|  |  |
| --- | --- |
|  | **ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ**  **ФАКУЛТЕТ КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ** |

**проект за оценка по СПр**

**Дисциплина: Системно програмиране**

**Тема: Магазинери „бъркат в касата“**

***Изготвил****:*

***Димитър Тодоров Колев***

Фак. № 381222063

Група: 91

IV курс, Киб. Сиг.

e-mail: [dimitakolev@tu-mail.bg](mailto:dimitakolev@tu-mail.bg)

***Ръководител****:*

Д. Андреев

**София, 2025**

Съдържание

[I. Анализ на изготвеното приложение 3](#_Toc193021633)

[1. Задание 3](#_Toc193021634)

[2. Анализ на заданието 3](#_Toc193021635)

[II. Функционално описание на приложението 3](#_Toc193021636)

[1. Инициализация на системата 3](#_Toc193021637)

[2. Обработка на операциите 3](#_Toc193021638)

[3. Обновяване на информацията в реално време 4](#_Toc193021639)

[4. Завършване и освобождаване на ресурси 4](#_Toc193021640)

[III. Изпълнение на функционалностите 4](#_Toc193021641)

[1. CashRegister 4](#_Toc193021642)

[2. Engine 5](#_Toc193021643)

[IV. Експериментални данни 5](#_Toc193021644)

[V. Приложение 7](#_Toc193021645)

# Анализ на изготвеното приложение

1. Задание

В един магазин служителите имали неприятния навик „да бъркат в касата“ и да взимат пари назаем. Непрекъснато някой взимал някаква сума пари, друг пък връщал. Моделирайте задачата, като отчетете следните ограничения: В един момент на касата може да „работи“ само един служител. Тоест не може двама да теглят пари едновременно, не могат и да внасят едновременно;

Ако в касата няма пари, а някой иска да изтегли, той ще се дръпне встрани и ще изчака, **докато в касата дойдат достатъчно**. Когато това се случи, той ще се вмъкне отново най-отпред на опашката и ще изтегли. Имайте предвид, че информацията за служителите трябва да се обновява постоянно и също така има предварително записана такава.

1. Анализ на заданието

Поставената задачата е за управление на достъпа до касата в магазин, където служите-лите могат да теглят и внасят пари при определени условия:

* Само един служител може да извършва операция в даден момент (не могат да теглят или внасят едновременно).
* Ако няма достатъчно пари в касата, тегленето се блокира, докато не бъдат внесени средства. Ако друг оператор иска да изтегли пари от каса, които са налични, то в касата има **достатъчно пари**, за да му бъдат дадени без значение дали има хора, които чакат на опашка.
* Чакащите служители запазват реда си и първият в опашката получава приоритет, когато в касата са постъпили налични средства.
* Данните за служителите и операциите трябва да се обновяват в реално време.

Задачата има голямо житеско приложение - банкомат. Ако няма пари, потребите-лите не могат да теглят, докато банката не го зареди или даден клиент не внесе пари в банкомата, за да си захрани банковата сметка.

# Функционално описание на приложението

Приложението ще симулира управление на касата в магазин с използване на многопоточност и синхронизация в C++. Основните функционалности са:

1. Инициализация на системата

* Зареждане на първоначалните данни за касата (налична сума).

1. Обработка на операциите

* Внасяне на пари.
* Теглене на пари.
* Управление на чакащите заявки - ако в касата няма достатъчно пари за теглене, служителят се поставя в опашка за изчакване.
* Когато парите бъдат внесени, първият чакащ служител получава достъп.

1. Обновяване на информацията в реално време

* След всяка операция се записват промени в баланса на касата.
* Служителите получават обратна връзка за успешни/неуспешни операции.

1. Завършване и освобождаване на ресурси

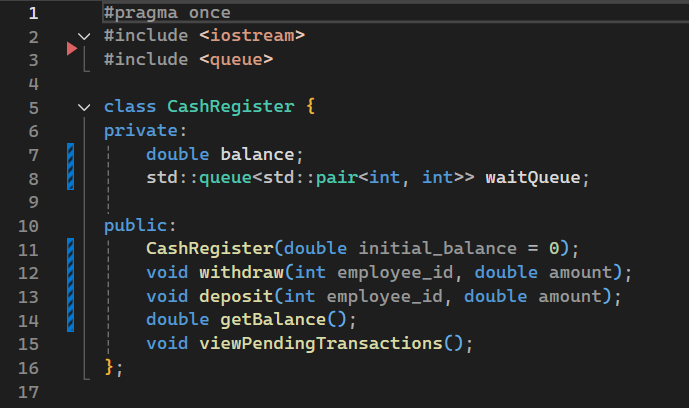
* Финален отчет за наличността в касата.
* Освобождаване на паметта.

Тези функционалности ще осигурят коректна работа на касата, като предотвратяват едновременен достъп и спазват правилата за работа с опашката на чакащите служители.

1. Изпълнение на функционалностите

Програмата се състои от два класа – CashRegister и Engine. CashRegister менежира процесите свързани с касата, Engine управлява процесите за прочитане на данни от клавиатурата.

1. CashRegister

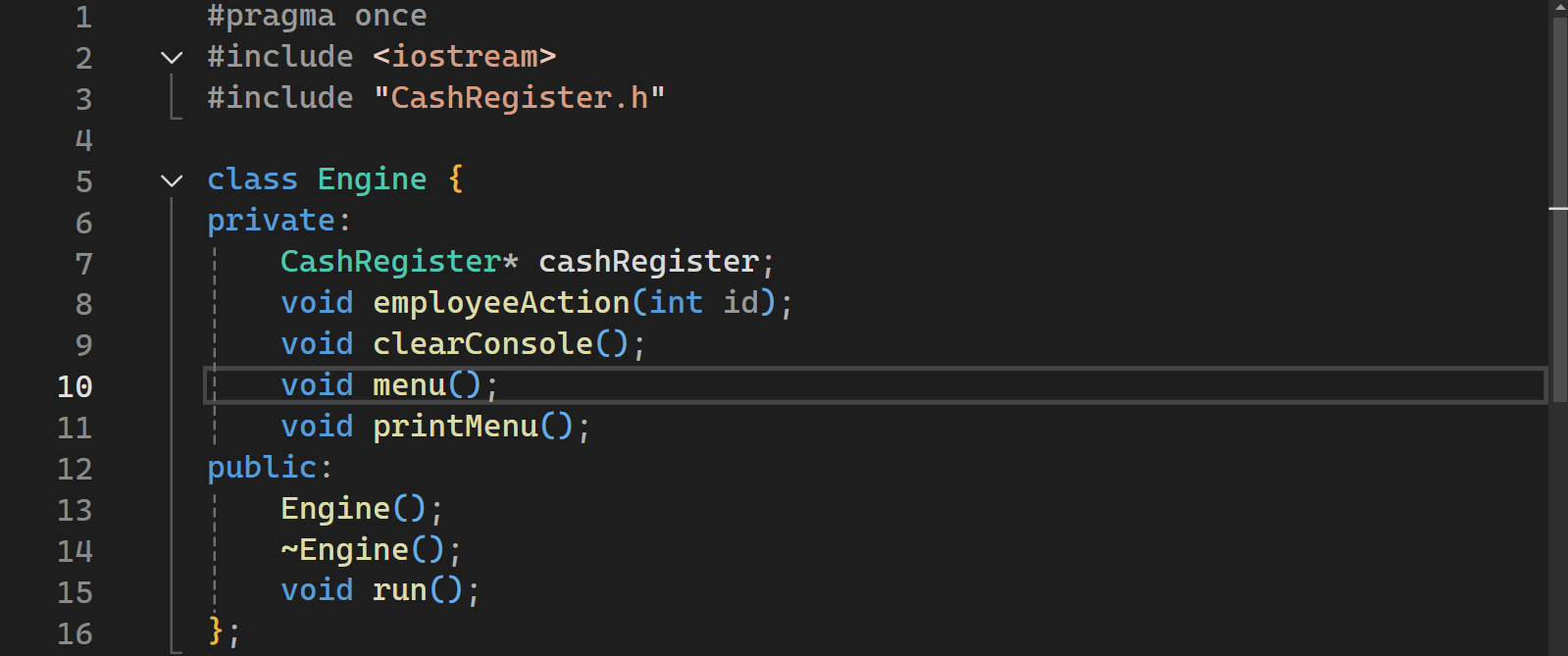


Фиг. 1 - клас CashRegister

Класът CashRegister има 5 фунцкии:

* CashRegister(double initial\_balance = 0) – констуктур с параметър начален баланс на касата
* void withdraw(int employee\_id, double amount) – фунцкия за взимане на пари от касата
* void deposit(int employee\_id, double amount) – фунцкия за внасяне на пари от касата
* double getBalance() – фунцкия за взимане на текущо състояние на касата. Връща колко пари има в касата.
* void viewPendingTransactions() – фунцкия, която отпечатва на екрана чакъщи транзакции за вземане на пари от касата.

1. Engine



Фиг. 2 - клас Engine

Класът Engine има 7 фунцкии:

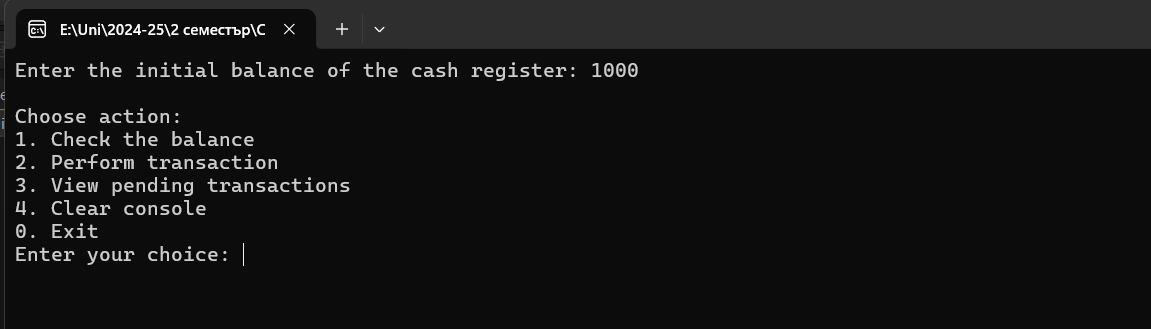
* Engine() - конструктор
* ~Engine() – деструктор
* void run() – фунцкия, която стартира програмата

void employeeAction(int id) – функция, която чете и обработва данни от клавиатурата за действие от даден служите – взимане/внасяне на пари от/в каса

* void clearConsole() – изчиства текста от конзолата.
* void menu() – обработва действията на потребителя
* void printMenu() – принтира главното меню в програмата

1. Експериментални данни

При стартиране на програмата, потребителят трябва да зададе начална наличност в касата (например 1000).



Фиг. 3 – задаване на начално състояние на касата

Потребителят може да избере какво действие да прави.

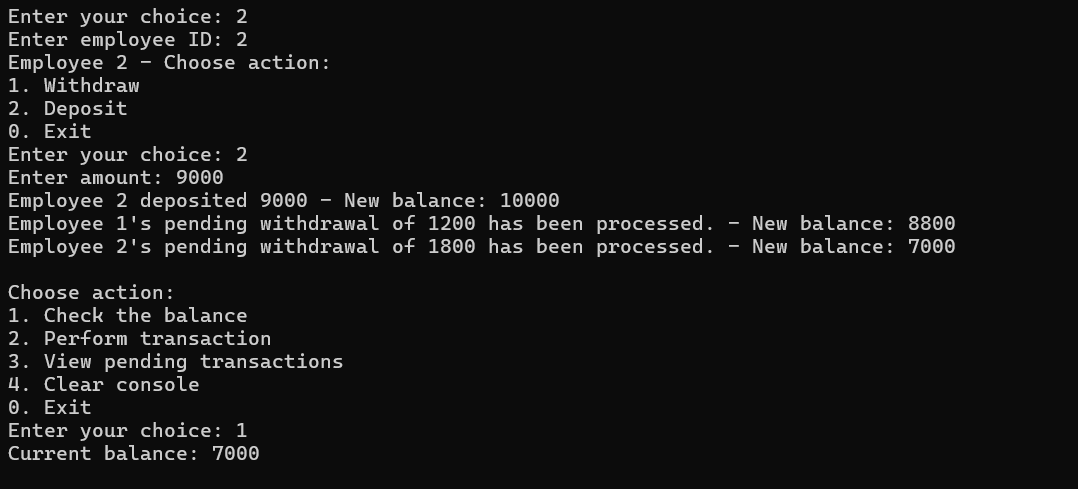
* 1. Проверка на баланса на касата
  2. Извършване на транзакция
  3. Проверка на чакащи транзакции
  4. Изчитване на конзолата
  5. 0 - Изход от програмата

Тук ще демонстрирам две транзакции за взимане на пари от касата, когато в касата няма достатъчно налични средства. Примерът е със слъжител 1, който иска да вземе 1200, както и служител 2, който иска да вземе 1800. В касата има наличност от 1000 – транзакциите за взимане на пари няма как да бъдат направени. Затова тези две транзакции са запазени в опашката, за да може да бъдат изпълнени, когато в касата постъпят нужните финансови средства. На фиг.4 съм показал резултатите от програмата с пробните данни и ситуацията описана в примера.

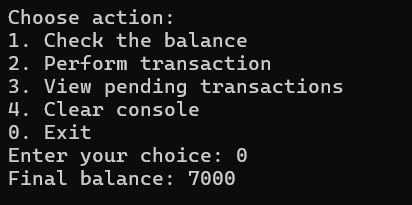


Фиг. 4 Транзакции за взимане на пари от касата

След направените заявки за теглене на пари от касата, идва служител 3 и иска да внесе 9000 в касата – фиг. 5. След като постъпят средвата в касата, програмата проверява дали има чакащи транзакции за взимана на пари. В този случай има. Автоматично служителите от опашката си взимат парите от касата. При проверка на наличното салдо в касата, то е 7000 (1000 – 1200 – 1800 + 9000). Няма чакащи транзакции в опашката.

 Фиг. 5 Транзакции за внасяне на пари в касата

Когато няма повече транзакции за въвеждане, потребителят на системата трябва да избере опция 0 – край на програмата. Фиг. 6 ни показва, че финален баланс на касата е 7000.



Фиг. 6 – финален баланс

1. Приложение

Кодът на моето приложение може да бъде дотъпен чрез: <https://github.com/dimitarkole/CashRegisterManager>