# BÁO CÁO BÀI TẬP

**Small Clinic Management System**
Môn: Lập Trình Hướng Đối Tượng – C++

## 1. Phân tích OOA (Object-Oriented Analysis)

**Bước 1: Xác định các đối tượng (Objects / Nouns):**

- Patient (Bệnh nhân)
- ChronicPatient (Bệnh nhân mãn tính)
- Doctor (Bác sĩ)
- Appointment (Lịch hẹn)
- Clinic (Quản lý chung)

**Bước 2: Xác định thuộc tính (Attributes):**

- Patient: id, name, age, history
- ChronicPatient: kế thừa Patient + condition, lastCheckup
- Doctor: id, name, specialty, appointments
- Appointment: date, time, reason, status, patientId, doctorId
- Clinic: danh sách bệnh nhân, bác sĩ, lịch hẹn

**Bước 3: Xác định phương thức (Methods / Verbs):**

- Patient: addHistory(), display(), scheduleAppointment()
- ChronicPatient: display(), scheduleAppointment() (override)
- Doctor: assignAppointment(), updateStatus(), display()
- Appointment: setStatus(), display()
- Clinic: addPatient(), addDoctor(), schedule(), showPatients(), showDoctors(), showAppointments()

**Bước 4: Quan hệ kế thừa (Inheritance):**

- `ChronicPatient` kế thừa từ `Patient`
- Các lớp khác quan hệ "has-a" (Clinic có Patient/Doctor/Appointment)

## 2. Thiết kế lớp & giải thích kế thừa

Em chọn `Patient` làm lớp cơ sở vì tất cả bệnh nhân đều có ID, tên, tuổi, và lịch sử khám.
Sau đó em tạo lớp `ChronicPatient` kế thừa từ `Patient` để thêm thông tin riêng như bệnh mãn tính, ngày khám gần nhất.

Trong thiết kế này:

- **Kế thừa** giúp tái sử dụng code, không cần viết lại toàn bộ lớp Patient.
- **Đa hình** (polymorphism) thể hiện qua phương thức `scheduleAppointment()`:
  - Patient thường thì "schedule as needed"
  - ChronicPatient thì "must schedule every 3 months"

# 3. Walkthrough code

- Em tạo 5 lớp: `Patient`, `ChronicPatient`, `Doctor`, `Appointment`, `Clinic`.
- Dùng `vector` để lưu danh sách.
- Trong `main()`, em viết menu để nhập dữ liệu và thực hiện chức năng.

Ví dụ:

- Người dùng chọn "1" → thêm Patient thường.
- Người dùng chọn "2" → thêm ChronicPatient.
- Người dùng chọn "3" → thêm Doctor.
- Người dùng chọn "6" → đặt lịch hẹn giữa Patient và Doctor.

# 4. Kiểm thử (Testing)

Em đã chạy thử với test case nhỏ:

**Input thao tác:**

1. Thêm bệnh nhân thường: `P01 Nam 30`
2. Thêm bệnh nhân mãn tính: `P02 Hoa 50 Diabetes 01-09-2025`
3. Thêm bác sĩ: `D01 Binh General`
4. Đặt lịch hẹn: `P02 D01 10-10-2025 09:00 Checkup`

**Output trên console:**

```
Hoa (Chronic) must schedule appointment every 3 months.
Appointment scheduled successfully.
Appointment: 10-10-2025 09:00, Reason: Checkup, Patient: P02, Doctor: D01,
Status: Scheduled
```

Điều này chứng tỏ hệ thống chạy đúng và thể hiện được kế thừa, đa hình.

# 5. Cách em sử dụng LLM (ChatGPT)

Trong quá trình làm bài, em có dùng ChatGPT để:

- Tìm tòi các class cần có trong hệ thống (Patient, Doctor, Appointment, Clinic).
- Hỏi về cách override hàm trong lớp kế thừa.
- Tham khảo cấu trúc menu console từ code mẫu đã có.

Em chỉ dùng ChatGPT để tham khảo ý tưởng và sửa lỗi nhỏ, toàn bộ code em tự viết và chỉnh lại cho phù hợp với yêu cầu bài tập.

# 6. Kết luận

Bài tập giúp em hiểu rõ hơn về:

- Cách áp dụng OOP vào bài toán thực tế (quản lý phòng khám).
- Kế thừa và đa hình trong C++ qua Patient và ChronicPatient.
- Thiết kế hệ thống gồm nhiều lớp có quan hệ chặt chẽ.