Đây là một chương trình C++ giải quyết bài toán Knapsack 0/1 bằng phương pháp lập trình động. Chương trình yêu cầu người dùng nhập số lượng món hàng (n) và trọng lượng tối đa của balo (w). Sau đó, chương trình yêu cầu người dùng nhập trọng lượng và giá trị của từng món hàng. Chương trình tính toán giá trị tối ưu mà balo có thể mang được và in ra giá trị đó, cùng với danh sách các món hàng được chọn.

1. Yêu cầu người dùng nhập số lượng món hàng (n) và trọng lượng tối đa của balo (w).

2. Tạo mảng a và b để lưu trữ trọng lượng và giá trị của từng món hàng, tương ứng. Kích thước của các mảng này là 100.

3. Tạo một mảng 2 chiều f có kích thước [100][101] để lưu trữ giá trị tối đa có thể đạt được cho các bài toán con khác nhau. Các hàng đại diện cho các món hàng và cột đại diện cho khả năng chứa trọng lượng của balo.

4. Tạo một mảng p có kích thước 100 để lưu trữ thứ tự các món hàng được chọn cho giải pháp tối ưu. Khởi tạo biến o bằng 0 để theo dõi số lượng các món hàng được chọn.

5. Đọc trọng lượng và giá trị của từng món hàng từ người dùng và lưu trữ chúng vào các mảng a và b, tương ứng, sử dụng một vòng lặp chạy từ 1 đến n.

6. Thực hiện lập trình động để tìm giá trị tối đa có thể đạt được cho các khả năng trọng lượng khác nhau của balo.

a. Duyệt qua từng món hàng từ 1 đến n bằng một vòng lặp.

b. Đối với mỗi món hàng, duyệt qua từng khả năng trọng lượng từ 1 đến w bằng một vòng lặp lồng nhau.

c. Nếu khả năng trọng lượng hiện tại (v) lớn hơn hoặc bằng trọng lượng của món hàng hiện tại (a[i]), tính toán giá trị tối đa có thể đạt được bằng cách bao gồm hoặc loại bỏ món hàng hiện tại trong balo. Cập nhật giá trị trong mảng f tương ứng:

- f[i][v] = max(f[i - 1][v - a[i]] + b[i], f[i - 1][v])

d. Nếu khả năng trọng lượng hiện tại (v) nhỏ hơn trọng lượng của món hàng hiện tại (a[i]), không thể bao gồm món hàng trong balo. Đặt f[i][v] bằng giá trị thu được bằng cách loại bỏ món hàng hiện tại:

- f[i][v] = f[i - 1][v]

7. In ra giá trị tối ưu mà balo có thể mang được: cout << "\nGiá trị tối ưu mà balo mang được: \n" << f[n][w] << '\n';

8. In ra danh sách các món hàng được chọn cho giải pháp tối ưu bằng cách truy vết ngược:

a. Bắt đầu từ món hàng cuối cùng (n) và trọng lượng còn lại (w).

b. Duyệt ngược từ n đến 0 bằng một vòng lặp.

c. Nếu giá trị tại f[i][w] không bằng giá trị tại f[i - 1][w], có nghĩa là món hàng i được chọn cho giải pháp tối ưu.

d. Lưu trữ chỉ số i trong mảng p và cập nhật trọng lượng còn lại bằng cách trừ trọng lượng của món hàng i.

e. Lặp lại các bước c và d cho đến khi ghi lại tất cả các món hàng được chọn.

9. In ra danh sách các món hàng được chọn cho giải pháp tối ưu bằng cách duyệt qua mảng p theo thứ tự ngược và in ra các giá trị:

+ for (int i = o - 1; i >= 0; --i)

+ cout << p[i] << " ";

10. Kết thúc chương trình bằng cách trả về 0 từ hàm main().

Thuật toán này sử dụng lập trình động để giải quyết bài toán Knapsack 0/1 một cách hiệu quả bằng cách chia nhỏ thành các bài toán con trùng lắp và lưu trữ các giải pháp trong một bảng (f). Bằng cách sử dụng bảng để tính toán giá trị tối đa theo cách lặp lại, thuật toán tránh tính toán trùng lặp và đạt được một giải pháp tối ưu.