Ruletka

17 marca 2020

Ruletka jest grą hazardową polegającą na losowaniu jednej liczby znajdującej się na kole ruletki przy pomocy kulki. Na kole są wszystkie liczby od 1 do 36 oraz pola dodatkowe. Ruletka nazwana jest grą szatańską ponieważ $1 + 2 + 3 + ... + 34 + 35 + 36 = 666$.

Najprostsza gra polega na tym, że gracz wybiera jedną z liczb z koła ruletki. Po zebraniu zakładów krupier kręci kołem, w przeciwnym kierunku rzuca kulkę. Pole, na które upadła kulka, wygrywa. Jeśli gracz poprawnie obstawił numer, otrzymuje 35-krotność zakładu. Jeśli błędnie, traci założone pieniądze.
Należy obliczyć stopę zwrotu dla 3 odmian ruletki: uczciwej, europejskiej oraz amerykańskiej. Uczciwa ruletka ma tylko liczby od 1 do 36. Europejska ma dodatkowo 0, którego się nie obstawia. Odmiana amerykańska ma dwa pole zerowe, 0 i 00.
Stopę zwrotu będziemy rozumieć jako: (wygrana)/(liczba zakładów $\cdot$ stawka) $\cdot$100%.

W prostym podejściu do tego zadania należy napisać program, który będzie wczytywał ze strumienia albo z parametrów wywołania programu liczbę, którą

gracz wybiera. Liczba ta musi mieścić się w przedziale od 1 do 36, tylko takie pola można obstawiać.
Następnie program N-krotnie losuje liczbę od 1 do 36 w przypadku uczciwej ruletki, od 0 do 36 w przypadku ruletki europejskiej lub od 0 do 37 w przypadku ruletki amerykańskiej. Liczba 37 może być naszym substytutem pola 00.
Każde losowanie prowadzi do zmiany sumarycznego stanu wygranej gracza. Jeśli obstawi poprawnie, jego wygrana zwiększa się o 35·stawka, jeśli przegra, traci wartość obstawionej stawki, czyli sumaryczna wygrana zmniejsza się o wartość stawki.

Przypominam, że losowanie liczby całkowitej z zakresu od *min* do *max* można zrealizować dzięki bibliotece random w następujący sposób:

```
std::random_device rd;
std::default_random_engine engine(rd());
std::uniform_int_distribution<int> dist(min, max);
int randomNumber=dist(engine);
```

Zachęcam do rozwiązania zadania obiektowo. Możecie skorzystać z zalążka kodu, umieszczonego na githubie.
Kod zawiera funkcję główną (*main*), mieszczącą się w pliku *szatanskaGra.cpp*.
Funkcja główna ma za zadanie wczytać obstawianą liczbę z parametrów wywołania programu, czyli z argumentów int argc i char *argv[], sprawdzić, czy użytkownik podał parametr wejściowy oraz czy mieści się on w przedziale od 1 do 36.
Ponadto funkcja główna ustala liczbę zakładów oraz stawkę.
Następnie tworzy intancję odpowiedniej odmiany ruletki oraz uruchamia funkcję `zagraj`, która jest pusta i musi być napisana.
Funkcja zagraj przyjmuje następujące argumenty: wskaźnik do obiektu ruletka, zakład, czyli wybraną przez gracza liczbę, liczbę zakładów oraz stawkę. Funkcja musi wykonać odpowiednią liczbę gier, dla każdej sprawdzić, czy wylosowana liczba zgadza się z wybraną przez gracza i odpowiednio zmieniać wartość zmiennej *wygrana*. Na końcu funkcja zwraca zmienną *wygrana*.
Losowanie pola z koła ruletki jest realizowane przez klasę Ruletka. Spójrzmy na jej plik nagłówkowy (Ruletka.h):

Dygresja: W języku c++ każda klasa powinna posiadać dwa pliki - plik nagłówkowy o rozszerzeniu h, hpp lub hh i plik źródłowy o rozszerzeniu c, cpp, cxx. Pliki nagłówkowe zwyczajowo umieszcza się w katalogu include, pliki źródłowe w katalogu src. Nie jest to koniecznie, ale najczęściej tak się właśnie robi.

Plik nagłówkwy klasy Ruletka wygląda następująco:

```
#ifndef RULETKA_H
#define RULETKA_H
#include <vector>
#include <random>
```

```
//tu zaczyna się deklaracja klasy abstrakcyjnej Ruletka
class Ruletka
{
public:
    Ruletka(std::default_random_engine& engine);
    ~Ruletka();
    virtual int zakrec() = 0;
protected:
    std::default_random_engine myEngine;
};

//po klasie dziedziczą trzy inne klasy:

class Uczciwa: public Ruletka
{
public:
    Uczciwa(std::default_random_engine& engine);
    ~Uczciwa(){};
    //zwraca liczbę losową od 1 do 36
    virtual int zakrec();
};

class Amerykanska: public Ruletka
{
public:
    Amerykanska(std::default_random_engine& engine);
    ~Amerykanska(){};
    virtual int zakrec();
};

class Europejska: public Ruletka
{
public:
    Europejska(std::default_random_engine& engine);
    ~Europejska(){};
    virtual int zakrec();
};

#endif
```

Klasa Ruletka jest klasą abstrakcyjną, posiada czysto wirtualną metodę `virtual int zakrec() = 0;`. Po tej klasie dziedziczą trzy inne klasy Uczciwa, Europejska i Amerykańska. Każda z nich ma własną implementację metody zakręć. Funkcja zakręć dla Uczciwej ruletki powinna zwracać liczbę od 1 do 36, dla Europejskiej od 0 do 36, dla Amerykańskiej od 0 do 36, przy czym 0 ma być dwa razy częściej od pozostałych. Metoda `virtual int zakrec()` powinna zostać zaimplementowana w pliku źródłowym *Ruletka.cpp*. W plikach źródłowych implementacje wszystkich metod. Plik źródłowy zawsze musi mieć include do pliku nagłówkowego (`#include ''Ruletka.hh''`). Ponadto aby kompilator wiedział, że implementujemy funckję, która należy do jakieś klasy musimy podać pełną nazwę funkcji, czyli: `NazwaKlasy::NazwaFunkcji(argumenty)`.
Funkcja nazwana `NazwaKlasy::NazwaKlasy(argumenty)` to KONSTRUKTOR, czyli funkcja wywoływana podczas tworzenia instancji klasy.

Aktualnie metoda `zakrec()` nic nie robi i nic nie zwraca. Jej uzupełnienie zostawiam Państwu:

```
 #include "Ruletka.h"
Ruletka::Ruletka(std::default_random_engine& engine)
{
    myEngine = engine;
}

Ruletka::~Ruletka() {}

Uczciwa::Uczciwa(std::default_random_engine& engine): Ruletka(engine) {}

Europejska::Europejska(std::default_random_engine& engine): Ruletka(engine) {}

Amerykanska::Amerykanska(std::default_random_engine& engine): Ruletka(engine) {}

int Uczciwa::zakrec()
{

}

int Europejska::zakrec()
{

}

int Amerykanska::zakrec()
{

}
```

Proszę zwrócić uwagę na konstruktory klas Uczciwa, Europejska i Amerykańska:

```
Uczciwa::Uczciwa(std::default_random_engine& engine): Ruletka(engine) {}

Europejska::Europejska(std::default_random_engine& engine): Ruletka(engine) {}

Amerykanska::Amerykanska(std::default_random_engine& engine): Ruletka(engine) {}
```

Konstruktory są puste, nic nie robią z jednym wyjątkiem, wywołują konstruktor klasy bazowej Ruletka. Zawsze jeśli dziedziczym po jakiej klasie mamy obowiązek wywołać konstruktor klasy bazowej.