

Sprawozdanie

1. Informacje ogólne

Temat projektu: Sklep Internetowy od strony administratora

Autorzy: Miłosz Piekarski, Krzysztof Koszyk, Krystian Ziaja

2. Wstęp

Projekt sklepu internetowego w C++ skupia się na zarządzaniu sklepem z poziomu konsoli przez administratora, wykorzystując techniki programowania obiektowego. System opiera się na klasach “Produkt” i “SprzetElektroniczny”, gdzie “Produkt” dziedziczy właściwości z klasy “SprzetElektroniczny”, pokazując zastosowanie dziedziczenia, polimorfizmu oraz klas abstrakcyjnych i wewnętrznych.

Klasa SprzetElektroniczny obejmuje podstawowe atrybuty i metody, które są rozwijane w klasie “Produkt”. Funkcje operacyjne systemu, takie jak przeglądanie, dodawanie, edycja, usuwanie produktów oraz generowanie raportów i odczyt z bazy danych, są zaimplementowane w functions.cpp i ściśle powiązane z logiką w Produkt.cpp.

Baza danych opiera się na plikach tekstowych, co pozwala na prosty, ale elastyczny sposób przechowywania informacji o produktach, kategoriach, klientach i zamówieniach. Podejście to zapewnia łatwość w obsłudze i aktualizacji danych, jednocześnie oferując praktyczne i realistyczne środowisko zarządzania sklepem internetowym.

3. Opis Implementacji

W tej sekcji skupimy się na opisanu każdej zakładki z *menu głównego naszego sklepu* które prezentuje się następująco:

```
=====
                        MENU GŁÓWNE
=====
1. Przeglądaj Produkty
2. Dodaj Produkt
3. Edytuj Produkt
4. Usuń Produkt
5. Generuj Raport
6. Odczytaj z Bazy Danych
7. Wyjście
=====

Wybierz opcję (1-7):
```

Przeglądaj Produkty

Ta podstawowa funkcjonalność pozwala na przeglądanie listy produktów znajdujących się w bazie. Możemy mieć wgląd w produkt w sposób standardowy, przez wybranie opcji nr 1. Ta część programu otwiera plik produkty.txt i pobiera liniami poszczególne produkty. Przy pomocy funkcji split - zaimplementowanej w pliku fileHelper.cpp - tworzony jest vector przechowujący dany produkt. Następnie inicjalizowany jest przez konstruktor obiekt klasy produkt i wywoływana jest na nim metoda wypisz, która pozwala na wyświetlenie danego produktu. Czynność powtarzana jest do momentu, aż wszystkie produkty nie zostaną zaczytane i wypisane. Aby schludnie wyświetlić produkt w konsoli, stworzono metodę formatuj, która pozwala na odpowiednie dopasowanie wielkości stringa z nazwą. Korzysta ona z metody ilePolskich, która sprawdza ilość polskich znaków w danej nazwie. Pomaga to w obliczeniu odpowiedniej ilości znaków potrzebnych do sformatowania wyświetlanej nazwy.

Implementacja opcji 1:

```
case 1:
{
    std::cout << "Wyświetlanie wszystkich produktów...\n";
    fstream file = OpenFile( path: "produkty.txt");
    string line;
    getline( &: file, &: line);

    cout << "ID" << " | Nazwa" << setw( n: 123) << " | Cena \t | Dostepnosc \t| Dodatkowe informacje\n" << endl;
    while(getline( &: file, &: line))
    {
        vector<string> productV = split( s: line, delim: ';');
        Produkt* p = new Produkt( id_p: stoi( str: productV[0]), id_kategorii_p: stoi( str: productV[1]), Nazwa_p: productV[2],
        p->wypisz();
    }

    file.close();
}
break;
```

Funkcje pomocnicze z pliku fileHandler.cpp:

```
fstream OpenFile(string path)
{
    string fullPath = "..\\dane\\" + path;
    fstream file;
    file.open( s: fullPath);
    return file;
}

vector<string> split(const string &s, char delim) {
    vector<std::string> result;
    stringstream ss ( str: s);
    string item;

    while (getline ( &: ss, &: item, delim)) {
        result.push_back (item);
    }

    return result;
}
```

Metoda wypisz z klasy Produkt:

```
void Produkt::wypisz()
{
    // cout << IDs.Id << " | " << Nazwa << " | ";
    // printf("%.2f", Cena );
    // cout << " \t\t\t" << Dostepnosc << " \t\t\t" << CechaSzczegolna << endl;

    string nazwa = this->formatujNazwe();
    cout << IDs.Id << " | " << nazwa << " | " << Cena << "\t | " << Dostepnosc << "\t\t\t" << CechaSzczegolna << endl;
}
```

Metoda formatujNazwe:

```
// wyplenienie białymi znakami w celu schludniejszego wyświetlenia w konsoli
string Produkt::formatujNazwe()
{
    string nazwaf = Nazwa;
    int len = 0;
    len = Nazwa.length();
    len = len - ilePolskich(str: Nazwa);
    int n = 0;
    if(IDs.Id < 10)
    {
        n = 81;
    }
    else
    {
        n = 80;
    }
    for(int j = len; j < n; j++)
    {
        nazwaf += " ";
    }
    return nazwaf;
}
```

Funkcja ilePolskich:

```
int ilePolskich(const string str) {
    // Convert narrow string to wide string
    wstring_convert<std::codecvt_utf8<wchar_t>> converter;
    wstring wideStr = converter.from_bytes(str);

    // Define a set of Polish characters
    wstring polishCharacters = L"ąęłńóśźżĄĆĘŁŃÓŚŻŻ";

    // Count occurrences of Polish characters
    int count = 0;
    for (wchar_t ch : wideStr) {
        if (polishCharacters.find(c: ch) != wstring::npos) {
            count++;
        }
    }

    return count;
}
```

Funkcjonalność 2 - Sortowanie produktu pozwala wybrać właściwość, według której zostaną posortowane produkty. Następnie przy pomocy wbudowanej funkcji `sort` sortowany jest wektor z produktami według wybranego kryterium i wypisywane są wszystkie posortowane produkty, podobnie jak we wcześniejszej opcji.

Wybranie kategorii sortowania:

```
std::cout << "Sortowanie produktow...\n";
fstream file = OpenFile(path: "produkty.txt");
string line, elementy[6];
getline(&file, &line);

vector<string> Headers = split(s: line, delim: ';');

int wybor = -1;
while(wybor < 1 || wybor > Headers.size())
{
    cout << "Sortuj według kategorii: " << endl;
    for(int i = 2; i < Headers.size(); i++)
    {
        cout << (i - 1) << ". " << Headers[i] << endl;
    }
    cin >> wybor;
}
```

Sortowanie według wybranej kategorii:

```
while(getline( &: file, &: line))
{
    vector<string> productV = split( s: line, delim: ';' );
    Produkt* p = new Produkt( id_p: stoi( str: productV[0]), id_kategorii_p: stoi( str: productV[1]), Nazwa_p: productV[2],
    allProducts.push_back(p);
}

if(wybor == 2)
{
    sort( first: allProducts.begin(), last: allProducts.end(),
        comp: [wybor](const Produkt* a, const Produkt* b) -> bool {
            return a->getCena() < b->getCena();
        });
}

else if(wybor == 1)
{
    sort( first: allProducts.begin(), last: allProducts.end(),
        comp: [wybor](const Produkt* a, const Produkt* b) -> bool {
            return a->getNazwa() < b->getNazwa();
        });
}

else if(wybor == 3)
{
    sort( first: allProducts.begin(), last: allProducts.end(),
        comp: [wybor](const Produkt* a, const Produkt* b) -> bool {
            return a->getDostepnosc() < b->getDostepnosc();
        });
}

else if(wybor == 4)
{
    sort( first: allProducts.begin(), last: allProducts.end(),
        comp: [wybor](const Produkt* a, const Produkt* b) -> bool {
            return a->getCechaSzczegolna() < b->getCechaSzczegolna();
        });
}
```

Funkcjonalność 3 - Pozwala na wyświetlenie produktów z wybranej kategorii. Kategorie są wyświetlane z pliku kategorie.txt. Użytkownik musi wybrać daną kategorię i następnie otrzymuje listę tylko tych produktów, które są związane z daną kategorią.

Wyświetlenie wszystkich dostępnych kategorii i wybór jednej z nich:

```
std::cout << "Filtrowanie produktów...\n";

fstream file = OpenFile(path: "kategorie.txt");

string line;
getline(&file, &line);

vector<int> categoryId;
while(getline(&file, &line))
{
    vector<string> category = split(s: line, delim: ';');

    cout << category[0] << ". " << category[1] << endl;
    categoryId.push_back(stoi(str: category[0]));
}
file.close();

int wybor = -1;
while(wybor < 1 || wybor > categoryId[categoryId.size() - 1])
{
    cout << "Wybierz kategorie: (1 - " << categoryId.size() << ") " << "\n";
    cin >> wybor;
}
wybor--;
```

Wyświetlanie produktów tylko z danej kategorii:

```
vector<Produkt*> allProducts;

while(getline(&file2, &line2))
{
    vector<string> productV = split(s: line2, delim: ';');
    Produkt* p = new Produkt(id_p: stoi(str: productV[0]), id_kategorii_p: stoi(str: productV[1]), Nazwa_p: productV[2],
    if(p->getId_Kategorii() == categoryId[wybor])
    {
        allProducts.push_back(p);
    }
}
for(Produkt* p : allProducts)
{
    p->wypisz();
}

file2.close();
```

Przykład działania opcji 1:

```
Wprowadź swój wybór (1-7): 1
Menu Przeglądania Produktów:
1. Wyświetl wszystkie produkty
2. Sortuj produkty
3. Filtrowanie produktów
4. Powrót do Menu Głównego
Wprowadź swój wybór (1-4):1
Wyświetlanie wszystkich produktów...
ID| Nazwa | Cena | Dostepnosc | Dodatkowe informacje
1 | Laptop Dell XPS 13 | 4500.00 | 20 | Outlet
2 | Laptop HP Spectre x360 | 3800.00 | 15 | Nowy
3 | iPhone 13 Pro | 4999.00 | 30 | Outlet
4 | Samsung Galaxy S21 | 3499.00 | 25 | Outlet
5 | Mysz bezprzewodowa Logitech MX Master 3 | 99.99 | 50 | Laserowa
6 | Mysz przewodowa A4 tech | 19.99 | 100 | Optyczna
7 | Słuchawki bezprzewodowe Sony WH-1000XM4 | 899.00 | 15 | Bluetooth 5
8 | Słuchawki douszne JBL Free X | 129.99 | 30 | MiniJack
9 | Kamera internetowa Logitech C920 | 199.00 | 10 | 3Mpx
10 | Kamera sportowa GoPro HERO10 | 549.00 | 20 | 5Mpx
11 | Telewizor Samsung QLED Q80A | 3299.00 | 5 | Full HD
12 | Telewizor LG OLED C1 | 4999.00 | 8 | 4K
13 | Monitor Dell Ultrasharp U2719D | 999.00 | 12 | 60 Hz
14 | Monitor HP Pavilion 27xw | 649.00 | 18 | 144 Hz
15 | Smartwatch Apple Watch Series 7 | 1799.00 | 10 | Wodoodporny
16 | Smartwatch Samsung Galaxy Watch 4 | 699.00 | 15 | Wodoodporny
17 | Głośniki Creative Pebble 2.0 | 29.99 | 25 | MiniJack
18 | Głośniki Bluetooth JBL Flip 5 | 119.99 | 20 | Bluetooth 5.1
19 | Klawiatura mechaniczna Razer BlackWidow Elite | 169.99 | 7 | Red
20 | Klawiatura Logitech K380 | 39.99 | 30 | Brown
```

Przykład działania opcji 2:

```
Wprowadź swój wybór (1-7): 1
Menu Przeglądania Produktów:
1. Wyświetl wszystkie produkty
2. Sortuj produkty
3. Filtrowanie produktów
4. Powrót do Menu Głównego
Wprowadź swój wybór (1-4):2
Sortowanie produktow...
Sortuj według kategorii:
1. Nazwa
2. Cena
3. Dostępność
4. Cecha
2
ID| Nazwa | Cena | Dostepnosc | Dodatkowe informacje
6 | Mysz przewodowa A4 tech | 19.99 | 100 | Optyczna
17 | Głośniki Creative Pebble 2.0 | 29.99 | 25 | MiniJack
20 | Klawiatura Logitech K380 | 39.99 | 30 | Brown
5 | Mysz bezprzewodowa Logitech MX Master 3 | 99.99 | 50 | Laserowa
18 | Głośniki Bluetooth JBL Flip 5 | 119.99 | 20 | Bluetooth 5.1
8 | Słuchawki douszne JBL Free X | 129.99 | 30 | MiniJack
19 | Klawiatura mechaniczna Razer BlackWidow Elite | 169.99 | 7 | Red
9 | Kamera internetowa Logitech C920 | 199.00 | 10 | 3Mpx
10 | Kamera sportowa GoPro HERO10 | 549.00 | 20 | 5Mpx
14 | Monitor HP Pavilion 27xw | 649.00 | 18 | 144 Hz
16 | Smartwatch Samsung Galaxy Watch 4 | 699.00 | 15 | Wodoodporny
7 | Słuchawki bezprzewodowe Sony WH-1000XM4 | 899.00 | 15 | Bluetooth 5
13 | Monitor Dell Ultrasharp U2719D | 999.00 | 12 | 60 Hz
15 | Smartwatch Apple Watch Series 7 | 1799.00 | 10 | Wodoodporny
11 | Telewizor Samsung QLED Q80A | 3299.00 | 5 | Full HD
4 | Samsung Galaxy S21 | 3499.00 | 25 | Outlet
2 | Laptop HP Spectre x360 | 3800.00 | 15 | Nowy
1 | Laptop Dell XPS 13 | 4500.00 | 20 | Outlet
12 | Telewizor LG OLED C1 | 4999.00 | 8 | 4K
3 | iPhone 13 Pro | 4999.00 | 30 | Outlet
```

Przykład działania opcji 3:


```

Wprowadź swój wybór (1-7): 1
Menu Przeglądania Produktów:
1. Wyświetl wszystkie produkty
2. Sortuj produkty
3. Filtrowanie produktów
4. Powrót do Menu Głównego
Wprowadź swój wybór (1-4):3
Filtrowanie produktów...
1. Laptopy
2. Smartfony
3. Myszki
4. Słuchawki
5. Kamery
6. Telewizory
7. Monitory
8. Smartwatche
9. Głośniki
10. Klawiatury
Wybierz kategorię: (1 - 10)
9
17 | Głośniki Creative Pebble 2.0 | 29.99 | 25 | MiniJack
18 | Głośniki Bluetooth JBL Flip 5 | 119.99 | 20 | Bluetooth 5.1

```

Dodaj Produkt

Ta opcja programu pozwala na dodanie nowego produktu do istniejącej już listy. Zadaniem użytkownika jest podać kategorię produktu z dostępnych w pliku `kategorie.txt`, a następnie uzupełnić informacje o produkcie. Podstawową walidację wpisanych danych realizują wyrażenia regularne (regex). Sprawdzają czy użytkownik podał poprawną cenę i dostępność produktu. Jeśli wszystkie dane są wpisane poprawnie, produkt zostaje dodany do listy. Realizowane jest to za pomocą metod `uzupelnijDane` i `dodajProdukt`.

Funkcje sprawdzające poprawność danych:

```

bool sprawdzCene(string C)
{
    string r = "^[1-9][0-9]*.{1}[0-9]{2}$";
    if(regex_match(s: C, re: regex(s: r)))
    {
        return true;
    }
    else
    {
        return false;
    }
}

bool sprawdzDostepnosc(string D)
{
    string r = "^[0-9]*$";
    if(regex_match(s: D, re: regex(s: r)))
    {
        return true;
    }
    else
    {
        return false;
    }
}

```

Metoda uzupełnij dane przydziela następne wolne Id produktowi, prosi użytkownika o wybranie kategorii i realizuje uzupełnienie reszty potrzebnych danych z uwzględnieniem ich poprawności.

Przydzielenie Id:

```

//Przydzielenie następnego Id produktu na podstawie produktów w pliku
fstream file = OpenFile( path: "produkty.txt");
string line;
while(getline(&: file, &: line))
{
    //przejsie do ostatniej linijki
}
Idp = stoi( str: line.substr( pos: 0, n: line.find( c: ';' ))) + 1;

```

Uzupełnienie danych:

```

try
{
    this->IDs.Id = Idp;
    this->IDs.Id_Kategorii = Idk;

    string cenaP, dostepnosc;
    std::cin.clear();
    std::cin.ignore(n: std::numeric_limits<std::streamsize>::max(), delim: '\n');
    cout << "Podaj nazwe produktu:\n";
    getline( &: cin, &: this->Nazwa);
    cout << "Podaj cene produktu: (np.: 220.99)\n";
    getline( &: cin, &: cenaP);
    while(!sprawdzCene( C: cenaP))
    {
        cout << "Podaj poprawna cene produktu: (np.: 220.99)\n"; //Cena jako float w postaci XXXX.XX
        getline( &: cin, &: cenaP);
    }
    this->Cena = cenaP;
    cout << "Podaj dostepnosc produktu:\n";
    getline( &: cin, &: dostepnosc);
    while(!sprawdzDostepnosc( D: dostepnosc))
    {
        cout << "Podaj poprawna dostepnosc produktu:\n"; // int
        getline( &: cin, &: dostepnosc);
    }
    this->Dostepnosc = stoi( str: dostepnosc);
    cout << "Podaj ceche produktu:\n";
    getline( &: cin, &: this->CechaSzczegolna);
}

```

Metoda dodaj produkt realizuje wpis do pliku obiektu, który został zaopatrzony w dane jak wyżej. Uwzględnia ona przyjętą przez nas konwencje przetrzymywania danych w pliku:

```

bool Produkt::dodajProdukt() {
    fstream plik = OpenFile( path: "produkty.txt");
    plik.seekp(0, ios_base::end);
    plik.seekg(0, ios_base::end);
    plik << "\n";
    string produkt = to_string( val: this->IDs.Id) + ";" + to_string( val: this->IDs.Id_Kategorii) + ";" +
    plik << produkt;
    plik.close();

    return true;
}

```

Przykład działania:

```
Wprowadź swój wybór (1-7): 2
Menu Dodawania Produktu:
1. Dodaj nowy produkt
2. Powrót do Menu Głównego
Wprowadź swój wybór (1-2): 1
1. Laptopy
2. Smartfony
3. Myszki
4. Słuchawki
5. Kamery
6. Telewizory
7. Monitory
8. Smartwatche
9. Głośniki
10. Klawiatury
Podaj numer kategorii produktu
1
Podaj nazwę produktu:
Lenovo Legion 5
Podaj cenę produktu: (np.: 220.99)
2222.aa
Podaj poprawną cenę produktu: (np.: 220.99)
2222.00
Podaj dostępność produktu:
14
Podaj cechę produktu:
Nowy
21 | Lenovo Legion 5 | 2222.00 | 14 | Nowy

Menu Dodawania Produktu:
1. Dodaj nowy produkt
2. Powrót do Menu Głównego
Wprowadź swój wybór (1-2): 2
```

Sprawdzenie poprawności dodawania:

```

Menu Głównie:
1. Przeglądaj Produkty
2. Dodaj Produkt
3. Edytuj Produkt
4. Usuń Produkt
5. Generuj Raport
6. Odczytaj z Bazy Danych
7. Wyjście

Wprowadź swój wybór (1-7): 1
Menu Przeglądania Produktów:
1. Wyświetl wszystkie produkty
2. Sortuj produkty
3. Filtrowanie produktów
4. Powrót do Menu Głównego
Wprowadź swój wybór (1-4):3
Filtrowanie produktów...
1. Laptopy
2. Smartfony
3. Myszki
4. Słuchawki
5. Kamery
6. Telewizory
7. Monitory
8. Smartwatche
9. Głośniki
10. Klawiatury
Wybierz kategorie: (1 - 10)
1
1 | Laptop Dell XPS 13 | 4500.00 | 20 | Outlet
2 | Laptop HP Spectre x360 | 3800.00 | 15 | Nowy
21 | Lenovo Legion 5 | 2222.00 | 14 | Nowy

```

Jak widać produkt po dodaniu został wyświetlony poprawnie.

Edytuj Produkt

Plik Produkt.cpp implementuje funkcje związane z produktami, w tym możliwość edycji produktu. Funkcja “menuEdycjiProduktu” pozwala na modyfikację atrybutów produktu, takich jak nazwa, cena, dostępność i specjalne cechy. Funkcja ta jest metodą klasy “Produkt” co oznacza, że działa na atrybutach konkretnego obiektu tej klasy. Pozwala to na modyfikację stanu obiektu w bezpieczny i kontrolowany sposób.

```

void menuEdycjiProduktu() {
    while (true) {
        std::cout << "Podaj ID produktu do edycji (lub 0, aby powrócić do menu głównego): ";
        int idProduktu;
        std::cin >> idProduktu;
        std::cin.ignore();

        if (idProduktu == 0) {
            std::cout << "Powrót do menu głównego." << std::endl;
            break;
        }

        Produkt produkt;
        if (!produkt.wczytajProdukt(idProduktu)) {
            std::cout << "Produkt o podanym ID nie istnieje." << std::endl;
            continue;
        }
    }
}

```

Na początku funkcja pyta nas o ID Produktu, a następnie przechodzi do pliku Produkt.cpp gdzie znajduje się metoda “wczytajProdukt” która na podstawie ID Produktu szuka w bazie danych danego produktu i zwraca *true* lub *false*

```

bool Produkt::wczytajProdukt(int idProduktu) {
    std::string sciezkaDoPliku = "../dane/produkty.txt"; // Ustawiamy względną ścieżkę do folderu 'dane'
    std::fstream plik( s: sciezkaDoPliku);

    if (!plik.is_open()) {
        std::cerr << "Nie można otworzyć pliku produkty.txt" << std::endl;
        return false;
    }

    std::string linia;
    while (getline( &: plik, &: linia)) {
        std::stringstream liniaStream(str: linia);
        std::string segment;
        std::vector<std::string> segmenty;
    }
}

```

Jeśli ta metoda zwraca *true* to funkcja “menuEdycjiProduktu” przyjmuje od użytkownika nowe dane do edycji

```

// Zbieranie nowych danych produktu od użytkownika
std::string nowaNazwa, nowaCena, nowaDostepnosc, nowaCecha;
std::cout << "Podaj nową nazwę produktu (obecna: " << produkt.getNazwa() << "): ";
getline( &: std::cin, &: nowaNazwa);
nowaNazwa = nowaNazwa.empty() ? produkt.getNazwa() : nowaNazwa;

```

a następnie zostaje wywołana kolejna metoda “edytujProdukt” która zamienia dane

```
bool Produkt::edytujProdukt(int idProduktu, const string& nowaNazwa, const string& nowaCena, const string& nowaDostepnosc,
vector<string> linie;
string linia;
ifstream plikWe( s: "../dane/produkty.txt");
bool znaleziono = false;

while (getline( &: plikWe, &: linia)) {
    stringstream ss( str: linia);
    string itemId;
    getline( &: ss, &: itemId, delim: ';');
    itemId.erase( first: remove_if( first: itemId.begin(), last: itemId.end(), pred: ::isspace), last: itemId.end());

    if (!itemId.empty() && std::all_of( first: itemId.begin(), last: itemId.end(), pred: ::isdigit) && stoi( str: itemId) ==
        znaleziono = true;
        string zmodyfikowanaCena = nowaCena.empty() ? this->Cena : nowaCena;
        string zmodyfikowanaDostepnosc = nowaDostepnosc.empty() ? std::to_string( val: this->Dostepnosc) : nowaDostepnosc;

        string zmodyfikowanaLinia = to_string( val: idProduktu) + ";" + to_string( val: IDs.Id_Kategorii) + ";" + nowaNazwa;
        linie.push_back(zmodyfikowanaLinia);
        continue;
    }
```

Przykład działania programu:

```
Menu Głównie:
1. Przeglądaj Produkty
2. Dodaj Produkt
3. Edytuj Produkt
4. Usuń Produkt
5. Generuj Raport
6. Odczytaj z Bazy Danych
7. Wyjście

Wprowadź swój wybór (1-7): 3
Podaj ID produktu do edycji (lub 0, aby powrócić do menu głównego): 13
ID | Nazwa | Cena | Dostepnosc | Dodatkowe informacje
13 | Monitor Dell Ultrasharp U2719D | 999.00 | 12 | 60 Hz
Czy na pewno chcesz edytować ten produkt? (tak/nie): tak
Podaj nową nazwę produktu (obecna: Monitor Dell Ultrasharp U2719D): Monitor MSI MAG271CQR
Podaj nową cenę produktu (obecna: 999): 1452
Podaj nową dostępność produktu (obecna: 12): 4
Podaj nową cechę szczególną produktu (obecna: 60 Hz): 144
Produkt został pomyślnie zaktualizowany.
```

Usuń Produkt

W celu usuwania produktów została stworzona metoda “usunProdukt” w pliku Produkt.cpp która demonstuje obiektowe podejście do usuwania produktu z bazy danych sklepu internetowego. Zastosowanie obiektowości: funkcja “menuUsuwanieProduktu” jest częścią klasy “Produkt”, co oznacza, że operuje na danych i zachowaniach związanych z obiektami tej klasy.

Na początku funkcja prosi nas o ID Produktu

```
void menuUsunanieProduktu() {
    int idProduktu;
    cout << "Podaj ID produktu do usunięcia (lub 0 aby powrócić do menu głównego): ";
    cin >> idProduktu;

    // Jeśli użytkownik wpisał 0, powrót do menu głównego
    if (idProduktu == 0) {
        cout << "Powrót do menu głównego." << endl;
        return;
    }

    Produkt produkt;
    if (!produkt.usunProdukt(idProduktu)) {
        cout << "Nie udało się usunąć produktu." << endl;
    }
}
```

następnie przechodzimy do metody “usunProdukt” a następnie wyszukuje odpowiedni produkt i zapisuje informacje o nim po czym zostaje usunięty.

```
bool Produkt::usunProdukt(int idProduktu) {
    vector<string> linie;
    string linia;
    ifstream plikWe("s://dane/produkty.txt");
    bool znaleziono = false;
    string produktDoUsuniecia;

    if (getline(&plikWe, &linia)) {
        linie.push_back(linia); // Zachowaj nagłówek
    }

    while (getline(&plikWe, &linia)) {
        stringstream ss(str(linia));
        string itemId;
        getline(&ss, &itemId, delim: ';');
        itemId.erase(first: remove_if(first: itemId.begin(), last: itemId.end(), pred: ::isspace), last: itemId.end());

        if (!itemId.empty() && std::all_of(first: itemId.begin(), last: itemId.end(), pred: ::isdigit) && stoi(str(itemId)) == idProduktu)
            znaleziono = true;
        produktDoUsuniecia = linia; // Zapisz informacje o produkcie do usunięcia
        continue;
    }
    linie.push_back(linia);
}
plikWe.close();
```

Przykład działania programu:


```
Menu Głównie:
1. Przeglądaj Produkty
2. Dodaj Produkt
3. Edytuj Produkt
4. Usuń Produkt
5. Generuj Raport
6. Odczytaj z Bazy Danych
7. Wyjście

Wprowadź swój wybór (1-7): 4
Podaj ID produktu do usunięcia (lub 0 aby powrócić do menu głównego): 9
ID produktu: 9
ID kategorii: 5
Nazwa: Kamera internetowa Logitech C920
Cena: 199.00
Dostępność: 10
Cecha: 3Mpx
Czy na pewno chcesz usunąć ten produkt? (tak/nie): tak
Produkt został pomyślnie usunięty.
```

Generuj Raport

Aby użytkownik programu miał możliwość sprawdzenia danych statystycznych jego sklepu, ma również dostęp do opcji “Generuj Raport”. Tworzy ona raport, który w zależności od preferencji użytkownika będzie brała pod uwagę jeden zakres spośród 7, 31 lub 365 dni. Na końcu programu od użytkownika zależy także czy wygenerowany przez niego raport ma zostać zapisany czy też nie.

```

void menuGenerowaniaRaportu() {
    int wybor;
    do {
        std::cout << "Menu Generowania Raportu:\n";
        std::cout << "1. Wybierz zakres czasowy dla raportu\n";
        std::cout << "2. Powrót do Menu Głównego\n";
        std::cout << "Wprowadź swój wybór (1-2): ";
        std::cin >> wybor;

        switch (wybor) {
            case 1: {
                int wybor_terminu;
                std::cout<<"Wybierz zakres, dla którego chcesz wygenerować raport \n";
                std::cout << "1. Tydzień\n";
                std::cout << "2. Miesiąc\n";
                std::cout << "3. Rok\n";
                std::cout << "\n\n0. Powrót.";
                std::cin>>wybor_terminu;
                while(wybor_terminu<1 || wybor_terminu>3){
                    std::cout<<"Nieprawidłowy wybór. Proszę wybrać liczbę od 1 do 3";
                    std::cin>>wybor_terminu;
                }
                Raport raport_do_zapisu;
                switch(wybor_terminu){
                    case 1: {
                        raport_do_zapisu = generuj_raport( wybor: 7, wybor_str: "Tydzień");
                        break;
                    }
                    case 2: {
                        raport_do_zapisu = generuj_raport( wybor: 31, wybor_str: "Miesiąc");
                        break;
                    }
                    case 3: {
                        raport_do_zapisu = generuj_raport( wybor: 365, wybor_str: "Rok");
                        break;
                    }
                }
            }
        }
    }
}

```

Gdy użytkownik wybierze jedną spośród opcji czasowych zostaje wywołana funkcja "generuj_raport"

```

Raport generuj_raport(int wybor, std::string wybor_str){
    Zamowienie::wczytaj_zamowienia();
    vector<Zamowienie> zamowienia_do_raportu;
    zamowienia_do_daty(wybor, &zamowienia_do_raportu);
    map<int, Specyfikacja> Statystyka;
    Oblicz_przychod(zamowienia_do_raportu, &Statystyka);
    Specyfikacja *najlepsi = new Specyfikacja[3];
    float suma = 0;
    for(auto i: Statystyka.begin(); i != Statystyka.end(); i++){
        if(i->second.ilosc > najlepsi[2].ilosc){
            if(i->second.ilosc > najlepsi[1].ilosc){
                if(i->second.ilosc > najlepsi[0].ilosc){
                    najlepsi[2] = najlepsi[1];
                    najlepsi[1] = najlepsi[0];
                    najlepsi[0] = i->second;
                }else{
                    najlepsi[2] = najlepsi[1];
                    najlepsi[1] = i->second;
                }
            }else{
                najlepsi[2] = i->second;
            }
            suma += i->second.cena;
        }
    }

    std::cout<<"Raport za ostatnich "<<wybor<<" dni:\n";
    std::cout<<"Najlepiej sprzedajace sie produkty:\n";
    for(int i=0; i<3; i++){
        najlepsi[i].ilosc > 0 ? std::cout<<i+1<<" "<<najlepsi[i].nazwa<<" -- "<<najlepsi[i].ilosc<<" sprzedanych sztuk\n":std::cout<<" ";
    }
    std::cout<<"Lacny obrot w tym okresie wynosil "<<suma<<" zlotych\n";
    Raport::wczytaj_raporty();
    Raport g( p_zakres_czasowy: wybor_str, p_najlepszy_produkt: najlepsi[0].nazwa, p_ID_produktu: najlepsi[0].id, p_dochod: suma);
    return g;
}

```

Jest ona odpowiedzialna za tworzenie raportu, funkcja statyczna "Zamowienie::wczytaj_zamowienia" wczytuje wszystkie zamowienia do statycznej listy "lista_zamowien", z której w funkcji "zamowienia_do_daty" wybierane będą te zamowienia, których odstęp czasowy od dnia, w którym generowany jest raport jest mniejsza bądź równa wybranemu.

```

void zamowienia_do_daty(int wybor, std::vector<Zamowienie> &zamowienia_do_raportu){
    for(Zamowienie el: Zamowienie::lista_zamowien){
        if(el.roznica() <= wybor) zamowienia_do_raportu.push_back(el);
    }
}

```

Różnica ta jest obliczana w metodzie klasy "Zamowienia", w której data ma swoją klasę wewnętrzną

```

int Zamowienie::roznica() {
    date dzis;
    int roznica_dzien = 0;
    int roznica_miesiac = 0;
    int roznica_rok = dzis.podaj( tryb: 3) - this->data_zamowienia->podaj( tryb: 3);
    if (roznica_rok > 0) {
        roznica_miesiac = roznica_rok * 12;
    }
    roznica_miesiac = roznica_miesiac + (dzis.podaj( tryb: 2) - this->data_zamowienia->podaj( tryb: 2));
    if (roznica_miesiac > 0) {
        roznica_dzien = floor( X: roznica_miesiac * 30.5);
    }
    roznica_dzien = roznica_dzien + (dzis.podaj( tryb: 1) - this->data_zamowienia->podaj( tryb: 1));
    return roznica_dzien;
}

```

Następnie nasza funkcja zajmuje się właściwymi danymi statystycznymi, oblicza ile sztuk danego produktu zostało sprzedane oraz oblicza wygenerowany przez niego przychód.

```

void Oblicz_przychod(std::vector<Zamowienie> zamowienia_do_raportu, map<int,Specyfikacja> &value){
    Produkt::wczytaj_produkty();
    for(Zamowienie el: zamowienia_do_raportu){
        value[el.zwroc_id()].ilosc += el.zwroc_ilosc();
    }
    for(Produkt el: Produkt::lista_produkтов){
        try{
            int index = el.GetId();
            value.at( k: index);
            value[index].id = index;
            value[index].cena = (float)value[index].ilosc * el.getCena();
            value[index].nazwa = el.getNazwa();
        }
        catch(const exception &e){}
    }
}

```

Na koniec funkcja szuka 3 najlepiej sprzedających się produktów oraz informuje użytkownika o łącznym wygenerowanym przez sklep przychodzie. Funkcja również tworzy oraz zwraca obiekt raportu, który zawiera wszystkie zdobyte informacje.

```

    char odp;
    std::cout<<"Czy chcesz zapisac raport? (Y/N): ";
    std::cin>>odp;
    if(odp == 'Y'){
        raport_do_zapisu.dopisz_do_pliku();
    }
    break;
}
case 2:
    break; // Powrót do menu głównego
default:
    std::cout << "Nieprawidłowy wybór. Proszę wprowadzić liczbę od 1 do 2.\n";
    break;
}
std::cout << "\n\n";
} while (wybor != 2);
}

```

Zapisywany do pliku raport zawiera także swoje unikalne id oraz date, kiedy dokładnie został wygenerowany.

Odczytaj z Bazy Danych

Ostatnią opcją możliwą do wywołania jest opcja odczytu bazy danych. Pozwala ona użytkownikowi odczytać zawartość przechowywanych w bazie danych. W pierwszym kroku pyta ona użytkownika jaką bazę chce odczytać

```
void menuOdczytuZBazyDanych() {
    int wybor;
    do {
        std::cout << "Menu Odczytu z Bazy Danych:\n";
        std::cout << "1. Produkty\n";
        std::cout << "2. Kategorie\n";
        std::cout << "3. Zamówienia\n";
        std::cout << "4. Klienci\n";
        std::cout << "5. Raporty\n";
        std::cout << "6. Powrót do Menu Głównego\n";
        std::cout << "Wprowadź swój wybór (1-6): ";
        std::cin >> wybor;

        switch (wybor) {
            case 1: {
                std::cout << "Odczyt z tabeli Produkty...\n";
                Produkt::wczytaj_produkty();
                Produkt::wypisz_liste();
                std::cout<<"\nKliknij dowolny przycisk by kontynuować...";
                system( Command: "pause");
                break;
            }
            case 2:{
                std::cout << "Odczyt z tabeli Kategorie...\n";
                kategoria::wczytaj_kategorie();
                std::cout<<"Id Kategorii | Nazwa \n";
                kategoria::wypisz_liste();
                std::cout<<"\nKliknij dowolny przycisk by kontynuować...";
                system( Command: "pause");
                break;
            }
        }
    }
}
```

```

case 3: {
    std::cout << "Odczyt z tabeli Zamowienia...\n";
    Zamowienie::wczytaj_zamowienia();
    std::cout<<"Id Zamowienia | Data Zamowienia "<<"| Id Klienta | Id Produktu | Ilosc\n";
    Zamowienie::wypisz_liste();
    std::cout<<"\nKliknij dowolny przycisk by kontynuowac...";
    system( Command: "pause");
    break;
}
case 4: {
    std::cout << "Odczyt z tabeli Klienci...\n";
    Klient::wczytaj_klientow();
    std::cout<<"Id Klienta | Imie | Nazwisko | Miasto | Adres | Email\n";
    Klient::wypisz_liste();
    std::cout<<"\nKliknij dowolny przycisk by kontynuowac...";
    system( Command: "pause");
    break;
}
case 5: {
    std::cout << "Odczyt z tabeli Raporty...\n";
    Raport::wczytaj_raporty();
    std::cout<<"Id Raportu | Data Generowania | Zakres Czasowy | Najlepszy Produkt | Id Produktu | Dochód \n";
    Raport::wypisz_liste();
    std::cout<<"\nKliknij dowolny przycisk by kontynuowac...";
    system( Command: "pause");
    break;
}
case 6:
    break; // Powrót do menu głównego
default:
    std::cout << "Nieprawidłowy wybór. Proszę wprowadzić liczbę od 1 do 6.\n";
    break;
}
std::cout << "\n\n";
} while (wybor != 6);

```

Następnie przy użyciu metod statycznych wewnątrz naszych klas, wczytujemy nasze dane do statycznej listy, które następnie w metodzie “wypisz_liste()” zostają wypisane na ekran.

4. Wnioski

Projekt skutecznie demonstruje zastosowanie kluczowych koncepcji programowania obiektowego. Dziedziczenie, polimorfizm, oraz wykorzystanie klas abstrakcyjnych i wewnętrznych w klasach „Produkt” i „SprzętElektroniczny” pokazuje, jak można modelować złożone relacje i struktury w rzeczywistych aplikacjach. To podejście pozwala na modularność, łatwość w rozbudowie i utrzymaniu kodu.

Podział funkcjonalności systemu na oddzielne moduły, takie jak plik functions.cpp i Produkt.cpp, ułatwia zarządzanie kodem i jego rozwój. Modułowa architektura sprawia, że system jest bardziej zrozumiały,

łatwiejszy w debugowaniu i utrzymaniu, co jest kluczowe dla skuteczności projektu.

Projekt posiada znaczący potencjał do dalszego rozwoju. Możliwości rozszerzenia funkcjonalności, takie jak integracja z zaawansowanymi systemami baz danych, rozbudowa interfejsu użytkownika, czy implementacja dodatkowych funkcji biznesowych, mogą jeszcze bardziej zwiększyć jego wartość