**Trần Hoàng Triển - MSSV:46.01.104.196**

**Cài đặt bài toán xếp lịch**

**Dữ liệu vào:**

-Số lượng cuộc họp **n**

-Thời gian bắt đầu và kết thúc của từng cuộc họp lần lượt

**A screenshot of a computer

Description automatically generated with low confidence**

**Dữ liêu ra :**

-Các cuộc họp được tổ chức

A white background with black text

Description automatically generated with low confidence

**Miêu tả cách xử lý dữ liệu như sau:**

Sau khi nhập dữ liệu:

-Số lượng cuộc họp là 4

-Thời gian bắt đầu và kết thúc của từng cuộc họp như sau:

0 4

2 4

4 6

5 7

Chương trình sẽ xử lý như sau:

1. Vector `meetings` sẽ chứa 4 phần tử, mỗi phần tử là một cấu trúc `Meeting` chứa thông tin về thời gian bắt đầu và kết thúc của từng cuộc họp.

- `meetings[0]` có thời gian bắt đầu là 0 và kết thúc là 4.

- `meetings[1]` có thời gian bắt đầu là 2 và kết thúc là 4.

- `meetings[2]` có thời gian bắt đầu là 4 và kết thúc là 6.

- `meetings[3]` có thời gian bắt đầu là 5 và kết thúc là 7.

2. Vector `meetings` sẽ được sắp xếp theo thời gian kết thúc tăng dần:

- `meetings[0]`: 0 4

- `meetings[1]`: 2 4

- `meetings[2]`: 4 6

- `meetings[3]`: 5 7

3. Vector `schedule` được khởi tạo và có giá trị ban đầu là [0] (chọn cuộc họp đầu tiên).

4. Biến `lastEndTime` được khởi tạo với giá trị là 4 (thời gian kết thúc của cuộc họp đầu tiên).

5. Bắt đầu vòng lặp từ cuộc họp thứ hai (`i = 1`) đến cuộc họp cuối cùng (`i = 3`):

- Tại vòng lặp `i = 1`: Thời gian bắt đầu của cuộc họp thứ hai là 2. Vì thời gian này lớn hơn hoặc bằng `lastEndTime` (4), cuộc họp thứ hai không xung đột với cuộc họp đầu tiên. Cuộc họp thứ hai được thêm vào vector `schedule`, và `lastEndTime` được cập nhật thành 4 (thời gian kết thúc của cuộc họp thứ hai).

- Tại vòng lặp `i = 2`: Thời gian bắt đầu của cuộc họp thứ ba là 4. Vì thời gian này bằng `lastEndTime` (4), cuộc họp thứ ba không xung đột với cuộc họp thứ hai. Cuộc họp thứ ba được thêm vào vector `schedule`, và `lastEndTime` được cập nhật thành 6 (thời gian kết thúc của cuộc họp thứ ba).

- Tại vòng lặp `i = 3`: Thời gian bắt đầu của cuộc họp cuối cùng là 5. Vì thời gian này lớn hơn `lastEndTime` (6), cuộc họp cuối cùng xung đột với cuộc họp thứ ba. Cuộc họp cuối cùng không được thêm vào vector `schedule`, và `lastEndTime` được cập nhật thành 7 (thời gian kết thúc của cuộc họp cuối cùng).

6. Vector `schedule` chứa các chỉ số của các cuộc họp được tổ chức mà không có xung đột: - `schedule = [0, 2, ]`

7. Chương trình xuất ra màn hình danh sách các cuộc họp trong `schedule`:

- Các cuộc họp được tổ chức: 1, 3.

**Để thực thi chương trình trên, bạn có thể làm theo các bước sau:**

1. Mở trình biên dịch C++ (như Dev-C++, Visual Studio,).

2. Tạo một tệp tin mới và sao chép đoạn mã vào tệp tin đó.

3. Lưu tệp tin với phần mở rộng `.cpp` (ví dụ: `**XepLichHop.cpp**`).

4. Tiến hành chạy chương trình và nhập dữ liệu vào là số lượng cuộc họp **n** và thời gian bắt đầu và kết thúc của từng cuộc họp.

5. Trình biên dịch sẽ thực thi chương trình và hiển thị kết quả.

**Source code:**

#include <iostream>

#include <vector>

#include <algorithm>

struct Meeting {

int startTime;

int endTime;

};

bool sortByEndTime(const Meeting& m1, const Meeting& m2) {

return m1.endTime < m2.endTime;

}

int main() {

int n;

//cout << "Nhap so luong cuoc hop: ";

std::cin >> n;

std::vector<Meeting> meetings(n);

for (int i = 0; i < n; i++) {

//cout << "Nhap thoi gian bat dau va ket thuc cua cuoc hop " ;

std::cin >> meetings[i].startTime >> meetings[i].endTime;

}

std::sort(meetings.begin(), meetings.end(), sortByEndTime);

std::vector<int> schedule;

schedule.push\_back(0); // Chọn cuộc họp đầu tiên

int lastEndTime = meetings[0].endTime;

for (int i = 1; i < n; i++) {

if (meetings[i].startTime >= lastEndTime) {

schedule.push\_back(i); // Chọn cuộc họp không xung đột

lastEndTime = meetings[i].endTime;

}

}

std::cout << "Cac cuoc hop duoc to chuc: ";

for (int i : schedule) {

std::cout << i + 1 << " ";

}

std::cout << std::endl;

return 0;

}