Dokumentacja projektu zaliczeniowego z Algorytmów i Struktury Danych

Temat: Sklep wędkarski
Autor:Krystian Cieloch
10.02.2024

1.Wstęp

Projekt sklepu wędkarskiego “Barwena” napisanego w języku C++. Aplikacje umożliwia kupowanie różnych produktów wędkarskich z różnych kategorii, tj. Wędki, Kołowrotki, Przynęty, Spławiki, Żyłki i plecionki oraz Akcesoria. Zakupy są przechowywane w koszyku, a po dokonaniu zakupu otrzymujemy raport.

2. Sposób uruchomienia

Program jesteśmy w stanie uruchomić za pomocą pliku Makefile wykonując kolejne polecenia w terminalu:
-make
-make run
Po zakończeniu użytkowania programu możemy wyczyścić pliki za pomocą make clean

3. Opis implementacji

W tym punkcie opiszemy poszczególne części naszego kodu:

- StronaStartowa()

Ta funkcja wyświetla ekran powitalny sklepu. Za pomocą
this_thread::sleep_for(std::chrono::milliseconds(2000))
zatrzymujemy program na 2000ms, a dzięki system("clear") możemy wyczyścić ekran dzięki czemu zniknie nam ekran powitalny.

```
void StronaStartowa() {
    std::cout << "*************************************************************" << std::endl;
    std::cout << "*                  Sklep Wedkarski \"Barwena\"              *" << std::endl;
    std::cout << "*              Projekt obiektowej aplikacji C++             *" << std::endl;
    std::cout << "*                                                           *" << std::endl;
    std::cout << "*************************************************************" << std::endl;


    std::this_thread::sleep_for(std::chrono::milliseconds(2000));
    system("clear");
}
```

• wyswietl_kategorie()
Ta funkcja wyświetla kategorie znajdujące się w sklepie

```
void wyswietl_kategorie() {
    std::cout << "Wybierz kategorie produktu: " << std::endl;
    std::cout << "1. Wedki" << std::endl;
    std::cout << "2. Kolowrotki" << std::endl;
    std::cout << "3. Przynety" << std::endl;
    std::cout << "4. Splawiki" << std::endl;
    std::cout << "5. Zylki i plecionki" << std::endl;
    std::cout << "6. Akcesoria" << std::endl;
    std::cout << "7. Wyswietl koszyk" << std::endl;
    std::cout << "8. Zakoncz zakupy" << std::endl << "Wybor: " << std::endl;
}
```

• Class Produkt
To jest klasa abstrakcyjna, ma dwie wirtualne metody wyswietl() i pobierzCene(). Klasy które dziedziczą po tej funkcji muszą zawierać te metody.

```
class Produkt {
public:
    virtual void wyswietl() = 0;
    virtual double pobierzCene() const = 0;
    virtual ~Produkt() = default;
};
```

• Class Wedka

Klasa Wedka dziedziczy po klasie Produkt
Wedka(string n, double c, string p) inicjalizuje obiekt wedki przy użyciu podanych wartości.
Wedka() ustawia domyślne wartości, jeśli nie są dostarczone żadne inne.
wyswietl(): Wyświetla informacje o wedce, w tym nazwę, cenę i długość.
ustaw():Ustawia parametry wedki na podstawie dostarczonych wartości.
pobierzCene():Zwraca cenę wedki.

```cpp
class Wedka :public Produkt {
public:

    Wedka(std::string n, double c, std::string p) : nazwa(n), cena(c), parametr(p) {}
    Wedka() : nazwa("brak"), cena(0.0), parametr("brak") {}

    void wyswietl() override {
        std::cout << "Nazwa: " << nazwa << " Cena" << cena << " Dlugosc:" << parametr << " m" << std::endl;
    }
    std::string nazwa;
    double cena;
    std::string parametr;

    void ustaw(std::string naz, double cen, std::string par)
    {
        nazwa = naz;
        cena = cen;
        parametr = par;
    }

    double pobierzCene() const override{
        return cena;
    }
};
```

- Class Kolowrotek

Klasa Kolowrotek dziedziczy po klasie Produkt
Kolowrotek(string n, double c, string p) inicjalizuje obiekt kolowrotka przy użyciu podanych wartości.
Kolowrotek() ustawia domyślne wartości, jeśli nie są dostarczone żadne inne.
wyswietl(): Wyświetla informacje o kołowrotku, w tym nazwę, cenę i długość.
ustaw():Ustawia parametry kołowrotka na podstawie dostarczonych wartości.
pobierzCene():Zwraca cenę kołowrotka.

```cpp
class Kolowrotek : public Produkt {
public:
    Kolowrotek(const std::string& n, double c, const std::string& p)
        : nazwa(n), cena(c), parametr(p) {}
    Kolowrotek() : nazwa("brak"), cena(0.0), parametr("brak") {}

    void wyswietl() override {
        std::cout << "Nazwa: " << nazwa << " Cena: " << cena << std::endl;
    }

    void ustaw(const std::string& naz, double cen, const std::string& par) {
        nazwa = naz;
        cena = cen;
        parametr = par;
    }

    double pobierzCene() const override{
        return cena;
    }

private:
    std::string nazwa;
    double cena;
    std::string parametr;
};
```

• Class Przynęta

Klasa Przyenta dziedziczy po klasie Produkt
Przyneta(string n, double c, string p) inicjalizuje obiekt przynęty przy użyciu podanych wartości.
Przyneta() ustawia domyślne wartości, jeśli nie są dostarczone żadne inne.
wyswietl():Wyświetla informacje o przynęcie, w tym nazwę, cenę i kolor.
ustaw():Ustawia parametry przynęty na podstawie dostarczonych wartości.
pobierzCene():Zwraca cenę przynęty.

```
class Przyneta : public Produkt {
public:
    Przyneta(const std::string& n, double c, const std::string& p)
        : nazwa(n), cena(c), parametr(p) {}
    Przyneta() : nazwa("brak"), cena(0.0), parametr("brak") {}

    void wyswietl() override {
        std::cout << "Nazwa: " << nazwa << " Cena: " << cena << " Kolor: " << parametr << std::endl;
    }

    void ustaw(const std::string& naz, double cen, const std::string& par) {
        nazwa = naz;
        cena = cen;
        parametr = par;
    }

    double pobierzCene() const override{
        return cena;
    }

private:
    std::string nazwa;
    double cena;
    std::string parametr;
};
```

• Class Splawik

Klasa Splawik dziedziczy po klasie Produkt
Splawik(string n, double c, string p) inicjalizuje obiekt spławika przy użyciu podanych wartości.
Splawik() ustawia domyślne wartości, jeśli nie są dostarczone żadne inne.
wyswietl(): Wyświetla informacje o spławika, w tym nazwę, cenę i długość.
ustaw():Ustawia parametry spławika na podstawie dostarczonych wartości.
pobierzCene():Zwraca cenę spławika.

```cpp
class Splawik : public Produkt {
public:
    Splawik(const std::string& n, double c, const std::string& p)
        : nazwa(n), cena(c), parametr(p) {}
    Splawik() : nazwa("brak"), cena(0.0), parametr("brak") {}

    void wyswietl() override {
        std::cout << "Nazwa: " << nazwa << " Cena: " << cena << " Rodzaj: " << parametr << std::endl;
    }

    void ustaw(const std::string& naz, double cen, const std::string& par) {
        nazwa = naz;
        cena = cen;
        parametr = par;
    }

    double pobierzCene() const override{
        return cena;
    }

private:
    std::string nazwa;
    double cena;
    std::string parametr;
};
```

- Class Zylka

Klasa Zylka dziedziczy po klasie Produkt
Zylka(string n, double c, string p) inicjalizuje obiekt żyłki przy użyciu podanych wartości.
Zylka() ustawia domyślne wartości, jeśli nie są dostarczone żadne inne.
wyswietl(): Wyświetla informacje o żyłki, w tym nazwę, cenę i grubość.
ustaw():Ustawia parametry spławika na podstawie dostarczonych wartości.
pobierzCene():Zwraca cenę żyłki.

```cpp
class Zylka : public Produkt {
public:
    Zylka(const std::string& n, double c, const std::string& p)
        : nazwa(n), cena(c), parametr(p) {}
    Zylka() : nazwa("brak"), cena(0.0), parametr("brak") {}

    void wyswietl() override {
        std::cout << "Nazwa: " << nazwa << " Cena: " << cena << " Grubosc: " << parametr << " mm" << std::en
    }

    void ustaw(const std::string& naz, double cen, const std::string& par) {
        nazwa = naz;
        cena = cen;
        parametr = par;
    }

    double pobierzCene() const override{
        return cena;
    }

private:
    std::string nazwa;
    double cena;
    std::string parametr;
};
```

- Class Aksesoria

Klasa Akcesoria dziedziczy po klasie Produkt
Akcesoria(string n, double c, string p) inicjalizuje obiekt akcesorium przy użyciu podanych wartości.
Akcesoria() ustawia domyślne wartości, jeśli nie są dostarczone żadne inne.
wyswietl(): Wyświetla informacje o akcesorium, w tym nazwę, cenę i dodatkowy parametr.
ustaw():Ustawia parametry akcesorium na podstawie dostarczonych wartości.
pobierzCene(): Zwraca cenę akcesorium.

```
class Akcesoria : public Produkt {
public:
    Akcesoria(const std::string& n, double c, const std::string& p)
        : nazwa(n), cena(c), parametr(p) {}
    Akcesoria() : nazwa("brak"), cena(0.0), parametr("brak") {}

    void wyswietl() override {
        std::cout << "Nazwa: " << nazwa << " Cena: " << cena << std::endl;
    }

    void ustaw(const std::string& naz, double cen, const std::string& par) {
        nazwa = naz;
        cena = cen;
        parametr = par;
    }

    double pobierzCene() const override{
        return cena;
    }

private:
    std::string nazwa;
    double cena;
    std::string parametr;
};
```

- **Class Kup**
- Class Koszyk

  klasa reprezentująca koszyk zakupów.
  vector<Produkt*> produkty: Wektor przechowujący wskaźniki do produktów w koszyku.
  dodajProdukt(Produkt* produkt): Metoda dodająca produkt do koszyka.
  wyczyscKoszyk(): Metoda usuwająca wszystkie produkty z koszyka.

```
class Kup {
private:
    class Koszyk {
    public:
        void dodajProdukt(Produkt* produkt) {
            produkty.push_back(produkt);
        }

        void wyczyscKoszyk() {
            produkty.clear();
        }

        const std::vector<Produkt*>& getProdukty() const {
            return produkty;
        }

    private:
        std::vector<Produkt*> produkty;
    };
```

- koszykowyKoszt()

Oblicza i zwraca łączny koszt zakupów w koszyku.

```
double koszykowyKoszt() const {
    double koszt = 0.0;
    for (Produkt* produkt : koszyk.getProdukty()) {
        koszt += produkt->pobierzCene();
    }
    return koszt;
}
```

- czasAktualny()

Zwraca bieżącą datę i czas w formie ciągu znaków.

```
std::string czasAktualny() {
    std::time_t teraz = std::time(nullptr);
    std::tm czasTeraz;
    localtime_r(&teraz, &czasTeraz);

    char czasChar[80];
    std::strftime(czasChar, sizeof(czasChar), "%Y-%m-%d %H:%M:%S", &czasTeraz);

    return std::string(czasChar);
}
```

- dodajProduktDoKoszyka(Produkt* produkt)

Dodaje produkt do koszyka

```
void dodajProduktDoKoszyka(Produkt* produkt) {
    koszyk.dodajProdukt(produkt);
}
```

- dokonajZakupu()

Dokonuje zakupu, generuje raport, wyświetla zakupione produkty i czyści koszyk.

```
void dokonajZakupu() {
    std::ofstream raport("raport.txt", std::ios::app);

    raport << "Data zakupu: " << czasAktualny() << std::endl;
    raport << "Koszt całkowity zakupow: " << koszykowyKoszt() << " PLN" << std::endl;
    raport << "----------------------------------------" << std::endl;

    std::cout << "Kupione produkty: " << std::endl;
    std::cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << std::endl;
    for (Produkt* produkt : koszyk.getProdukty()) {
        produkt->wyswietl();
        std::cout << std::endl;
    }

    std::cout << std::endl << "Za zakupy zaplacisz: " << koszykowyKoszt() << " PLN" << std::endl;

    raport.close();
    koszyk.wyczyscKoszyk();
}
```

- wyswietlKoszyk()

Funkcja wyświetla aktualna zawartość koszyka oraz łączną kwotę produktów w koszyku

```
void wyswietlKoszyk() {
    if (koszyk.getProdukty().empty()) {
        std::cout << "Koszyk jest pusty" << std::endl;
    }
    else {
        std::cout << "Zawartosc koszyka:" << std::endl;
        for (Produkt* produkt : koszyk.getProdukty()) {
            produkt->wyswietl();
        }
        std::cout << "Laczna kwota: " << koszykowyKoszt() << " PLN" << std::endl;
    }
}
```

- **Main()**

- Tworzenie obiektów

```
Kup kupowanie;
```

- Tablice przechowujące produkty różnych kategorii

```
Wedka wedki[5]; int wedkii = 0;
Kolowrotek kolowrotki[5]; int kolowrotkii = 0;
Przyneta przynety[7]; int przynetyi = 0;
Splawik splawiki[6]; int splawikii = 0;
Zylka zylki[6]; int zylkii = 0;
Akcesoria akcesor[9]; int akcesori = 0;
```

- Otwarcie pliku produkty.txt w trybie do odczytu, odczyt danych i inicjalizacja tablic produktów

```
while (plik_wejsciowy >> n >> c >> p >> kategoria)
{

    //cout<<n<<" "<<c<<" "<<p<<" "<<kategoria<<endl;

    switch (kategoria) {
    case 1:
        wedki[wedkii].ustaw(n, c, p);
        wedkii++;
        break;

    case 2:
        kolowrotki[kolowrotkii].ustaw(n, c, p);
        kolowrotkii++;
        break;

    case 3:
        przynety[przynetyi].ustaw(n, c, p);
        przynetyi++;
        break;

    case 4:
        splawiki[splawikii].ustaw(n, c, p);
        splawikii++;
        break;

    case 5:
        zylki[zylkii].ustaw(n, c, p);
        zylkii++;
        break;

    case 6:
        akcesor[akcesori].ustaw(n, c, p);
        akcesori++;
        break;
```

- Pętla obsługująca interakcję z użytkownikiem. Wyświetla dostępne kategorie produktów. Pozwala użytkownikowi wybrać kategorię i dodawać produkty do koszyka. Pyta użytkownika, czy chce kontynuować zakupy.

```cpp
int koniec_zakupow = 0;

while (!koniec_zakupow)
{

    wyswietl_kategorie();

    std::cin >> kategoria;

    system("clear");

    if (kategoria == 1) {
        std::cout << "----------------------------------------" << std::endl;
        for (int i = 0; i < wedkii; i++)
        {
            std::cout << " " << i+1 << ".  ";
            wedki[i].wyswietl();
        }
        std::cout << "----------------------------------------" << std::endl;
        int wybieranie = -1;
        std::cout << "Ktory prokdukt dodac do koszyka: " << std::endl;
        std::cin >> wybieranie;
        kupowanie.dodajProduktDoKoszyka(&wedki[wybieranie-1]);

    }
    else if (kategoria == 2) {
        std::cout << "----------------------------------------" << std::endl;
        for (int i = 0; i < kolowrotkii; i++)
        {
            std::cout << " " << i+1 << ".  ";
            kolowrotki[i].wyswietl();

        }
        std::cout << "----------------------------------------" << std::endl;
        int wybieranie = -1;
        std::cout << "Ktory prokdukt dodac do koszyka: " << std::endl;
        std::cin >> wybieranie;
        kupowanie.dodajProduktDoKoszyka(&kolowrotki[wybieranie-1]);

    }
    else if (kategoria == 3) {
        std::cout << "----------------------------------------" << std::endl;
        for (int i = 0; i <przynetyi; i++)
        {
            std::cout << " " << i+1 << ".  ";
            przynety[i].wyswietl();
        }
        std::cout << "----------------------------------------" << std::endl;
        int wybieranie = -1;
        std::cout << "Ktory prokdukt dodac do koszyka: " << std::endl;
        std::cin >> wybieranie;
        kupowanie.dodajProduktDoKoszyka(&przynety[wybieranie-1]);

    }
    else if (kategoria == 4) {
        std::cout << "----------------------------------------" << std::endl;
        for (int i = 0; i <splawikii; i++)
        {
            std::cout << " " << i+1 << ".  ";
            splawiki[i].wyswietl();
        }
        std::cout << "----------------------------------------" << std::endl;
        int wybieranie = -1;
        std::cout << "Ktory prokdukt dodac do koszyka: " << std::endl;
        std::cin >> wybieranie;
        kupowanie.dodajProduktDoKoszyka(&splawiki[wybieranie-1]);

    }
```

```cpp
    else if (kategoria == 5) {
        std::cout << "----------------------------------------" << std::endl;
        for (int i = 0; i <zylkii; i++)
        {
            std::cout << " " << i+1 << ".  ";
            zylki[i].wyswietl();
        }
        std::cout << "----------------------------------------" << std::endl;
        int wybieranie = -1;
        std::cout << "Ktory prokdukt dodac do koszyka: " << std::endl;
        std::cin >> wybieranie;
        kupowanie.dodajProduktDoKoszyka(&zylki[wybieranie-1]);

    }
    else if (kategoria == 6) {
        std::cout << "----------------------------------------" << std::endl;
        for (int i = 0; i <akcesori; i++)
        {
            std::cout << " " << i+1 << ".  ";
            akcesor[i].wyswietl();
        }
        std::cout << "----------------------------------------" << std::endl;
        int wybieranie = -1;
        std::cout << "Ktory prokdukt dodac do koszyka: " << std::endl;
        std::cin >> wybieranie;
        kupowanie.dodajProduktDoKoszyka(&akcesor[wybieranie-1]);
    }
```

```cpp
    else if (kategoria == 7) {
        kupowanie.wyswietlKoszyk();
        std::cout << "Nacisnij Enter, aby kontynuować...";
        std::cin.ignore();
        std::cin.get();
    }

    else if (kategoria == 8) {
        koniec_zakupow = 1;
    }

    system("clear");
```

- Dokonuje zakupów na podstawie produktów w koszyku

```
kupowanie.dokonajZakupu();
```

4.Przykałdowe wywołanie programu

- Strona startowa

```
krystian@macbook-air-krystian Projekt % make run
./sklep.out
*************************************************************
*                  Sklep Wedkarski "Barwena"                *
*              Projekt obiektowej aplikacji C++               *
*
*        *
*************************************************************
```

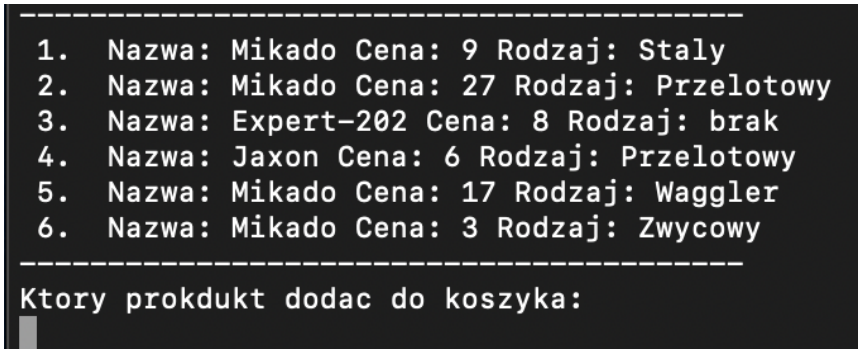
- Wybór kategorii

```
Wybierz kategorie produktu:
1. Wedki
2. Kolowrotki
3. Przynety
4. Splawiki
5. Zylki i plecionki
6. Akcesoria
7. Wyswietl koszyk
8. Zakoncz zakupy
Wybor:
```

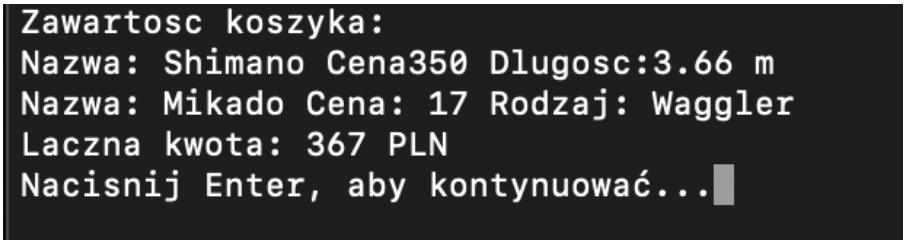
- Wybór produktu

```
----------------------------------------
 1.  Nazwa: Shimano Cena350 Dlugosc:3.66 m
 2.  Nazwa: Daiwa Cena519 Dlugosc:3 m
 3.  Nazwa: Mikado Cena207 Dlugosc:3.96 m
 4.  Nazwa: Jaxon Cena380 Dlugosc:3.5 m
 5.  Nazwa: Madcat Cena550 Dlugosc:3.23 m
----------------------------------------
Ktory prokdukt dodac do koszyka:
```

- Dalsze zakupy



- Wyświetlanie koszyka



- Zakończenie zakupów