Lập Trình C (cơ bản)

Chương 05. Danh sách, ngăn xếp và hàng đợi FIFO

Soạn bởi: TS. Nguyễn Bá Ngọc

2021

Nội dung

- Bài tập sử dụng danh sách móc nối kép
- Ngăn xếp và hàng đợi FIFO

Nội dung

- Bài tập sử dụng danh sách móc nối kép
- Ngăn xếp và hàng đợi FIFO

Bài tập 5.1. Danh sách bài hát

Cho một danh sách các bài hát được lưu trong tệp songs.txt. Thông tin về mỗi bài hát trong danh sách được lưu trên một dòng trong tệp songs.txt bao gồm Tên bài hát và đường dẫn của tệp bài hát trong máy tính người dùng. Tên bài hát và đường dẫn được ngăn cách bằng chuỗi "*#*" (giả sử không xuất hiện trong tên và đường dẫn).
Ví dụ một dòng trong songs.txt:

```
Chiếc lá thu phai*#*/home/nva/musics/chiec-la-thu-phai.mp3
```

Yêu cầu: Viết chương trình đọc và hiển thị danh sách tên bài hát từ tệp songs.txt, khi người dùng chọn 1 bài hát thì hiển thị đường dẫn đến tệp tương ứng. Chương trình cấp phát bộ nhớ động để lưu tên và đường dẫn bài hát, hạn chế dư thừa.

Bài tập 5.1: Cấu trúc thông tin bài hát

- Lưu bài hát theo cấu trúc sau:

```
typedef struct song_s {
    char *name;
    char *path;
} *song_t;
```

  - Trong đó name là con trỏ tới tên bài hát,
  - path là con trỏ tới đường dẫn bài hát.

**Gợi ý: Sử dụng một mảng đệm để đọc từng dòng trong songs.txt. Với mỗi dòng đọc được, thực hiện tách các thành phần name và path, tính độ dài và cấp phát bộ nhớ cho cấu trúc song_s để lưu bài hát tương ứng.*

Bài tập 5.1: Giao diện danh sách bài hát

Chương trình đọc dữ liệu từ songs.txt và hiển thị danh sách bài hát kèm với STT, sau đó hỏi người dùng chọn 1 bài hát (STT) để phát. Ví dụ danh sách bài hát:

```
1. Chiếc lá thu phai
2. Mưa hồng
3. Nắng thủy tinh

Chọn một bài để phát, stt = 1

/home/nva/musics/chiec-la-thu-phai.mp3
```

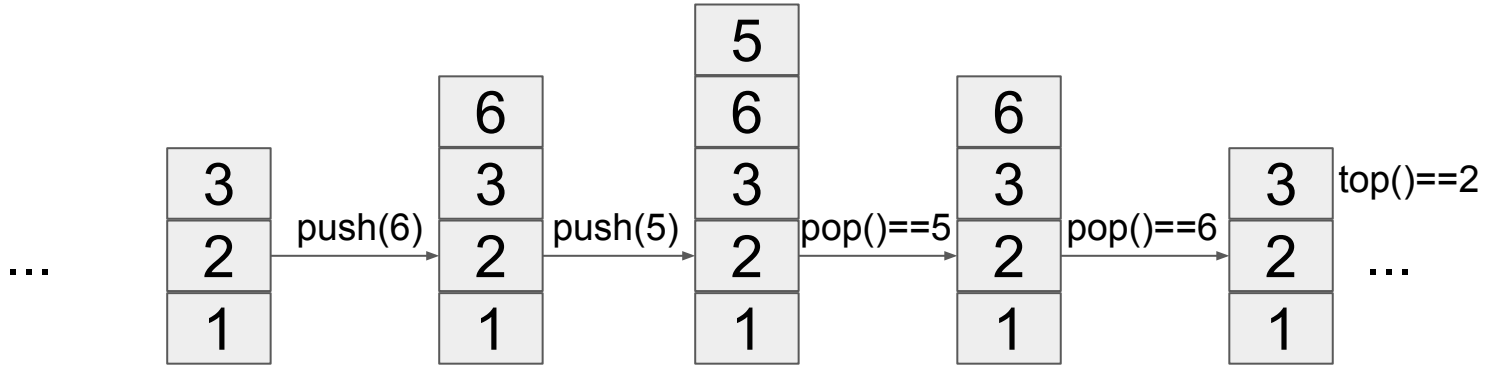
Chương trình lặp bước chọn bài để người dùng chọn nhiều lần.

Nội dung

- Bài tập sử dụng danh sách móc nối kép
- Ngăn xếp và hàng đợi FIFO

Ngăn xếp

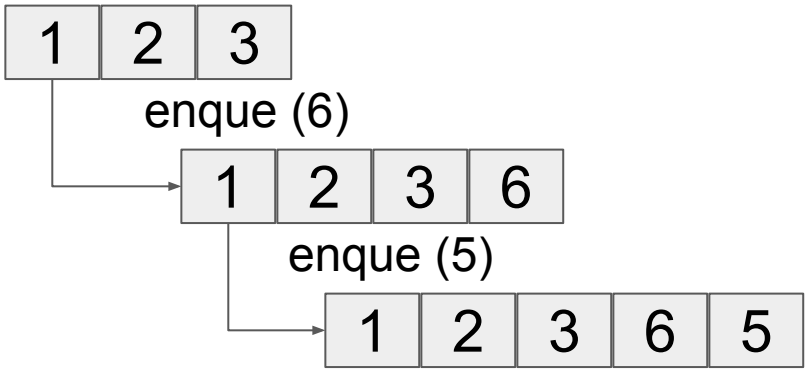
- Cấu trúc lưu trữ truy cập tuần tự, các phần tử được lấy ra theo thứ tự ngược với thứ tự đưa vào:
  - Thao tác đưa phần tử vào thường được gọi là push
  - Thao tác lấy phần tử ra thường được gọi là pop
  - Thao tác đọc phần tử sẽ được lấy ra ở lượt pop tiếp theo thường được gọi là top
- Ví dụ:

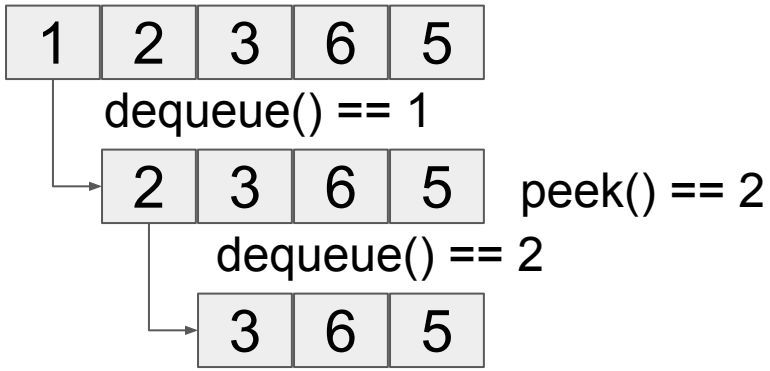


Hàng đợi FIFO

- Cấu trúc lưu trữ truy cập tuần tự, các phần tử được lấy ra theo thứ tự ngược với thứ tự được đưa vào:
  - Thao tác đưa phần tử vào thường được gọi là enqueue
  - Thao tác lấy phần tử ra thường được gọi là dequeue
  - Thao tác đọc phần tử sẽ được lấy ra trong lần dequeue tiếp theo thường được gọi là peek
- Ví dụ:

...





Triển khai ngăn xếp và hàng đợi FIFO

- Ngăn xếp và hàng đợi FIFO có thể được triển khai dựa trên mảng cấp phát động, hoặc đơn giản nhất là dựa trên danh sách liên kết.
  - Về bản chất là đóng gói các giao diện danh sách liên kết.
  - => Về mặt lý thuyết hoàn toàn có thể sử dụng giao diện danh sách liên kết với lô-gic tương tự, tuy nhiên sự khác biệt nằm ở cách dùng từ trong diễn đạt.
- Có thể sử dụng danh sách liên kết đơn với các con trỏ tới phần tử đầu tiên và cuối cùng trong danh sách, cho phép thêm phần tử mới vào đầu hoặc cuối danh sách.

Tham khảo triển khai dựa trên SLL (Singly Linked List) ở địa chỉ: https://github.com/bangoc/cgen

Triển khai ngăn xếp và hàng đợi FIFO$_{(2)}$

Giao diện thao tác với dữ liệu kiểu long:

- Ngăn xếp:
  - `void sll_stack_push_l(sll_t list, long value);`
  - `long sll_stack_pop_l(sll_t list);`
  - `long sll_stack_top_l(sll_t list);`
  - push - đưa vào ngăn xếp (đầu danh sách), pop - đọc và xóa phần tử (đầu danh sách), top - chỉ đọc, không xóa phần tử.
- Hàng đợi FIFO:
  - `void sll_fifo_enqueue_l(sll_t list, long value);`
  - `long sll_fifo_dequeue_l(sll_t list);`
  - `long sll_fifo_peek_l(sll_t list);`
  - enqueue - đưa vào hàng đợi (cuối danh sách), dequeue - đọc và xóa phần tử (đầu danh sách), peek - chỉ đọc, không xóa phần tử.

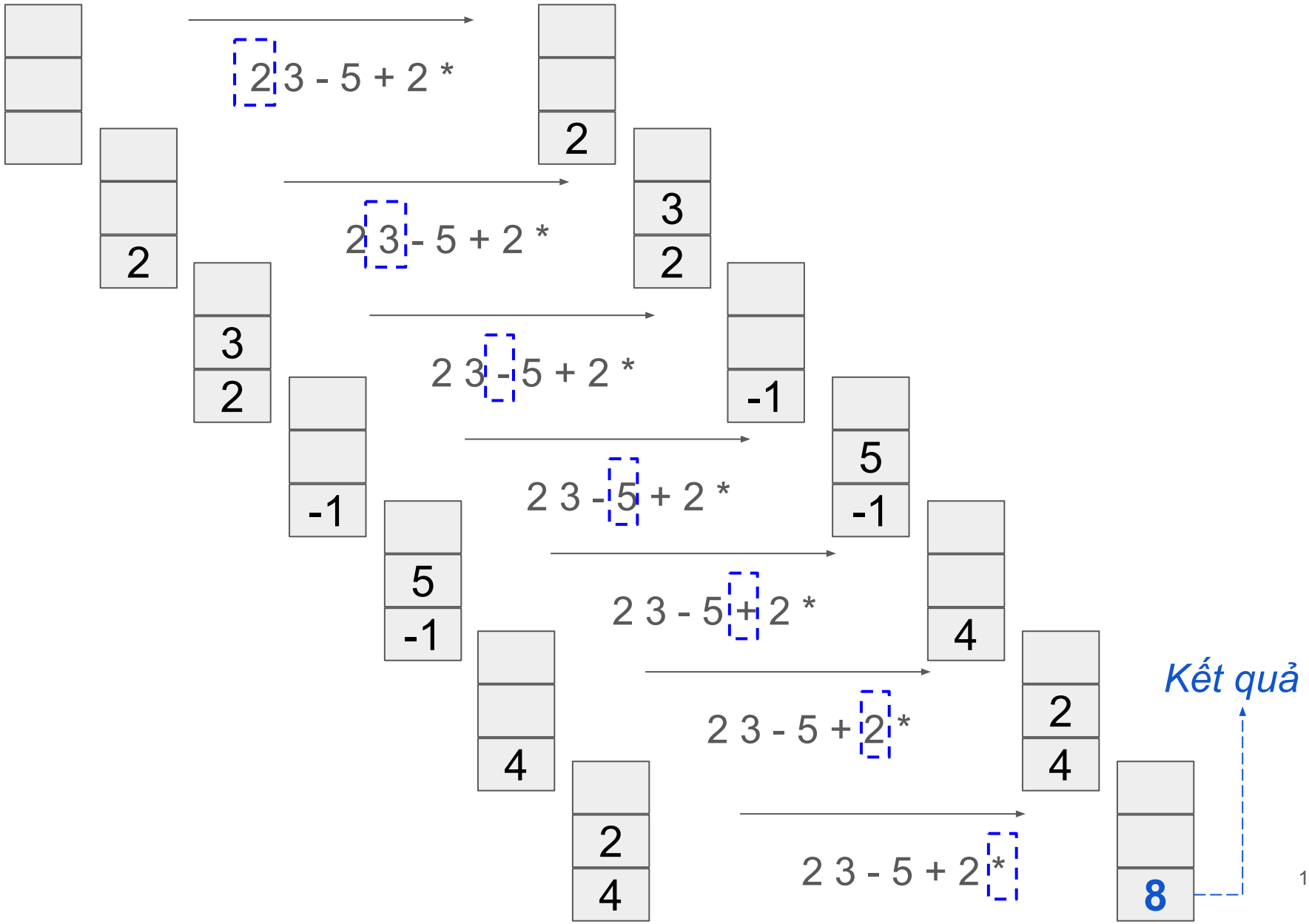
Bài tập 5.2. Tính biểu thức ở dạng hậu tố

- Viết chương trình đọc một biểu thức hậu tố từ bàn phím cho tới khi người dùng bấm phím Enter.
- Trong phạm vi bài tập này chúng ta chỉ xét các phép toán cơ bản: +, -, *, và /.
- Sau đó chương trình tính và in kết quả ra màn hình nếu biểu thức hợp lệ, hoặc in ra thông báo biểu thức không hợp lệ (kèm theo gợi ý vị trí lỗi) trong trường hợp ngược lại.
- Trong phiên bản đầu tiên chúng ta giả sử các toán tử đều là số thực không âm, các thành phần được ngăn cách bởi khoảng trắng.
- Ví dụ:
  - 1 2 + 3 * => 9
  - 2 3 - 5 + 2 * 15 / => 0.533
  - 1 2 3 + + + => Không hợp lệ: Thành phần 6, thiếu toán hạng.

Bài tập 5.2: Hướng dẫn dựa trên ngăn xếp

- Sử dụng một ngăn xếp kiểu số thực (ví dụ double)
- Thực hiện đọc tuần tự từng thành phần của biểu thức và xử lý thành phần đọc được:
  - Nếu đọc được 1 toán hạng thì đưa vào ngăn xếp.
  - Nếu đọc được 1 toán tử thì lấy ra 2 giá trị từ ngăn xếp.
    - Nếu lấy được thì thực hiện phép toán và đưa kết quả vào ngăn xếp, nếu ngược lại thì báo lỗi thiếu toán hạng.
  - Nếu đọc được 1 thành phần không phải toán hạng hoặc toán tử thì thông báo lỗi cú pháp.
  - Sau khi đọc hết các thành phần của biểu thức, nếu trong biểu thức còn tồn tại nhiều hơn 1 giá trị thì thông báo lỗi cú pháp.

Bài tập 5.2: Ví dụ xử lý biểu thức



Bài tập 5.3. Tìm đường thoát mê cung

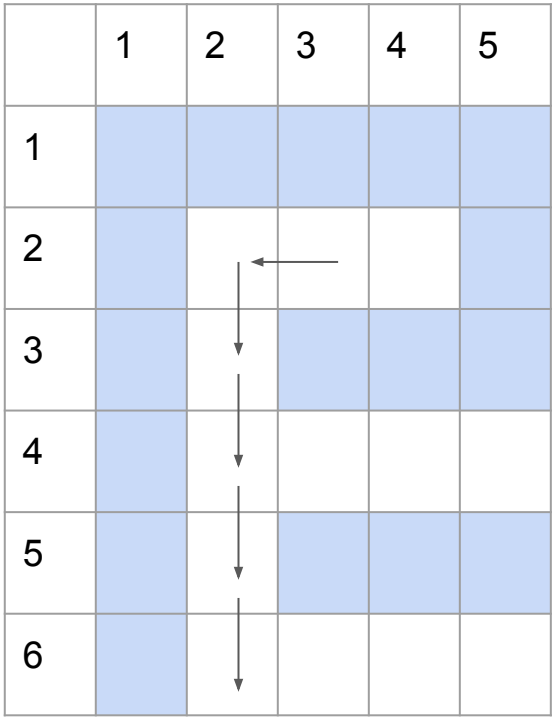
- Sơ đồ một mê cung được cho trong tệp maze.txt dưới dạng 1 ma trận kích thước m x n, các phần tử có thể nhận giá trị 0 hoặc 1, trong đó phần tử a_{ij} = 0 nếu vị trí tương ứng trong mê cung là khoảng trống, a_{ij} = 1 nếu ngược lại - vị trí tương ứng đã bị lấp đầy.
- Viết chương hỏi người dùng nhập vào 2 giá trị i và j là tọa độ tính theo dòng và cột của vị trí hiện tại trong mê cung, chương trình tìm đường đi ra biên và in ra chỉ dẫn di chuyển.
- Giả sử luôn tồn tại đường đi.

**Gợi ý: Sử dụng một ma trận đánh dấu các vị trí đã tới, một ma trận để truy vết ngược và một hàng đợi FIFO để triển khai giải thuật tìm đường đi.*

Bài tập 5.3. Hướng dẫn dựa trên hàng đợi

- Khởi tạo ma trận đánh dấu dd = {0}, hàng đợi FIFO q rỗng, ma trận truy vết bb = {0}
- Thêm vị trí {i, j} vào hàng đợi q và đánh dấu dd[i][j] = 1.
- Lặp cho tới khi hàng đợi rỗng hoặc đã đi tới biên:
  - Lấy vị trí {i, j} ra khỏi hàng đợi
  - Kiểm tra các lân cận {i', j'} trong số {i-1,j}, {i+1,j}, {i,j-1}, {i,j+1}
  - Nếu đã tới {i', j'} thì bỏ qua, nếu ngược lại thì:
    - Đánh dấu dd[i'][j'] = 1; Thêm vị trí {i', j'} vào q và lưu vết bb[i'][j'] = {i, j}
  - Nếu {i', j'} là vị trí biên thì ngừng lặp (đã tìm thấy đường đi).
- Xuất đường đi (luôn tồn tại đường đi theo điều kiện đề bài).
- Có nhiều cách khác nhau để xuất đường đi được lưu theo hình thức truy vết ngược theo chiều thuận:
  - Lưu các vị trí trong đường đi vào mảng rồi xuất ra (đơn giản).
  - Lưu các vị trí trong đường đi vào ngăn xếp rồi xuất ra.
  - v.v..

Bài tập 5.3: Ví dụ minh họa



maze.txt

```
6 5
1 1 1 1 1
1 0 0 0 1
1 0 1 1 1
1 0 0 0 0
1 0 1 1 1
1 0 0 0 0
```

Nhập vào tọa độ vị trí hiện tại: 2 3

Đường ra: (2, 3) -> (2, 2) -> (3, 2) -> (3, 4) -> (3, 5) -> (3, 6)

Nhập vào tọa độ vị trí hiện tại: 3 1

Vị trí không hợp lệ (đã bị lấp đầy)