B3,

C0 + C4 + C4 + C4 + C4 + C4 + D9};

return piec[wiersz];

}

case 6:

{

string szesc[5] = {DA + C4 + C4 + C4 + C4 + C4 + BF,

B3 + " o o " + B3,

B3 + " o o " + B3,

B3 + " o o " + B3,

C0 + C4 + C4 + C4 + C4 + C4 + D9};

return szesc[wiersz];

}

}

}

void TKosci::WyswietlKosci()

{

cout << DrukujKosc(kosci[0], 0) << "\t" << DrukujKosc(kosci[1], 0) << "\t" << DrukujKosc(kosci[2], 0) << "\t" << DrukujKosc(kosci[3], 0) << "\t" << DrukujKosc(kosci[4], 0) << "\n"

<< DrukujKosc(kosci[0], 1) << "\t" << DrukujKosc(kosci[1], 1) << "\t" << DrukujKosc(kosci[2], 1) << "\t" << DrukujKosc(kosci[3], 1) << "\t" << DrukujKosc(kosci[4], 1) << "\n"

<< DrukujKosc(kosci[0], 2) << "\t" << DrukujKosc(kosci[1], 2) << "\t" << DrukujKosc(kosci[2], 2) << "\t" << DrukujKosc(kosci[3], 2) << "\t" << DrukujKosc(kosci[4], 2) << "\n"

<< DrukujKosc(kosci[0], 3) << "\t" << DrukujKosc(kosci[1], 3) << "\t" << DrukujKosc(kosci[2], 3) << "\t" << DrukujKosc(kosci[3], 3) << "\t" << DrukujKosc(kosci[4], 3) << "\n"

<< DrukujKosc(kosci[0], 4) << "\t" << DrukujKosc(kosci[1], 4) << "\t" << DrukujKosc(kosci[2], 4) << "\t" << DrukujKosc(kosci[3], 4) << "\t" << DrukujKosc(kosci[4], 4) << endl;

}

void TKosci::Opcje()

{

CzyJoker();

if(ilosc\_rzutow < 3 && wybor != 'b')

{

cout << "Zmiana stanu ko\x98\ci:\tOpcje:\n";

cout << "[1] "; czy\_zatrzymane[0] ? cout << "Zatrzymane\t\t" : cout << "Niezatrzymane\t"; cout << "[A] Rzu\x86 ponownie\n";

cout << "[2] "; czy\_zatrzymane[1] ? cout << "Zatrzymane\t\t" : cout << "Niezatrzymane\t"; cout << "[B] Wybierz kategori\xA9\n";

cout << "[3] "; czy\_zatrzymane[2] ? cout << "Zatrzymane\n" : cout << "Niezatrzymane\n";

cout << "[4] "; czy\_zatrzymane[3] ? cout << "Zatrzymane\n" : cout << "Niezatrzymane\n";

cout << "[5] "; czy\_zatrzymane[4] ? cout << "Zatrzymane\n" : cout << "Niezatrzymane" << endl;

do

{

wybor = getch();

if(wybor == 'A' || wybor == 'B') wybor += 32;

switch(wybor)

{

case '1': czy\_zatrzymane[0] ? czy\_zatrzymane[0] = false : czy\_zatrzymane[0] = true; return;

case '2': czy\_zatrzymane[1] ? czy\_zatrzymane[1] = false : czy\_zatrzymane[1] = true; return;

case '3': czy\_zatrzymane[2] ? czy\_zatrzymane[2] = false : czy\_zatrzymane[2] = true; return;

case '4': czy\_zatrzymane[3] ? czy\_zatrzymane[3] = false : czy\_zatrzymane[3] = true; return;

case '5': czy\_zatrzymane[4] ? czy\_zatrzymane[4] = false : czy\_zatrzymane[4] = true; return;

case 'a': LosujKosci(); ilosc\_rzutow++; return;

case 'b': return;

}

} while(!((int)wybor >= 49 && (int)wybor <= 53)

|| wybor != 'a' || wybor != 'b');

}

else

{

if(!czy\_wybrane[tura][jedynki]) cout << "[A] JEDYNKI\t\t" << setw(3) << IloscPunktow(jedynki) << " pkt.\n";

if(!czy\_wybrane[tura][dwojki]) cout << "[B] DW\xE0JKI\t\t" << setw(3) << IloscPunktow(dwojki) << " pkt.\n";

if(!czy\_wybrane[tura][trojki]) cout << "[C] TR\xE0JKI\t\t" << setw(3) << IloscPunktow(trojki) << " pkt.\n";

if(!czy\_wybrane[tura][czworki]) cout << "[D] CZW\xE0RKI\t\t" << setw(3) << IloscPunktow(czworki) << " pkt.\n";

if(!czy\_wybrane[tura][piatki]) cout << "[E] PI\xA4TKI\t\t" << setw(3) << IloscPunktow(piatki) << " pkt.\n";

if(!czy\_wybrane[tura][szostki]) cout << "[F] SZ\xE0STKI\t\t" << setw(3) << IloscPunktow(szostki) << " pkt.\n";

if(!czy\_wybrane[tura][trzy\_jedn]) cout << "[G] TRZY JEDNAKOWE\t" << setw(3) << IloscPunktow(trzy\_jedn) << " pkt.\n";

if(!czy\_wybrane[tura][cztery\_jedn]) cout << "[H] CZTERY JEDNAKOWE\t" << setw(3) << IloscPunktow(cztery\_jedn) << " pkt.\n";

if(!czy\_wybrane[tura][full]) cout << "[I] FULL\t\t" << setw(3) << IloscPunktow(full, czy\_joker) << " pkt.\n";

if(!czy\_wybrane[tura][maly\_strit]) cout << "[J] MA\x9DY STRIT\t\t" << setw(3) << IloscPunktow(maly\_strit, czy\_joker) << " pkt.\n";

if(!czy\_wybrane[tura][duzy\_strit]) cout << "[K] DU\xBDY STRIT\t\t" << setw(3) << IloscPunktow(duzy\_strit, czy\_joker) << " pkt.\n";

if(!czy\_wybrane[tura][general]) cout << "[L] GENERA\x9D\t\t" << setw(3) << IloscPunktow(general) << " pkt.\n";

if(!czy\_wybrane[tura][szansa]) cout << "[M] SZANSA\t\t" << setw(3) << IloscPunktow(szansa) << " pkt.\n";

cout << endl;

do

{

wybor = getch();

if((int)wybor >= 65 && (int)wybor <= 90) wybor += 32;

//cout << wybor << " " << (int)wybor-97 << endl; system("pause");

} while (!((int)wybor >= 97 && (int)wybor <= 122) || czy\_wybrane[tura][(int)wybor-97]);

short kategoria\_temp = (int)wybor-97;

punktacja[tura][kategoria\_temp] = IloscPunktow(kategoria\_temp, czy\_joker, czy\_podpowiedz);

punktacja[tura][razem] += IloscPunktow(kategoria\_temp, czy\_joker, czy\_podpowiedz);

czy\_wybrane[tura][kategoria\_temp] = true;

if(czy\_wybrane[tura][jedynki] && czy\_wybrane[tura][dwojki] && czy\_wybrane[tura][trojki] &&

czy\_wybrane[tura][czworki] && czy\_wybrane[tura][piatki] && czy\_wybrane[tura][szostki])

{

short suma = 0;

for(int i = 0; i < 6; i++)

suma += punktacja[tura][i];

if(suma >= 63 && !czy\_wybrane[tura][premia])

{

punktacja[tura][premia] += 35;

punktacja[tura][razem] += 35;

czy\_wybrane[tura][premia] = true;

}

}

if(czy\_uzyto\_jokera[tura] && !czy\_przyznano\_premie\_jokera[tura])

{

punktacja[tura][premia] += 100;

punktacja[tura][razem] += 100;

czy\_przyznano\_premie\_jokera[tura] = true;

czy\_wybrane[tura][premia] = true;

}

czy\_zakonczyc\_ture = true;

}

}

short TKosci::IloscPunktow(const short kategoria, bool joker, bool czy\_podpowiedz)

{

vector<short> k = {kosci[0], kosci[1], kosci[2], kosci[3], kosci[4]};

sort(k.begin(), k.end());

short suma = 0;

bool czy\_prawda = false;

switch(kategoria)

{

case trzy\_jedn:

{

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

if(k[i] == k[i+1] && k[i+1] == k[i+2])

czy\_prawda = true;

if(czy\_prawda)

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

suma += k[i];

return suma;

}

case cztery\_jedn:

{

for(int i = 0; i < 2; i++)

if(k[i] == k[i+1] && k[i+1] == k[i+2] && k[i+2] == k[i+3])

czy\_prawda = true;

if(czy\_prawda)

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

suma += k[i];

return suma;

}

case full:

{

if(joker)

{

if(czy\_podpowiedz)

return 25;

czy\_uzyto\_jokera[tura] = true;

return 25;

}

if((k[0] == k[1] && k[1] == k[2]

&& k[3] == k[4])

||

(k[0] == k[1]

&& k[2] == k[3] && k[3] == k[4]))

return 25;

else return 0;

}

case maly\_strit:

{

if(joker)

{

if(czy\_podpowiedz)

return 30;

czy\_uzyto\_jokera[tura] = true;

return 30;

}

unordered\_set<short> temp;

temp.insert(k.begin(), k.end());

while(!k.empty()) k.pop\_back();

for(const int&x: temp) k.push\_back(x);

short roznica = -6;

if(k[1] - k[0] == 1) roznica = 1;

else if(k[1] - k[0] == -1) roznica = -1;

for(int i = 0; i < 2; i++)

for(int j = i; j < i+3; j++)

if(k[j] != k[0] + j\*roznica)

return 0;

return 30;

}

case duzy\_strit:

{

if(joker)

{

if(czy\_podpowiedz)

return 40;

czy\_uzyto\_jokera[tura] = true;

return 40;

}

//k.erase(unique(k.begin(), k.end()));

short roznica = -6;

if(k[1] - k[0] == 1) roznica = 1;

else if(k[1] - k[0] == -1) roznica = -1;

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

if(k[i] != k[0] + i\*roznica)

return 0;

return 40;

}

case general:

{

short ilosc\_oczek = k[0];

for(int i = 1; i < 5; i++)

if(k[i] != ilosc\_oczek)

return 0;

return 50;

}

case szansa:

{

for(int i = 0; i < 5; i+=1)

suma += k[i];

return suma;

}

default: // JEDYNKI, ..., SZOSTKI

{

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

if(k[i] == kategoria+1)

suma += kategoria+1;

return suma;

}

}

return suma;

}

void TKosci::Zapisz()

{

fstream zapis\_podglad;

string imieP1, imieP2;

ClearScreen();

cout << "Podaj imi\xA9 gracza P1: "; cin >> imieP1;

cout << "Podaj imi\xA9 gracza P2: "; cin >> imieP2;

zapis\_podglad.open("lista\_gier.txt", ios::app);

zapis\_podglad << imieP1 << "," << punktacja[P1][razem] << ","

<< punktacja[P2][razem] << "," << imieP2 << endl;

zapis\_podglad.close();

fstream zapis\_szczegoly;

zapis\_szczegoly.open("szczegoly\_gier.txt", ios::app);

for(int gracz = 0; gracz < 2; gracz++)

for(int kategoria = 0; kategoria < 15; kategoria++)

zapis\_szczegoly << punktacja[gracz][kategoria] << ",";

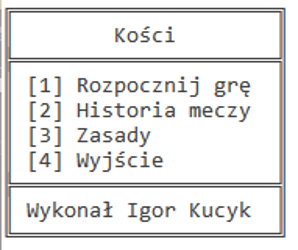
zapis\_szczegoly << endl;

zapis\_szczegoly.close();

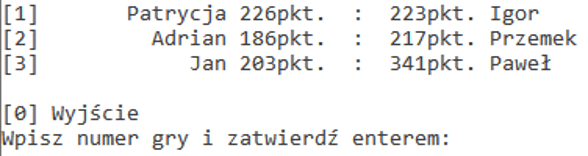
}

1. **Przykłady działania**

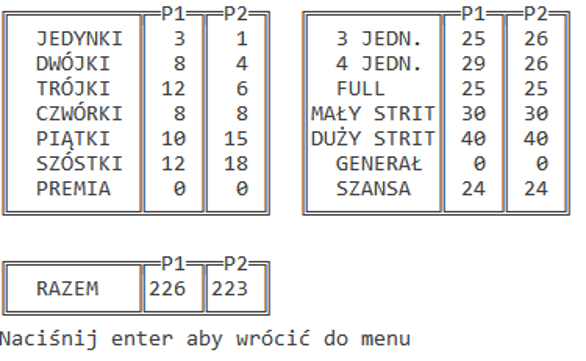
**1.**

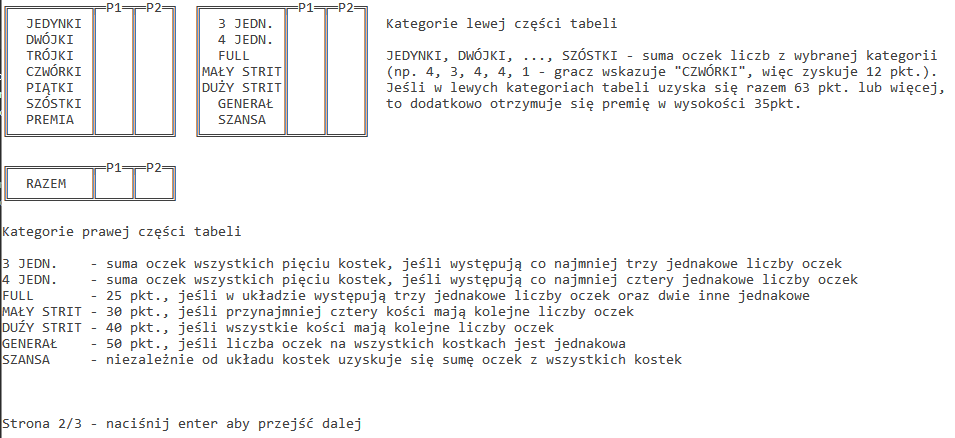
****

**2.**

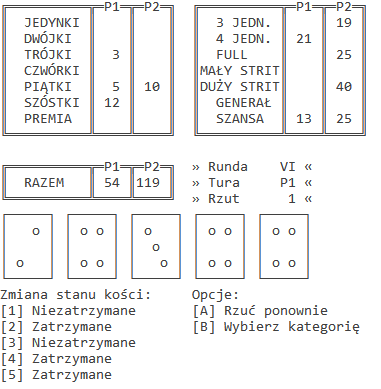
****

**3.**

****

**4. **

**5.**

****

**6.**

****