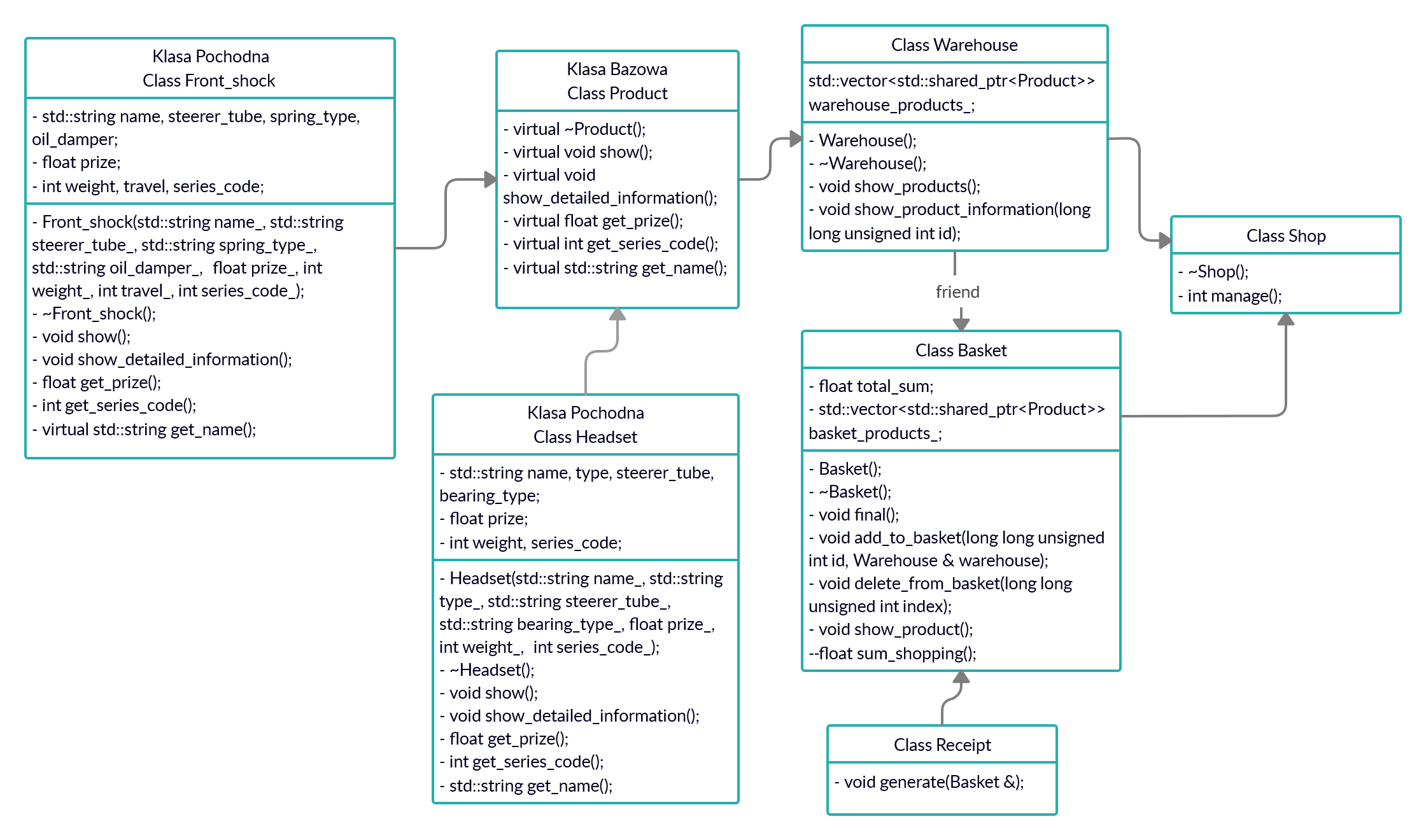
1. **Dane kontaktowe:**
   1. Mateusz Chechłowski
   2. Przedmiot: PROE
   3. Semestr: II
   4. Nazwisko prowadzącego: mgr Inż. Marcin Bączyk
2. **Temat programu:**

**„SKLEP ROWEROWY”**

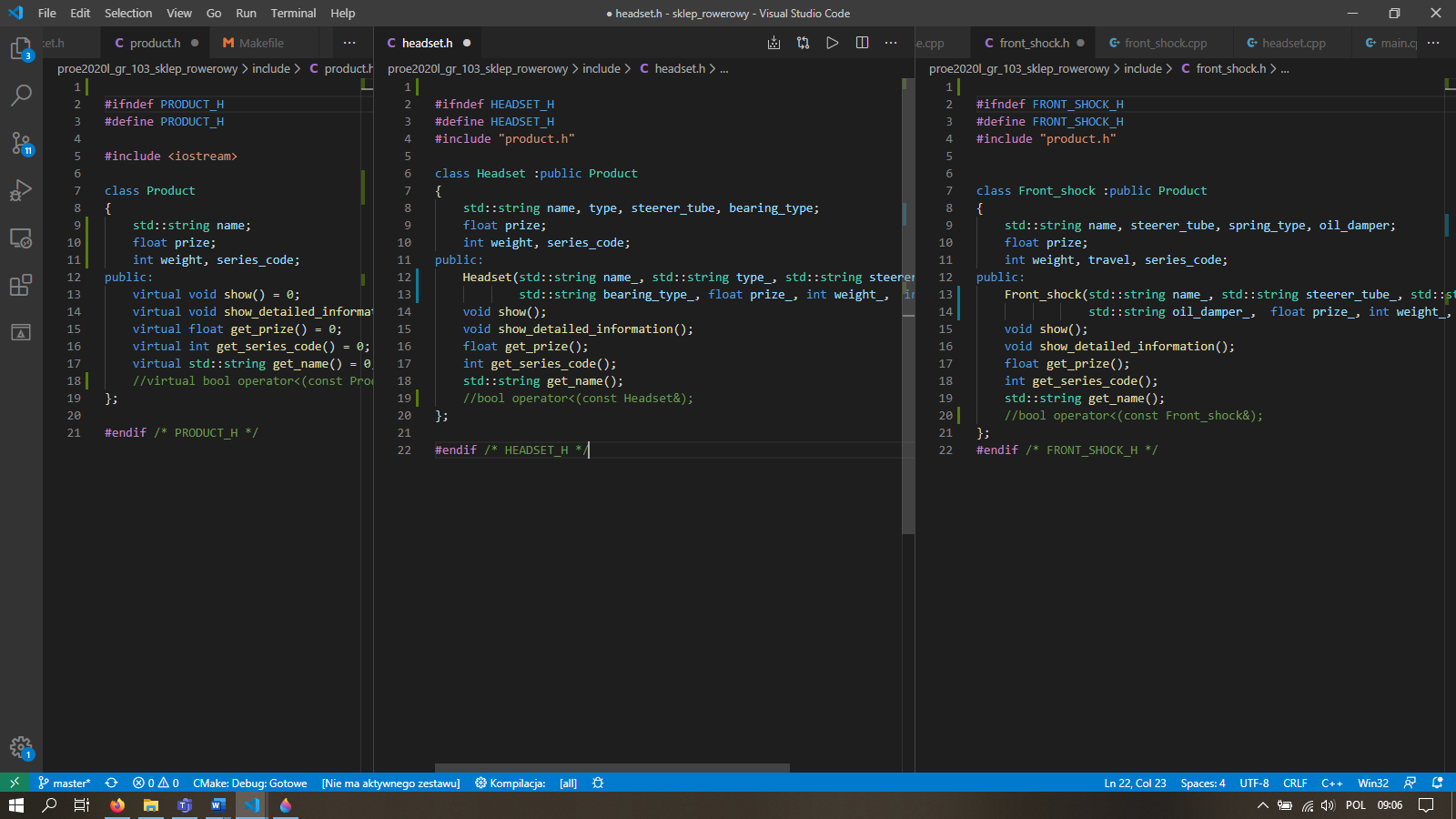
1. **Specyfikacja wstępna:**
   1. **Temat**

„Program obsługujący sklep rowerowy”

* 1. **Interfejs aplikacji**
     1. Uruchamianie z poziomu powłoki:
        1. Podczas wywołania programu zostanie wyświetlona lista komend dostępnych do użytkownika takich jak: dodaj produkt do koszyka, usuń produkt z koszyka, zobacz listę dostępnych produktów, wygeneruj paragon(zapis do pliku) itp.
        2. Program będzie robił niezbędne rzeczy podczas obsługi klienta.
        3. Sposób prezentacji wyników:
           1. Wyświetlanie wyników w terminalu
           2. Zapis do pliku
  2. **Analiza obiektów występujących w trakcie działania aplikacji** 
     1. **Shop,** obsługujący sklep
     2. **Basket,**  obsługujący koszyk
     3. Klasa **Basket** przyjaźni się z klasą **Warehouse**
     4. **Receipt,** obsługujący paragon
     5. **Warehouse,** magazyn przechowujący produkty
     6. **Product(**klasa abstrakcyjna do dodawania produktów w sklepie)
     7. Klasa **Product** jest klasą bazową, klasy takie jak: **Front\_shock**, **Headset**(czyli części rowerowe lub rowery) są klasami pochodnymi(wykorzystany został tutaj **polimorfizm**)

1. **Diagram klas:**
2. **Opis najciekawszych fragmentów kodu:**

Najważniejszym elementem programu, chyba jest zastosowanie polimorfizmu, ponieważ bardzo ułatwiło to tworzenie programu. Utworzyłem klasę **Product** i według jej „przepisu” tworzyłem obiekty. Dzięki temu mogłem utworzyć wektor produktów i z przyjemnością go wykorzystywać.

 Kolejnym ciekawym elementem jest sam interfejs, czyli mechanizm obsługi sklepu: wyświetlanie szczegółowych informacji dzięki czemu nie ma „zaśmieconego ” interfejsu. Jest możliwość dodawania i usuwania z koszyka oraz generowanie paragonu.

1. **Krótki opis najważniejszych klas i metod:** 
   1. Klasa **Product:**

Klasa **Product** jest **klasą abstrakcyjną,** której klasami pochodnymi są produkty w sklepie

* + 1. Atrybuty:
       1. **std::string name;** -> nazwa
       2. **float prize;** -> cena
       3. **int weight, series\_code;** -> waga, numer seryjny
    2. Metody:
       1. **virtual void show() = 0;** -> wyświetla: indeks, nazwę, wagę, cenę
       2. **virtual void show\_detailed\_information() = 0;** -> wyświetla szczegółowe informacje o produkcie
       3. **virtual float get\_prize() = 0;** -> pobiera cenę produktu
       4. **virtual int get\_series\_code() = 0;** -> pobiera numer seryjny
       5. **virtual std::string get\_name() = 0;** -> pobiera nazwę
  1. Klasa **Warehouse:**

Klasa **Warehouse** przechowuje produkty

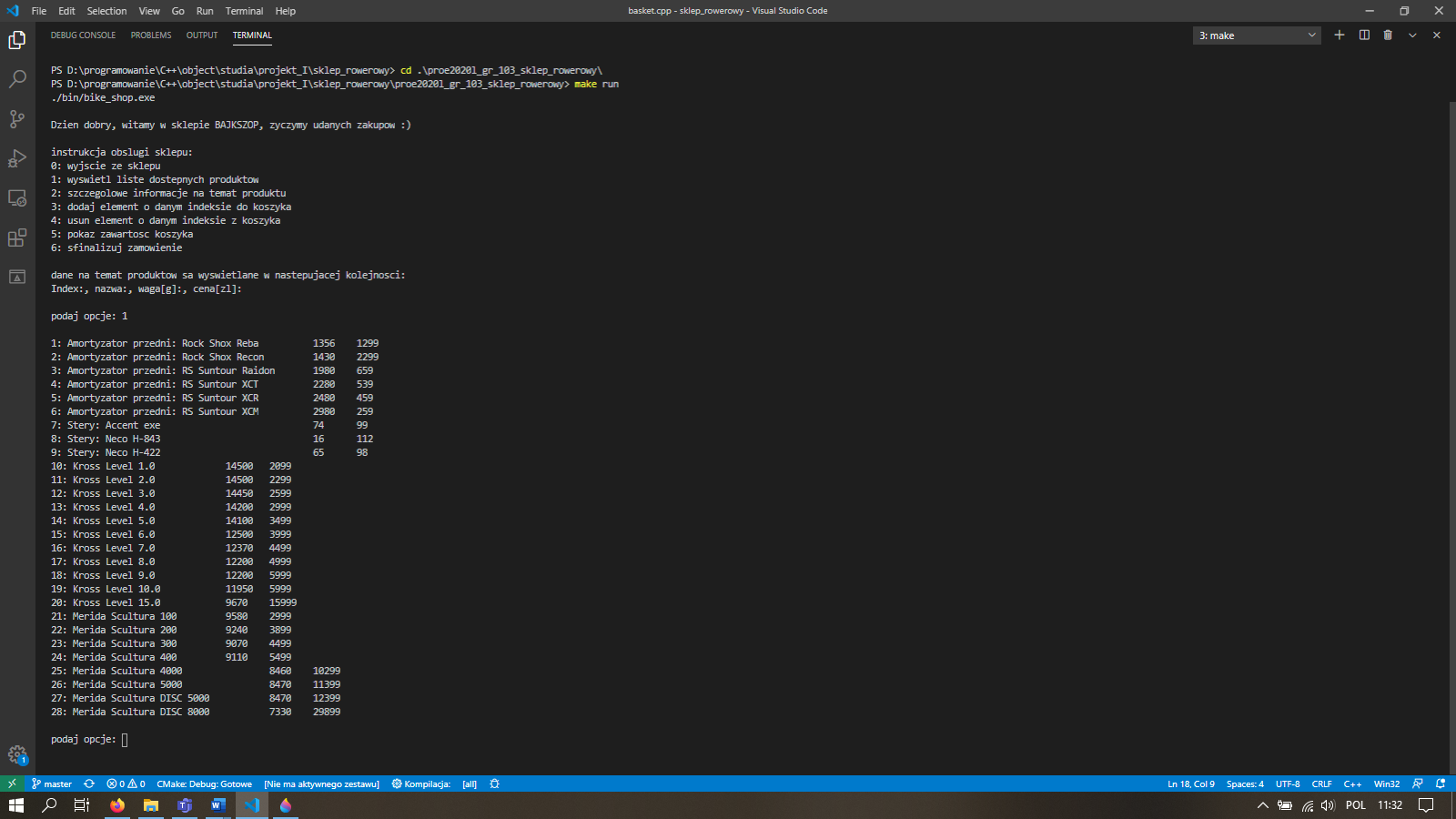
* + 1. Atrybuty:
       1. **std::vector<std::shared\_ptr<Product>>warehouse\_products\_;** -> wektor produktów w magazynie

1. Metody:
   1. **Warehouse();** -> konstruktor
   2. **~Warehouse();** -> destruktor
   3. **void show\_products();** -> wyświetla zawartość magazynu
   4. **void show\_product\_information(long long unsigned int id);** -> wyświetla szczegółowe informacje dla danego produktu
   5. Klasa  **Basket:**

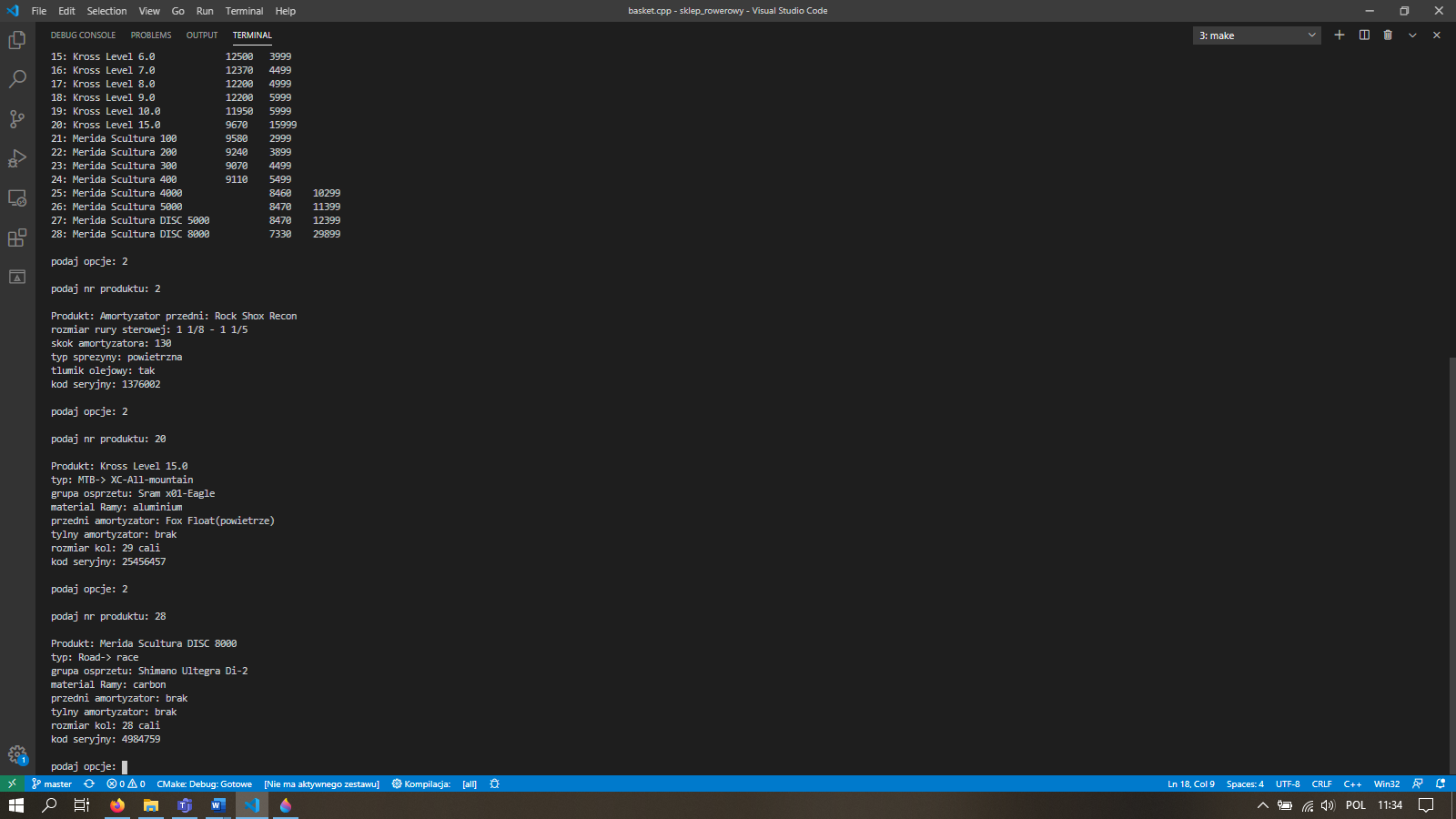
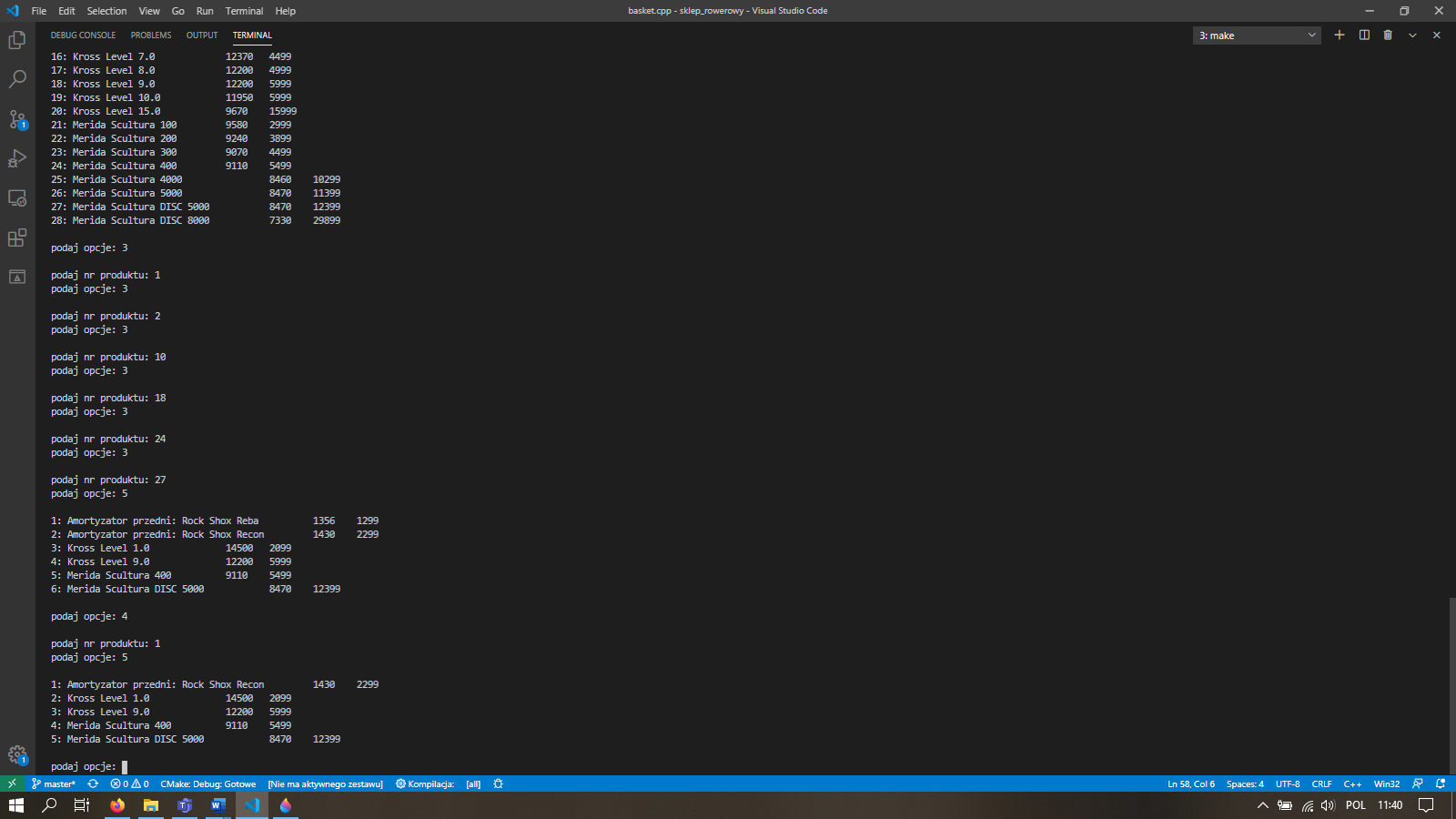
Klasa **Basket** przyjaźni się z klasą **warehouse**

* + 1. Atrybuty:
       1. **float total\_sum;** -> całkowita suma zakupów
       2. **std::vector<std::shared\_ptr<Product>> basket\_products\_;**  -> wektor przechowujący elementy w koszyku
    2. Metody:
       1. **Basket();** -> konstruktor
       2. **~Basket();**  -> destruktor
       3. **void final();** -> podsumowuje zamówienie
       4. **void add\_to\_basket(long long unsigned int id, Warehouse & warehouse);** -> dodaje produkt o wybranym indeksie do koszyka
       5. **void delete\_from\_basket(long long unsigned int index);** -> usuwa produkt o wybranym indeksie z koszyka
       6. **void show\_product();** -> wyświetla elementy w koszyku
       7. **float sum\_shopping();** -> generuje kwotę zamówienia
       8. **friend std::ostream &operator<<(std::ostream &, Basket&);** -> przeciążenie operatora
  1. Klasa **Receipt:**
     1. Metody:
        1. **Void generate(Basket &)** -> generuje paragon
  2. Klasa **Shop:**
     1. Metody:
        1. **Int manage()** -> zarządza sklepem

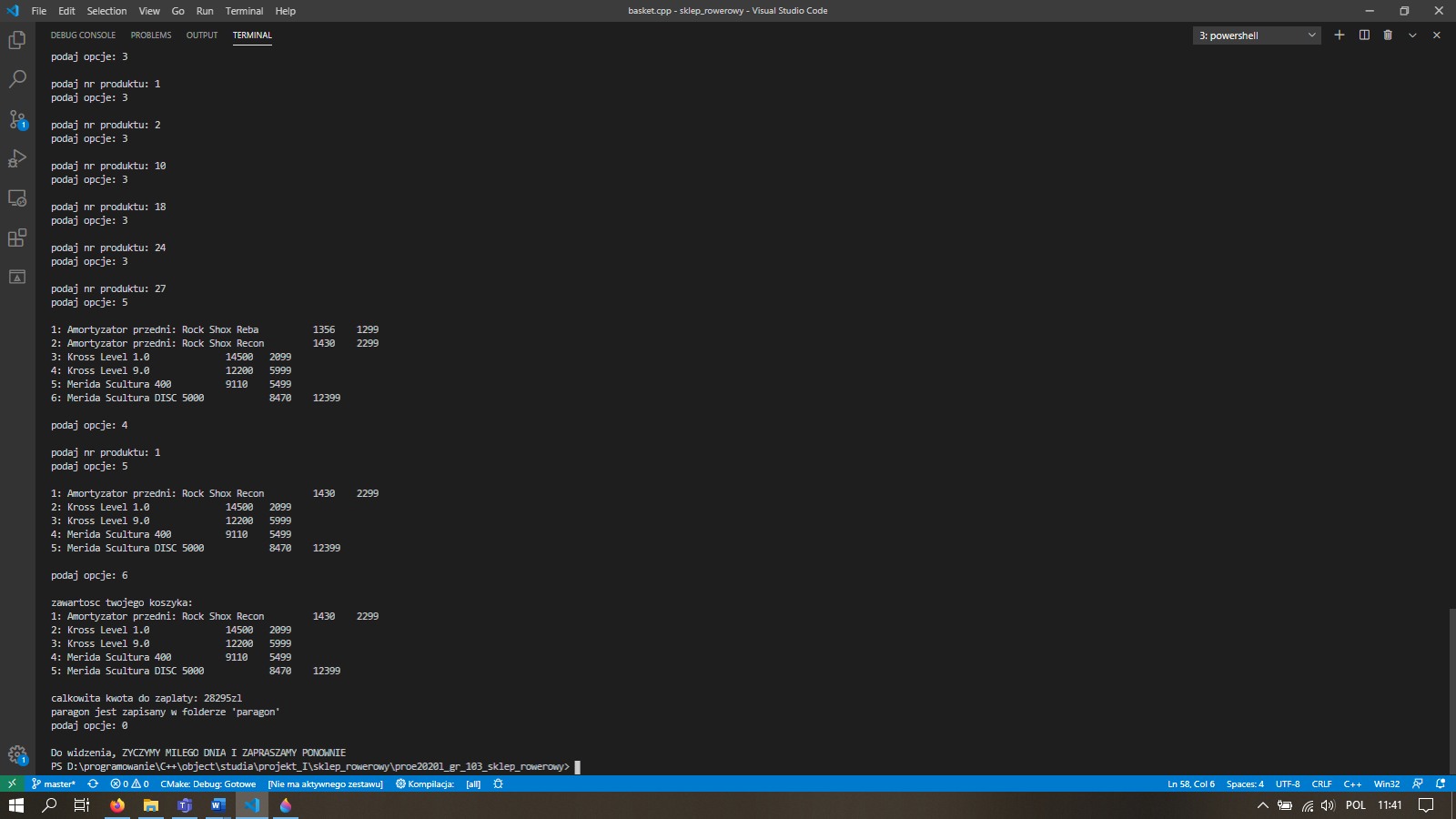
1. **Krótka instrukcja posługiwania się programem**
   1. **Po włączenie programu wyświetlą się takie napisy:**
   2. **"Dzień dobry, witamy w sklepie BAJKSZOP, życzymy udanych zakupów :)"**
   3. **"instrukcja obsługi sklepu:"**
   4. **"0: wyjście ze sklepu"**
   5. **"1: wyświetl listę dostępnych produktów"**
   6. **"2: szczegółowe informacje na temat produktu"**
   7. **"3: dodaj element o danym indeksie do koszyka"**
   8. **"4: usuń element o danym indeksie z koszyka"**
   9. **"5: pokaz zawartość koszyka"**
   10. **"6: sfinalizuj zamówienie"**
   11. **"dane na temat produktów są wyświetlane w następującej kolejności: Index: , nazwa: , waga[g]: , cena[zł]:**
2. **Przykład działania programu:**

**Wyświetlenie dostępnych produktów:**

**Szczegółowe informacje na temat produktu:**

**Dodawanie i usuwanie elementów z koszyka oraz wyświetlanie jego zawartości:**

**Finalizacja zamówienia i wygenerowanie paragonu:**

**Paragon:**