ДОДАТОК А

Код програми

```
#include <iostream>
#include <string>
#include <vector>
#include <ctime>
#include <cstdlib>
#include <clocale>

// Структура для опису деталі
struct Detail {
    std::string name;
    int quantity;
    double price;
};

// Структура для опису замовлення
struct Order {
    int orderCode;
    std::string customerName;
    std::string orderDate;
    std::vector<Detail> details;
    double totalPrice;
    bool isPaid;
    std::string status; // "Прийнято", "В роботі", "Готово", "Відправлено", "Скасовано"
};

// Функція для генерації коду замовлення
int generateOrderCode() {
    std::srand(std::time(0)); // Ініціалізуємо генератор випадкових чисел
    return std::rand() % 9000 + 1000; // Генеруємо випадковий код (наприклад, 1000-9999)
}

// Функція для підрахунку загальної вартості замовлення
double calculateTotalPrice(const std::vector<Detail>& details) {
    double total = 0.0;
    for (const auto& detail : details) {
```

```
        total += detail.quantity * detail.price;
    }
    return total;
}

int main() {
    setlocale(LC_ALL, "Ukrainian");

    Order currentOrder;

    // 1. Опис замовлення
    std::cout << "Введіть ім'я замовника: ";
    std::getline(std::cin, currentOrder.customerName);

    currentOrder.orderDate = __DATE__; // Поточна дата

    // Додавання деталей до замовлення
    char addMoreDetails = 'y';
    do {
        Detail newDetail;
        std::cout << "Введіть назву деталі: ";
        std::getline(std::cin, newDetail.name);
        std::cout << "Введіть кількість: ";
        std::cin >> newDetail.quantity;
        std::cout << "Введіть ціну за одиницю: ";
        std::cin >> newDetail.price;
        std::cin.ignore(); // Очищаємо буфер введення

        currentOrder.details.push_back(newDetail);

        std::cout << "Додати ще деталь? (y/n): ";
        std::cin >> addMoreDetails;
        std::cin.ignore(); // Очищаємо буфер введення
    } while (addMoreDetails == 'y');
```

```
    // 2. Підрахунок загальної вартості
    currentOrder.totalPrice = calculateTotalPrice(currentOrder.details);
    std::cout << "Загальна вартість замовлення: " << currentOrder.totalPrice << std::endl;

    // 3. Оплата замовлення
    char confirmPayment;
    std::cout << "Замовлення оплачено? (y/n): ";
    std::cin >> confirmPayment;
    currentOrder.isPaid = (confirmPayment == 'y');

    // 4. Створення коду (фрейму) замовлення
    currentOrder.orderCode = generateOrderCode();
    std::cout << "Код замовлення: " << currentOrder.orderCode << std::endl;

    // 5. Відстеження статусу замовлення
    currentOrder.status = "Прийнято"; // Початковий статус
    std::cout << "Статус замовлення: " << currentOrder.status << std::endl;

    // Тут можна додати логіку для зміни статусу замовлення (наприклад, через функції)
    // Наприклад, зміна статусу на "В роботі", "Готово", "Відправлено"

    return 0;
}
```

Код модуля INTELEKT

```
#include <iostream>
#include <string>
#include <vector>
#include <map>
#include <clocale>


enum class OrderStatus {
    New,
    InProgress,
    Completed,
    Shipped,
    Cancelled
};

enum class Priority {
    Low,
    Medium,
    High
};

std::string statusToString(OrderStatus status) {
    switch (status) {
    case OrderStatus::New: return "Нове";
    case OrderStatus::InProgress: return "В роботі";
    case OrderStatus::Completed: return "Готово";
    case OrderStatus::Shipped: return "Відправлено";
    case OrderStatus::Cancelled: return "Скасовано";
    default: return "Невідомий статус";
    }
}

std::string priorityToString(Priority p) {
    switch (p) {
```

```
    case Priority::Low: return "Низький";
    case Priority::Medium: return "Середній";
    case Priority::High: return "Високий";
    default: return "Невідомий";
    }
}

class Order {
private:
    int id;
    OrderStatus status;
    Priority priority;
    int daysSinceCreation;

    bool isValidTransition(OrderStatus current, OrderStatus next) const {
        if (current == OrderStatus::Cancelled || current == OrderStatus::Shipped)
            return false;

        switch (current) {
        case OrderStatus::New:
            return next == OrderStatus::InProgress || next == OrderStatus::Cancelled;
        case OrderStatus::InProgress:
            return next == OrderStatus::Completed || next == OrderStatus::Cancelled;
        case OrderStatus::Completed:
            return next == OrderStatus::Shipped;
        default:
            return false;
        }
    }

public:
    Order(int id, Priority priority, int daysSinceCreation)
        : id(id), status(OrderStatus::New),
        priority(priority), daysSinceCreation(daysSinceCreation) {
    }
```

```
    bool setStatus(OrderStatus newStatus) {
        if (isValidTransition(status, newStatus)) {
            status = newStatus;
            std::cout << "Статус замовлення #" << id << " змінено на: "
                << statusToString(newStatus) << std::endl;
            return true;
        }
        else {
            std::cout << "Помилка: не можна змінити статус замовлення #" << id
                << " з \"" << statusToString(status)
                << "\" на \"" << statusToString(newStatus) << "\"" << std::endl;
            return false;
        }
    }

    void printStatus() const {
        std::cout << "Замовлення #" << id
            << " | Статус: " << statusToString(status)
            << " | Пріоритет: " << priorityToString(priority)
            << " | Створено: " << daysSinceCreation << " днів тому" << std::endl;
    }

    OrderStatus getStatus() const { return status; }
    Priority getPriority() const { return priority; }
    int getDaysSinceCreation() const { return daysSinceCreation; }
};

// ======= Аналітичні функції =======

void printOrderDistribution(const std::vector<Order>& orders) {
    std::map<OrderStatus, int> distribution;
    for (const auto& order : orders) {
        distribution[order.getStatus()]++;
    }
```

```
    std::cout << "\n--- Розподіл замовлень по статусах ---\n";
    for (const auto& entry : distribution) {
        std::cout << statusToString(entry.first) << ": " << entry.second << std::endl;
    }
}

void forecastCapacity(const std::vector<Order>& orders) {
    int inProgressCount = 0;
    int highPriorityCount = 0;
    int overdueCount = 0;

    for (const auto& order : orders) {
        if (order.getStatus() == OrderStatus::InProgress) {
            inProgressCount++;
            if (order.getPriority() == Priority::High) {
                highPriorityCount++;
            }
            if (order.getDaysSinceCreation() > 10) { // Умова прострочення
                overdueCount++;
            }
        }
    }

    std::cout << "\n--- Прогноз завантаженості ---\n";

    if (inProgressCount == 0) {
        std::cout << "☑ Повна доступність — замовлення відсутні в роботі." << std::endl;
        return;
    }

    if (overdueCount > 0) {
        std::cout << "⚠ Є " << overdueCount << " прострочених замовлень. Термінове
втручання!" << std::endl;
    }
```

```
        if (highPriorityCount > 3) {
            std::cout << "⚠ Багато важливих замовлень (високий пріоритет) — обмежена доступність." << std::endl;
        }

        if (inProgressCount > 5) {
            std::cout << "⚠ Загальне навантаження високе (" << inProgressCount << " замовлень в роботі)." << std::endl;
        }
        else {
            std::cout << "☑ Поточне навантаження допустиме. Можна брати нові замовлення." << std::endl;
        }
    }

    // ======= Основна програма =======
    int main() {
        setlocale(LC_ALL,"Ukrainian");

        std::vector<Order> orders;

        orders.emplace_back(1001, Priority::High, 12);
        orders.emplace_back(1002, Priority::Medium, 3);
        orders.emplace_back(1003, Priority::Low, 1);
        orders.emplace_back(1004, Priority::High, 15);
        orders.emplace_back(1005, Priority::Medium, 6);
        orders.emplace_back(1006, Priority::High, 2);
        orders.emplace_back(1007, Priority::Low, 5);
        orders.emplace_back(1008, Priority::Medium, 7);
        orders.emplace_back(1009, Priority::Low, 11);
        orders.emplace_back(1010, Priority::High, 9);

        // Імітуємо зміну статусів
        for (int i = 0; i < 7; ++i) {
```

```
        orders[i].setStatus(OrderStatus::InProgress);
    }

    orders[8].setStatus(OrderStatus::Cancelled);
    orders[9].setStatus(OrderStatus::Completed);

    // Виведення
    for (const auto& order : orders) {
        order.printStatus();
    }

    printOrderDistribution(orders);
    forecastCapacity(orders);

    return 0;
}
```

ДОДАТОК Б

Апробація роботи

ІНТЕЛЕКТУАЛЬНА СИСТЕМА ОБРОБКИ ЗАМОВЛЕНЬ НА ВИГОТОВЛЕННЯ ПРОДУКЦІЇ ПІДПРИСМСТВА

Швачка Владислав Володимирович
ORCID ID: 0009-0008-3396-8129
здобувач вищої освіти факультету
автоматизації та комп'ютерно—інтегрованих технологій
Харківський національний університет радіоелектроніки, Украі'на

Науковий керівник: Сезонова Ірина Костянтинівна
ORCID ID: 0000-0002-9396-7434
канд. техн. наук, доцент, професор кафедри автоматизації,
комп'ютерно—інтегрованих технологій та робототехніки
Харківський національний університет радіоелектроніки, Украі'на

Автоматизація — це основа сучасної обробки замовлень. Автоматизовані системи можуть виконувати повторювані завдання, такі як розміщення замовлень, відстеження статусу доставки та оновлення інформації про інвентаризацію продукції підприсмства на складі, що значно зменшує витрати на робочу силу та помилки.

Програмне забезпечення для управління замовленнями (OMS): централізована платформа, яка керує всіма етапами обробки замовлень від прийому замовлення до доставки. OMS, повна назва Order Management System — це програмне забезпечення, яке здійснює централізоване та покрокове управління обробкою замовлень: від їх надходження до отримання кліснтом.

Для виробничого підприсмства система OMS не може існувати окремо від MES (Manufacturing Execution System, система управління виробництвом), вона с її підсистемою і пов'язана з іншими підсистемами, зокрема з WMS. Програмне забезпечення для управління складом (WMS, повна назва Warehouse Management System): оптимізус операції на складі, забезпечуючи точне відстеження інвентаризації та ефективне виконання замовлень.

Аналогом системи обробки замовлень для виробничого підприемства можна вважати CRM—системи (customer relationship management), які автоматизують взаемовідносинами з клієнтами. Існують як вузькопрофільні, так і універсальні рішення. Кожне з них мас інтеграцію з e—mail сервісами, мобільний застосунок. Деякі мають

можливість прогнозування продажів, аналітику та допомагають в організації завдань. Українські CRM—системи закривають потреби, в основному, малого та великого бізнесу. Для виробництва можна використовувати такі системи як Perfectum та BJet ERP, які с частиною ERP— системи підприемства, її вбудованим модулем.

Основною концепцісю програми обробки замовлень е підсистема приймання та супроводження заказів через сайт виробничого підприемства. Програма мае дружній інтерфейс, через який користувач може замовити товар підприсмства, а також за її допомогою відстежити статус (етап виконання) свого замовлення. Користувач створюс замовлення на сайті, його оплачуе. Далі користувачу повідомляють номер замовлення ID, за допомогою якого він може відстежити на сайті статус свого замовлення.

Алгоритм її роботи:

– на підставі повідомлення на сайті система створює картку клієнта в базі даних;

– далі за допомогою автоматизації відбувається сортування повідомлень і коментарів за номенклатурою продукції, кількістю, строками виконання замовлення тощо;

– далі працює алгоритм виконання замовлення;

– система дае змогу автоматично надсилати необхідні дані на замовлення клієнту;

– система можна змінювати, програмувати і налаштовувати свої параметри.

Інтелектуальний модуль системи обробки замовлень отримав назву INTELEKT.

Реалізації управління статусами замовлення, виявлення критичних моментів (наприклад, наявність простроченого замовлення, відхиленого замовлення тощо) та прогнозування завантаженості реалізовано в модулі INTELEKT.

Основними об'єктами програми INTELEKT е переліки (enum) для статусів, клас Order і функції для зміни та виводу поточного статусу.

До коду INTELEKT включені наступні об'єкти:

– перелік OrderStatus (enum class), який містить всі можливі статуси;

– допоміжна функція statusToString для виведення статусу у вигляді рядка;

– клас Order, який представляю замовлення з ID та статусом;

– функція setstatus дозволяе змінювати статус замовлення;

– функція printstatus виводить поточний статус на екран.

Також в модулі INTELEKT здійснюються перевірки та обмеження на зміну статусів, щоб не можна було, наприклад, перейти з "Відправлено" назад на "В роботі" — тобто реалізовано логіку послідовності статусів.

Висновки. Інтелектуальний модуль системи обробки замовлень дас змогу здійснювати повний цикл опрацювання замовлення до його логічного завершения, а також надавати швидші відповіді на коментарі та запитання кліснтів. Модуль також здійснюс аналіз поточного стану системи замовлень виробничого підприемства, прогнозування замовлень та аналіз відмов. Наявність такого модуля в автоматизованій системі підприемства підвищуе досконалість виробництва та його заощадливість, що е ознакою заощадливого виробництва Lean production.

Список використаних джерел:

1. Вибирай свое: 8 українських CRM-систем для різних видів бізнесу (02.06.2025) https://sendpulse.ua/blog/ukrainian-crm-systems
2. Взаемодія ощадливого виробництва і автоматизація (02.06.2025). https://ru.rememo.io/Ыog/lean-manufacturing-and-automation.

ДОДАТОК В

Демонстраційний матеріал

Формат	Позиція	Найменування	Додаткові відомості
		Текстові документи	
А4	ГЮИК. 442351.012 ПЗ	Пояснювальна записка	63 с.
		Додаткові матеріали	
		Код програми	5 с.
		Апробація роботи	4 с.
		Демонстраційний матеріал	10 с.

Змін.	Арк.	Номер докум.	Підпис	Дата	ГЮИК.442351.012 ВД			
Розроб.		Швачка В. В.			Розроблення інтелектуальної системи обробки замовлень для виробничого підприємства	Літера	Аркуш	Аркушів
Перевір.		Сезонова І. К.				Н	1	1
Н. контр.		Стародубцев М.П.				Кафедра КІТАР ХНУРЕ		
Затв.		Невлюдов І.Ш.						