Zadania z podstaw programowania obiektowego – część II

1. Napisać program do obsługi zamówień. W tym celu należy zaimplementować klasy ElementZamowienia oraz Zamowienie.

   Klasa ElementZamowienia reprezentuje pojedynczą pozycję zamówienia i zawiera prywatne pola:

   - `string nazwa_`
   - `double cena_`
   - `int liczbaSztuk_`

   Dodatkowo klasa ta powinna mieć następujące metody publiczne:

   - konstruktor bezparametrowy,
   - konstruktor z parametrami umożliwiającymi przypisanie wartości początkowych polom klasy,
   - `string toString()` zwracającą łańcuch znaków opisujący element zamówienia, np. "Chleb 4.00 zł, 2 szt., łącznie 8.00 zł"
   - `double obliczKoszt()` zwracającą łączny koszt danej pozycji zamówienia z uwzględnieniem rabatu,
   - `double obliczRabat()` wyznaczającą rabat dla klienta, jeżeli klient zamówił co najmniej 5 sztuk tego towaru to rabat wynosi 10%, w przeciwnym razie 0%

   Klasa Zamowienie zawiera listę zamówień i zawiera następujące pola prywatne:

   - `ElementZamowienia *elementy_` – dynamiczna tablica z elementami zamówienia
   - `int rozmiar_` – aktualna liczba elementów w zamówieniu
   - `int maksRozmiar_` – maks. liczba elementów w zamówieniu

   Dodatkowo klasa ta powinna mieć następujące metody publiczne:

   - konstruktor z jednym parametrem określającym maks. liczbę elementów zamówienia; w konstruktorze powinna zostać przydzielona pamięć dla tablicy `elementy_`

- destruktor zwalniający pamięć przydzieloną w konstruktorze
- `bool dodaj(const ElementZamowienia &p)` – dodaje podany element do zamówienia (dopisuje do tab. elementy_)
- `double obliczKoszt()` – oblicza całkowity koszt zamówienia
- `void pisz()` – wypisuje na ekranie informacje o zamówieniu tj. listę pozycji, łączny koszt oraz naliczony rabat

Przykład zastosowania klas:

```
int main(int argc, const char *argv[])
{
    Zamowienie z;
    z.dodaj(ElementZamowienia("Chleb", 4.0, 2));
    z.dodaj(ElementZamowienia("Mleko", 2.5, 1));
    z.dodaj(ElementZamowienia("Cukier", 4.0, 5));
    z.dodaj(ElementZamowienia("Papierosy", 9.0, 1));
    z.pisz();
    return 0;
}
```

Przykładowy wydruk:

```
Zamowienie:
    1. Chleb 4.00 zł, 2 szt., łącznie 8.00 zł
    2. Mleko 2.50 zł, 1 szt., łącznie 2.50 zł
    3. Cukier 4.00 zł, 5 szt., łącznie 18.00 zł
    4. Papierosy 9.00 zł, 1 szt., łącznie 9.00 zł
Koszt całkowity: 37.5 zł
Naliczony rabat: 2 zł
```

2. Napisać klasę `Obraz` umożliwiającą „rysowanie” w odcieniach szarości. Klasa ta powinna mieć następujące pola składowe:

   - `unsigned int szerokosc`;
   - `unsigned int wysokosc`;
   - `unsigned char* piksele` – tablica kolorów poszczególnych pikseli obrazka. Rozmiar tablicy to `szerokosc` × `wysokosc`;

   oraz następujące metody:

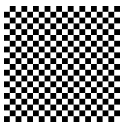
   - konstruktor bezparametrowy tworzący obrazek o wymiarach 0×0;
   - konstruktor z parametrami umożliwiającymi ustalenie szerokości i wysokości obrazka oraz przydzielający pamięć dla tablicy `piksele`;

- destruktor zwalniający pamięć przydzieloną w konstruktorze;
- `dajWysokosc` zwracającą wysokość obrazka;
- `dajSzerokosc` zwracającą szerokość obrazka;
- `dajRozmiar` zwracającą liczbę pikseli obrazka;
- `zapiszDoPlikuPGM` zapisującą obraz do pliku o ścieżce podanej w parametrze metody. Obraz powinien być zapisany w formacie *portable graymap format (PGM)*, którego format można przedstawić na przykładzie:

  ```
  P5
  100 100 255
  ... (jasność pikseli)
  ```

  gdzie „P5" to wymagany ciąg znaków w pierwszym wierszu, „100 100 255" w drugim wierszu oznaczają, odpowiednio, szerokość, wysokość oraz maksymalną jasność piksela. W kolejnym wierszu znajduje się ciąg bajtów określających nasycenie kolejnych pikseli obrazka. Do zapisu pikseli można zastosować, np. metodę `write(const char* s, streamsize n)` z klasy `std::ofstream`, należy przy tym pamiętać o otwarciu pliku w trybie `std::ios::binary`;
- `ustawPiksel(unsigned int x, unsigned int y, unsigned char kolor)` ustalającą kolor piksela o podanych współrzędnych;
- `rysujPoziomaLinie(unsigned int y, unsigned char kolor` rysującą poziomą linię od lewej do prawej krawędzi obrazka i o zadanym kolorze;
- `rysujPionowaLinie` analogicznie do poprzedniej, ale w pionie;
- `szachownica(unsigned int WymiarPola, unsigned char kolor)` rysuje wzór szachownicy zadanym kolorem o szerokości pola określonego przez pierwszy parametr. Przykład pokazuje rys. 10.1.

Wskazówka. Pliki w formacie PGM można wyświetlić, np. za pomocą strony `http://paulcuth.me.uk/netpbm-viewer`.



Rysunek 10.1: Przykładowy obrazek do Zad. 2.