**Урок 3.8.10**

**Задание №5:**

Создать родительский класс «Товар», а также дочерние классы  
«Холодильник»и «Чайник», придумать им соответствующие поля,хранящие  
информацию о типе товара (для дочерних классов), стоимости единицы товара,  
количества товара на складе. Добавить метод info(), выводящий полную  
информацию о товаре. Корректно реализовать конструкторы для всех классов.  
Перегрузить оператор сложения так,чтобы к любому товару можно было прибавлять  
целое число,в результате чего должно увеличиваться поле,соответствующее  
количеству товара. Сделать классы шаблонными чтобы стоимость единицы товара  
могла быть любым подходящим типом данных. Протестируйте работу классов.  
Отправьте преподавателю код программы и скриншот консоли.

#include <iostream>

using namespace std;

// Родительский класс Product

template <typename T>

class Product

{

private:

int quantity;

string name;

T price;

public:

Product(const string &name, T price = 0, int quantity = 0) : name(name), price(price), quantity(quantity)

{

cout << "Конструктор Product()\n";

};

// вывод общей информации о любом товаре

virtual void info() const

{

cout << "Название: " << name << endl;

printf("Стоимость единицы товара: %.2f руб.\n", price);

cout << "Количество товара на складе: " << quantity << endl;

}

// Перегрузка оператора сложения

Product &operator+=(int quantity)

{

this->quantity += quantity;

return \*this;

}

};

// Дочерний класс наследуется от Product

template <typename T>

class Fridge : public Product<T>

{

private:

string type;

public:

Fridge(const string &name, T price, int quantity, const string &type) : Product<T>(name, price, quantity)

{

this->type = type;

cout << "Конструктор Fridge()\n";

}

// Переопределение метода вывода дополнительной информации

void info() const override

{

Product<T>::info();

cout << "Тип холодильника: " << type << "\n\n";

}

};

// Дочерний класс Kettle

template <typename T>

class Kettle : public Product<T>

{

private:

string type;

public:

Kettle(const string &name, T price, int quantity, const string &type) : Product<T>(name, price, quantity)

{

this->type = type;

cout << "Конструктор Kettle()\n";

}

// Переопределение метода вывода дополнительной информации

void info() const override

{

Product<T>::info();

cout << "Тип чайника: " << type << "\n\n";

}

};

int main()

{

cout << "=== Подготовительный этап ===\n";

Fridge fridge("Холодильник Beko", (float)39500.94, 5, "двухкамерный");

Kettle kettle("Электрочайник Braun", (double)2500.58, 10, "с функцией термоса");

cout << "=== Подготовительный этап завершён ===\n\n";

fridge.info();

kettle.info();

cout << "=== На склад приехало 10 холодильников и 45 чайников ===\n\n";

fridge += 10;

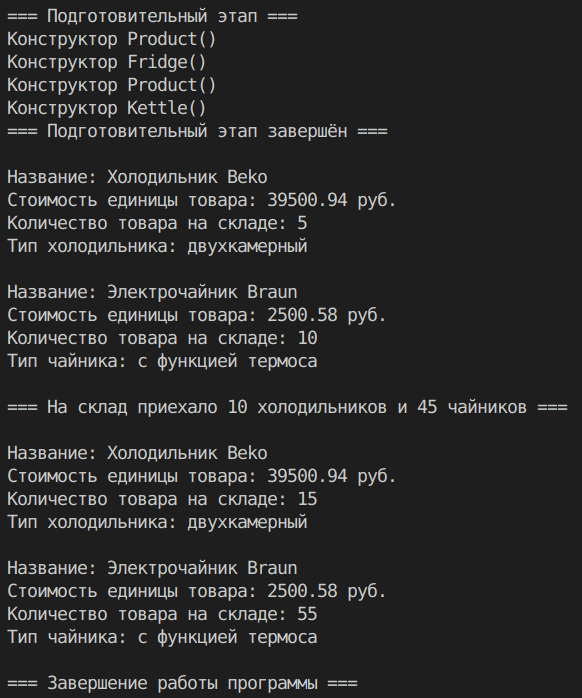
kettle += 45;

fridge.info();

kettle.info();

cout << "=== Завершение работы программы ===\n\n";

return 0;

}