AKADEMIA GÓRNICZO-HUTNICZA

Dokumentacja do projektu

62. Shopping Cart Simulator

z przedmiotu

Języki programowania obiektowego

Elektronika i Telekomunikacja, 3 rok

Wiktor Baran

Czwartek 15:00

prowadzący: Rafał Frączek

27.12.2025

1. Opis projektu

Celem projektu jest symulacja funkcjonowania wielobranżowego sklepu w środowisku konsolowym (CLI). Interakcja z systemem odbywa się poprzez intuicyjny zestaw komend sterujących. Aplikacja oferuje pełną obsługę procesu zakupowego: od przeglądania asortymentu, przez precyzyjne zarządzanie zawartością koszyka (dodawanie i usuwanie produktów), aż po weryfikację podsumowania zamówienia. Program prezentuje kalkulację kosztów, uwzględniając ceny netto, brutto oraz aktywne rabaty oraz umożliwia zapisanie aktualnego stanu koszyka.

2. Project description

The objective of this project is to simulate the operation of a multi-category store within a console environment (CLI). Interaction with the system is conducted via an intuitive set of control commands. The application offers comprehensive support for the purchasing process: from browsing the assortment and precise cart management (adding and removing products) to verifying the order summary. The program presents cost calculations, including net and gross prices as well as active discounts, and allows for saving the current state of the cart.

3. Instrukcja użytkownika

Program po uruchomieniu wyświetla pierwszą stronę sklepu na ,której znajduje się maksymalnie 5 produktów wraz z ich opisami oraz historię koszyka z poprzedniej sesji wraz z aktualnie obowiązującą zniżką (jeśli została zapisana). W przypadku wystąpienia błędów w bazie danych sklepu program wyświetla o tym informacje oraz pomija uszkodzone rekordy (walidacja typów zmiennych). W stopce interfejsu widoczna jest informacja o aktualnie przeglądanej stronie sklepu oraz ilości wszystkich produktów w sklepie. Interakcja ze sklepem odbywa się poprzez wpisywanie następujących komend:

1) **page <numer strony> -** polecenie pozwala na przechodzenie między kolejnymi stronami sklepu

2) **eop <liczba elementów na stronie>** - polecenie pozwala na zmianę ilości wyświetlanych produktów na poszczególnej stronie oraz przenosi automatycznie na stronę pierwszą sklepu

3) **add <id produktu> <ilość>** - polecenie pozwala na dodanie produktu w danej ilości do koszyka

4) **rem <id produktu> <ilość>** - polecenie pozwala na usunięcie produktu w danej ilości z koszyka

5) **clear** – polecenie pozwala na wyczyszczenie koszyka

6) **summary** – polecenie wyświetla aktualny stan koszyka wraz z informacjami na temat zniżek podatków jak i ceny wszystkich produktów w koszyku

7) **discount <kod promocyjny>** - polecenie pozwala dodać zniżkę do finalnej ceny wszystkich produktów (aktualnie zaimplementowane kody to: promo1 – 10% zniżki oraz promo2 - 20% zniżki)

8) **discount clear** – polecenie pozwala usunąć aktualnie obowiązującą zniżkę

9) **exit** – zakończenie działania programu

Po wpisaniu komendy `exit`, program zapyta użytkownika, czy chce zapisać aktualny stan koszyka wraz z aktualnie obowiązującą zniżką. Wybranie opcji twierdzącej (y) pozwoli na odtworzenie zakupów przy kolejnym uruchomieniu aplikacji.

Program do działania wymaga predefiniowanej listy produktów w pliku **database.txt,** jest w stanie wychwycić błędy w linijkach polegające na nieprawidłowym typie wprowadzonej danej aczkolwiek nie weryfikuje listy pod względem sensu logicznego danych. Po każdej edycji pliku database.txt należy wyczyścić plik **history_summary.txt** przechowujący historie koszyka.

4. Kompilacja

Projekt został zaimplementowany i przetestowany w środowisku **Microsoft Visual Studio**. Kod źródłowy korzysta wyłącznie ze standardowych bibliotek języka C++ (STL), w związku z czym **wymagana jest jedynie standardowa kompilacja**. Nie jest konieczne instalowanie ani dołączanie żadnych zewnętrznych bibliotek czy niestandardowa konfiguracja linkera.

Wymagania systemowe i kompilacyjne:

- **System operacyjny:** Microsoft Windows.
    - Uzasadnienie: Program wykorzystuje funkcje systemowe specyficzne dla konsoli Windows (`system("cls")` do czyszczenia ekranu oraz `system("pause")` do zatrzymywania działania), co czyni go nieprzenośnym na systemy Linux/macOS bez modyfikacji kodu.
- **Standard języka:** C++14 lub nowszy.
    - Uzasadnienie: Projekt wykorzystuje nowoczesne mechanizmy zarządzania pamięcią, w tym funkcję `std::make_unique` (do tworzenia inteligentnych wskaźników), która jest dostępna od standardu C++14.

Instrukcja kompilacji: Proces budowania aplikacji odbywa się poprzez standardowe polecenie kompilacji w środowisku Microsoft Visual Studio (opcja Build Solution / Kompiluj rozwiązanie).

5. Pliki źródłowe

Projekt składa się z następujących plików źródłowych:

Book.h , Book.cpp – deklaracja i implementacja klasy Book

Clothes.h , Clothes.cpp – deklaracja i implementacja klasy Clothes

Electronics.h, Electronics.cpp – deklaracja i implementacja klasy Electronics

Furniture.h , Furniture.cpp – deklaracja i implementacja klasy Furniture

Toys.h, Toys.cpp – deklaracja i implementacja klasy Toys

Product.h, Product.cpp – deklaracja i implementacja klasy Product

DefFunctions.h, DefFunctions.cpp – deklaracja i implementacja funkcji

ProjektTemat62JPO.cpp – główny plik programu

6. Zależności

Brak.

7. Opis klas

W projekcie utworzono następujące klasy:

Product – klasa bazowa reprezentująca ogólny produkt w sklepie.

- `string get_name() const` – zwraca nazwę produktu,
- `float get_nettoprice() const` – zwraca cenę netto produktu,
- `float get_bruttoprice() const` – zwraca cenę brutto produktu,

- `virtual void display() const` – metoda wirtualna, wyświetla podstawowe informacje o produkcie na standardowe wyjście.

Furniture – klasa reprezentująca meble, dziedzicząca po klasie Product.

- `string get_material() const` – zwraca informację o materiale wykonania,
- `string get_dimensions() const` – zwraca wymiary mebla,
- `string get_color() const` – zwraca kolor mebla,
- `void display() const` – nadpisuje metodę bazową, wyświetlając dodatkowo specyfikację mebla.

Clothes – klasa reprezentująca odzież, dziedzicząca po klasie Product.

- `int get_size() const` – zwraca rozmiar ubrania,
- `string get_material() const` – zwraca rodzaj tkaniny,
- `string get_colour() const` – zwraca kolor ubrania,
- `void display() const` – nadpisuje metodę bazową, prezentując atrybuty odzieżowe.

Electronics – klasa reprezentująca sprzęt elektroniczny, dziedzicząca po klasie Product.

- `string get_brand() const` – zwraca markę urządzenia,
- `string get_model() const` – zwraca model urządzenia,
- `int get_warranty_months() const` – zwraca długość gwarancji w miesiącach,
- `void display() const` – nadpisuje metodę bazową, wyświetlając specyfikację techniczną.

Book – klasa reprezentująca książki, dziedzicząca po klasie Product.

- `string get_author() const` – zwraca autora książki,
- `string get_isbn() const` – zwraca numer identyfikacyjny ISBN,
- `int get_pages() const` – zwraca liczbę stron,
- `void display() const` – nadpisuje metodę bazową, prezentując dane bibliograficzne.

Toys – klasa reprezentująca zabawki, dziedzicząca po klasie Product.

- `int get_min_age() const` – zwraca minimalny wiek dziecka,
- `string get_material() const` – zwraca materiał wykonania zabawki,
- `string get_type() const` – zwraca typ zabawki (np. edukacyjna),
- `void display() const` – nadpisuje metodę bazową, wyświetlając parametry zabawki.

8. Zasoby

W projekcie wykorzystywane są następujące pliki zasobów:

- database.txt – plik zawierający definicje asortymentu sklepu. Na podstawie zawartych w nim rekordów program, podczas sekwencji startowej, dynamicznie inicjalizuje obiekty produktów i dodaje je do danych aplikacji. Plik ten jest niezbędny do prawidłowego działania programu. Specyfikacja formatu rekordu:

Ogólny schemat wiersza: [TYP] [NAZWA] [CENA_NETTO] [CENA_BRUTTO] [ATRYBUTY_SPECYFICZNE]. Dane są oddzielane znakami białymi.

Gdzie [TYP] przyjmuje jedną z wartości:

- C – Clothes
- E – Electronics
- F – Furniture
- B – Books
- T – Toys

- history_summary.txt – plik generowany automatycznie przez program, zawierający zrzut stanu koszyka (indeksy produktów i ich ilości), wartość zmiennej discount, umożliwiający odtworzenie sesji zakupowej oraz liczbę wszystkich pozycji w podsumowaniu.

  Schemat zapisu: [LICZBA_POZYCJI] ([INDEKS_PRODUKTU] [ILOŚĆ]) ([INDEKS_PRODUKTU] [ILOŚĆ]) ([INDEKS_PRODUKTU] [ILOŚĆ]) ... [WARTOŚĆ_ZNIŻKI w %] .

9. Dalszy rozwój i ulepszenia

Projekt można w przyszłości rozbudować o następujące funkcjonalności:

1) System Kontroli Magazynowej Obecnie sklep pozwala na zakup nieskończonej liczby sztuk danego towaru.

2) System kont użytkowników i administratora Rozbudowa aplikacji o system logowania:

- **Klient:** Możliwość posiadania własnej, odseparowanej historii zakupów i zapisanych adresów dostawy.
- **Administrator:** Dostęp do ukrytego menu, pozwalającego na dodawanie, usuwanie i edycję produktów w sklepie bezpośrednio z poziomu konsoli, bez konieczności ręcznej edycji pliku tekstowego.

3) Zaawansowane filtrowanie i sortowanie Implementacja algorytmów sortowania (np. po cenie rosnąco/malejąco, alfabetycznie) oraz filtrowania (np. wyświetl tylko produkty z kategorii "Elektronika" tańsze niż 500 zł). Zwiększyłoby to znacznie użyteczność sklepu przy dużej liczbie produktów.

4) Moduł symulacji płatności i generowania faktur Obecnie proces zakupowy kończy się na wyświetleniu podsumowania (`summary`). Kolejnym krokiem w rozwoju aplikacji byłaby implementacja symulacji bramki płatniczej.

5) Implementacja Graficznego Interfejsu Użytkownika (GUI) Obecna wersja aplikacji opiera się na interfejsie tekstowym (CLI), co jest rozwiązaniem wydajnym, ale mało intuicyjnym dla przeciętnego użytkownika końcowego.

10. Inne

Brak.