Dokumentacja Projektu na Labolatorium z Programowania Obiektowego

Kacper Białecki – 234947

Spis treści

[Opis programu 1](#_Toc485933091)

[Zmienne globalne 1](#_Toc485933092)

[Klasy 1](#_Toc485933093)

[TeamsAndOthers.h 3](#_Toc485933094)

[GlobalFunctions.h 3](#_Toc485933095)

[Initialize.h 3](#_Toc485933096)

[UpdateFunctions.h 4](#_Toc485933097)

[Source.cpp 4](#_Toc485933098)

# Opis programu

Program ma za zadanie proste symulowanie zarządzania imprezą esportową. Użytkownik wciela się w rolę zarządcy i jego zadaniem jest odpowiednie rozdyponowanie środków by wydarzenie się udało. W zależności od szczęścia oraz danych wprawadzanych przez osobę odpowiednio odbędzie się impreza. Przykładowo, gdy użytkownik preznaczy za mało srodków na zaplecze techniczne, możliwe jest że część sprzętu się zepsuje. Na koniec działania programu prezentowane są statystyki z rozgrywki oraz wynik punktowy.

# Zmienne globalne

* Double:
  + Wallet – stan konta
  + Happiness – zadowolenie gości
  + Technicians, prizeMoney, amenities – pieniądze, które są przeznaczone dla techników, nagrody i udogodnienia
  + Visitors – liczba osób, które przyszły na wydarzenie
  + Ticketprice, ticketsales – cena oraz sprzedaż biletów
* Int:
  + Duration – czas rozgrywki (domysnie 4 dni)
  + updateCounter – zlicza ile razy została wywołana funkcja update()
  + gameDay – liczy, który jest aktualnie dzień rozgrywek
  + brokenCameras, brokenComputers – ile razy zepsuł się komputer oraz kamera
  + mode – zapamiętuje wybrany poziom trudności

# Klasy

W programie występują klasy reprezentujące obiekty, na których odbywają się operacje.

* Venue – klasa odpowiadająca za lokalizację wydarzenia. Zawiera ona:
  + String location, name – odpowiednio nazwę lokalizacji oraz nazwę obiektu
  + Double price, capacity – przechowują wartości ceny oraz maksymalnej pojemności widzów
  + Int prestige – zawiera klasyfikację według której kalkulowana jest popularność obiektu. Przyjmuje wartości od 1 do 4, gdzie 4 to najlepsze.
  + Konstruktor – przyjmuje zmienne string (loc, nam), double (pri, cap) oraz int pre, wszytkie odpowiednio sa przekazywane zmiennym klasy.
  + Void showVenue() – wypisuje statystyki i informacje o obiekcie sportowym.
  + Double getPrice(), string getName(), int getPrestige(), double getCap() odpowiednio zwracają cenę, nazwę, prestiż oraz maksymalną pojemność.
  + Deklarowany jest także obiekt klasy Venue – actual – który nastepnie jest w późniejszczych partiach programu zamieniany na wybór użytkownika.
* Sponsor – klasa reprezentująca sponosra wydarzenia.
  + String name – nazwa sponsora.
  + Double moneyOffer, happinessModifier – pierwsza zmienna przechowuje wartość oferty pieniężnej sponosra a druga zawiera wartość o którą będzie kalkulowane zadowolenie widzów.
  + Konstruktor – podobnie do Venue przyjmuje odpowiednie wartości do zmiennych klasy.
  + Double getOffer(), string getName(), double getMod() – odpowiednio zwracają wartości zmiennych klasy.
  + Zadeklarowao również obiekt \_choice, który jest w późniejszczej częsći zastępowany przez sponsora wybranego przez uzytkownika.
* Booth – klasa odpowiadająca za dostępne budki wystawców. W zależności od tego, ile uzytkownik na nie wyda, tym lepsi wystawcy bedą na wydarzeniu.
  + String name – przechowuje nazwę
  + Bool isGamer – odpowiada za to, czy budka jest skierowana do graczy. Jeżeli jest, tym więcej zysku będzie generować.
  + Double cashOutput, likeable – pierwsza zmienna przechowuje wartość liczby, dzięki której generowany jest zysk. Likeable przechowuje wartość, odpowiadającą za to, czy ludzie lubią danego wystawcę.
  + Konstruktor – podobnie do innych klas przyjmuje wartości zmimennych i przypisuje je do zmiennych obiektu klasy.
  + Konstruktor domyślny – wymagany do zainicjalizowania tabeli wystawców.
  + String getName(), double getLike() – zwracają odpowiednio nazwę wystawcy oraz to jak bardzo jest lubiany.
  + Double getCash() – funkcja zwracająca wartość wygenerowanego zarobku. Zarobek zależy od ilości gości, tego jak bardzo wystawca jest lubiany oraz cashOutput. Dla wystawców skierowanych do graczy zarobek jest zwiększony trzykrotnie.
* FoodTruck – klasa dziedzicząca po Booth. Odpowiada za foodtrucki na wydarzeniu.
  + Od Booth różni się tylko konstruktorem, gdyż nie przyjmuje wartości isGamer. Foodtrucki są skierowane po prostu dla każdego.
* Team – klasa odpowiadająca za drużyny grające w rozgrywkach na imprezie. Drużyny zczytywane są z pliku a ich siła oraz popularność jest generowana losowo.
  + String name – nazwa drużyny.
  + Int fame, strenght – przechowują wartosci sławy oraz siły drużyny. Od nich zależy czy dana drużyna wygra mecz. Wartosci tych zmiennych dla obu walczących ze sobą drużyn są porównywane w dalszej części kodu. Sława przyjmuje wartości od 0 do 9 a siła od 0 do 99;
  + Konstruktor – przypisuje losowe wartości do fame i strenght.
  + Void setName(string newName) – nadaje drużynie nazwę. Jest to robione poza konstruktorem, żeby ułatwić tworzenie tablicy drużyn.
  + String getName(), int getFame(), int getStrenght() – zwracają odpowiednio wartości name, fame i strenght.

# TeamsAndOthers.h

Plik ten zawiera kod, w którym deklarowane są dostępne lokalizacje dla wydarzenia, tablica, która zawierać będzie drużyny oraz budki wystawców i sponsorów. Dzielone one są na kategorie, dzieki którym w zalezności od wydatków użytkownika takie budki będą na wydarzeniu. Dodatkowo plik zawiera dwie funkcje wypisujące wystawców i foodtrucki. Funkcje te wykorzystywane są w dalszej części kodu.

# GlobalFunctions.h

Plik zawierajacy zestaw funkcji używanych w wielu miejscach kodu oraz funkcję inicjalizującą drużyny. Funkcja ta zczytuje nazwy drużyn z pliku teams.txt i przydziela je do kolejnych obiektów tablicy. Int checkInput() odpowiada za sprawdzanie, czy użytkownik aby na pewno podał liczbę oraz prosi go o ponownie podanie, w przypadku kiedy np. podał literę. Void clr() po prostu czyści konsolę. Jest to zrobione w osobnej funkcji, żeby można było krócej zapisać „system(„cls”)”. String toUpper(string a) to funkcja zamieniająca przyjetego strina na same wielkie litery. Używana jest po to, by wielkosci liter nie miały znaczenia gdy użytkownik podaje np. nazwę obiektu sportowego.

# Initialize.h

Plik zawiera 3 funkcje, z czego 2 znich odpowiadają bezpośrednio za rozgrywkę. Bool venueChoice(Venue theVenue) sprawdza czy stać osobę na dany obiekt sportowy. Jeżeli tak, to odjemuje od portfela odpowiednią ilość pieniędzy oraz ustawia wybrany obiekt jako actual, czyli ten na którym będą wykonywane operacje. Void init() to funkcja rozpoczynająca symulację. Pyta użytkownika o poziom trudności oraz czas trwania symulacji (domyślnie 4 dni). Nastepnie jest funkcja void intro().

Void intro() to funkcja, w której użytkownik podaje wartości zmiennych oraz swoje wybory, które używane są w symulacji. Na początku wypisywane są obiekty sportowe oraz zapytanie o wybór. Kolejno dostajemy możliwość wyboru jednego z dwóch losowo wybranych sponsorów. Prezentowana jest nam również wysokość granta pieniężnego, lecz już nie popularność firmy. Użytkownik musi pomyśleć, czy wziąć mniej pieniędzy ale mieć sponsora lubianego przez ludzi, czy po prostu wybrać większą kwotę i się nie martwić. Następnie osoba stawiana jest przed wyborem ilości pieniędzy, jakie wyda na zaplecze techniczne, nagrody pieniężne oraz udogodnienia dla gości (wystawcy, jedzenie). Od wartości jakie poda użytkownik zależy jak często i czy w ogóle będzie się psuł sprzęt, ile osób przyjdzie na imprezę oraz jacy wystawcy poza sponsorem pojawią się na wydarzeniu. Na samym końcu użytkownik ma możliwość sprecyzowania ceny biletów. W polu tym może wpisać 0 żeby na imprezę był wstęp wolny.

# UpdateFunctions.h

Plik zawiera wszystkie funkcje wykorzystywane w funkcji update(). Każda odpowiada za odpowiedni element symulacji.

* Void attendanceCalc() – liczy ile przyszło osób na imprezę oraz ilość pieniędzy zarobionych na biletach.
* String score(Team a, Team b) – zwraca wynik meczu bo3 w zalezności od siły zwycięskiej drużyny.
* Void rng() – funkcja odpowiadająca za psucie się sprzętu. Może się popsuć komputer albo kamera. Cena naprawy/wymiany jest losową liczbą z przedziału 0 do 17 pomnożoną przez 1000. Dodatkowo jeżeli coś się zepsuje to zliczne jest ile razy.
* Team teamCompare(Team a, Team b) – funkcja porównująca siły dwóch drużyn walczących ze sobą. Silniejsza wygrywa. W przypadku gdy siły sa równe decyduje popularność.
* Team teamCompareLoser(Team a, Team b) – zasadniczo funkcja zwracająca przegraną drużynę.
* Void games(int day) – funkcja odpowiedzialna za mecze. Przyjmuje argument day, żeby zdeterminować, który dzień rozgrywek powinna symulować. Mecze odbywają się w formacie 8 : 4 : 2 : 1 meczy w 4 dni. Gdy symulacja trwa więcej niż 4 dni to mecze sa rozgrywane w ostatnie 4 dni wydarzenia. By znaleźć zwycięzcę danego meczu wykorzystywana jest funkcja teamCompare.
* Void \_happiness(int updateNo) - funkcja odpowiadająca za liczenie zadowolenia gości. Jest ono sumą zadowoleń zwracanych przez budki występujące na wydarzeniu oraz sponsora imprezy. updateNo jest tutaj by wyświetlić informację o opinii gości na temat sponsora tylko w pierwszy dzień symulacji.
* Void money() – funkcja zliczająca zarobki danego dnia. Sumuje ona wartości zwracane przez budki na wydarzeniu. Dodatkowo, jeżeli zadowolenie było niższe niż dzień wcześniej, ludzie zostawiają mniej pieniędzy.

# Source.cpp

Serce programu. Zawiera ono funkcję update(), która odpowiada za całą symulację, oraz int main(). Update() codziennie zlicza ile gości danego dnia przyszło i dodaje tą wartość o ogólnej puli. Podobnie zlicza ile razy dany obiekt sportowy był przepełniony. Przedstawia użytkownikowi dane oraz wyniki symulacji dzięki wywoływaniu funkcji z UpdateFunctions.h. Zawarte jest sprawdzenie czy przypadkiem użytkownik nie zbankrutował (jeśli tak to program się kończy). Wyświetlane są także rozgrywki i mecze grane danego dnia wraz z ich wynikami. Pod koniec dana jest możliwość wydania części zarobionych pieniędzy na techników, by w przypadku gdy psuje się sprzęt było to mniej prawdopodobne. Po ostatnim dniu (bądź bankructwie) prezentowane są użytkownikowi statystyki zbeirane podczas rozgrywki oraz finalna liczba uzyskanych punktów.  
 W funkcji main() kolejno sa wywoływane funkcje: init(), intro() oraz update() zapętlony tyle razy ile dni ma trwać symulacja.