**Постановка задачи ЭЛЕКТРОННАЯ КАССА**

**Общая постановка**

Разработать программу «Электронная касса». Программа должна имитировать работу кассового аппарата по сканированию товаров и формированию чека за покупку. Каждый товар идентифицируется штрих-кодом. Один и тот же товар может сканироваться несколько раз, но в чек информация о каждом товаре входит в виде «наименование – стоимость за единицу (для упрощения в рублях без копеек) – количество – общая стоимость за товар». Чек состоит не менее чем из одной записи указанного вида. Чек дополнительно включает общую стоимость товаров в покупке, суммарную скидку и итоговую сумму к оплате (все в рублях). **Каждый товар описывается штрих-кодом, наименованием, стоимостью за единицу товара, скидкой в процентах от стоимости.** Скидки устанавливаются на каждый товар независимо (в диапазоне от 1 до 50%). Программа должна предоставлять следующие операции:

1) «сканировать» очередной товар,

2) вывести описание отсканированного товара,

3) добавить данные о товаре в чек,

4) сформировать чек за покупку,

5) рассчитать итоговую сумму к оплате.

**Этап 1.**

**Разработать систему классов для общей постановки задачи.**

Разработка этапа.

1. Выделение из текста постановки набора класса.
2. Определение необходимого набора функций по тексту постановки.
3. Формальное окончание написания объявлений классов.

Решение:

Определим набор классов, набор полей классов, набросаем план функций и методов классов.

**Товар**: код, название, цена, скидка, количество (от этого поля и его вариаций зависит подход к реализации, можно упростить себе жизнь используя 2 количества – на складе и в корзине)

* Необходимые конструкторы
* Перегрузка оператора ==, !=
* Получить информацию о товаре
* Перегрузка вывода <<

**Каталог товаров (склад)**: массив товаров, количество уникальных товаров

* Конструктор (по умолчанию – считать из файла (в лучшем случае), организовать рандомные данные для экономии времени разработки)
* Деструктор

**Корзина**: массив товаров (массив адресов/указателей товаров), количество уникальных товаров, указатель на отсканированный товар

* Конструктор (по умолчанию)
* Деструктор
* «Сканировать» очередной товар
* Вывести описание отсканированного товара
* Добавить данные о товаре в чек
* Сформировать чек за покупку
* Рассчитать итоговую сумму к оплате
* Служебные методы: очистка корзины, перегрузка оператора += для добавления товара в корзину.

**Замечание.** По мере необходимости добавить служебные доп. функции. Возможно для корректной работы потребуется перегрузка оператора присваивания =.

**Этап 2.**

**Реализовать методы.**

Разработка этапа.

1. Уход в разработку реализации оговорённых методов. На данном этапе оговорённая структура подвергается изменениям по сильной необходимости (вообще говоря, старайтесь не отступать от намеченного плана).
2. Параллельное тестирование реализованного функционала.
3. В конце (не обязательно) – реализация небольшого консольного приложения, дополнение структуры методами, удобными для работы с пользователем. До наступления данного этапа код уже должен быть проверен, скорректирован в соответствиями с нормами, формально, будто готов к сдаче «заказчику».

Решение: остаётся на студентов, некоторые тонкости реализации перегрузки операторов тут.

**Промежуточный этап (дизайнерский) – оформление интерфейса.**

Этот этап будет рассмотрен позже.

**Этап 3.**

**Встраивание разработанной «библиотеки» в визуальное приложение (назначение кнопок).**

Этот этап будет рассмотрен позже.

**МИНИ-подзадача в классе**

**Постановка.** Реализовать часть функционала так, чтобы была возможность реализовать функцию добавления товара в корзину (метод add и перегрузка оператора +=).

**Решение.** Предложенное преподавателем решение:

#include <iostream>

#include <string>

class Product {

std::string code;

std::string name;

int price;

int discount;

int number;

public:

Product() {

code = 1;

name = "Product 1";

price = 320;

discount = 3;

number = 0;

};

bool operator==(const Product& prod) { return this->code == prod.code; };

friend std::ostream& operator<<(std::ostream& out, const Product& prod);

friend class Busket;

};

typedef Product\* ProductLink;

class Busket {

ProductLink\* goods;

int size;

int count;

ProductLink last;

public:

Busket() {

size = 50;

count = 0;

last = nullptr;

goods = new ProductLink[size];

};

~Busket() { delete[] goods; };

void setLast(ProductLink \_last) { last = \_last; }

void add() { \*this += last; };

friend std::ostream& operator<< (std::ostream& out, const Busket& busket);

private: // делаем часть методов скрытыми (служебными)

Busket& operator+=(ProductLink prod) {

for (int i = 0; i < this->count; i++) {

if (\*this->goods[i] == \*prod) { // используется оператор сравнения двух товаров

this->goods[i]->number += 1;

return \*this;

}

}

this->goods[count] = prod; // вообще говоря, используется оператор =

this->goods[count]->number += 1;

this->count++;

return \*this;

};

};

std::ostream& operator<< (std::ostream& out, const Product& prod) {

out << prod.name << " " << prod.price << " " << prod.discount

<< " " << (int)(prod.price \* (100 - prod.discount) / 100.0);

return out;

};

std::ostream& operator<< (std::ostream& out, const Busket& busket) {

for (int i = 0; i < busket.count; i++)

out << i + 1 << ". " << \*(busket.goods[i]) << std::endl;

return out;

};

int main() {

Product prod1;

Busket basket;

basket.setLast(&prod1); // пока нет метода сканирования

basket.add();

std::cout << basket;

system("pause");

return 0;

}