**BỘ GIÁO DỤC VÀ ĐÀO TẠO**



**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM KHOA CÔNG NGHỆ THÔNG TIN**

🙞🙞🙜🙜

**MÔN HỌC: CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**

**GVHD: TS. HUỲNH XUÂN PHỤNG**

**Đề Tài Cuối Kỳ**

**VIẾT CHƯƠNG TRÌNH MÔ PHỎNG DẠNG HÌNH**

**ẢNH CHO CÁC CẤU TRÚC DỮ LIỆU STACK, QUEUE, DEQUEUE, LINK LIST VỚI CÁC PHÉP TOÁN INSERT, DELETE VÀ CÀI ĐẶT THUẬT TOÁN, VIẾT CHƯƠNG TRÌNH MINH HỌA HÌNH ẢNH CHO THUẬT TOÁN TOPOLOGICAL SORT**

**MÃ MÔN HỌC: DASA230179\_22\_1\_08**

**NHÓM SỐ: 18**

**TÊN THÀNH VIÊN NHÓM - MSSV:**

Nguyễn Minh Khánh 21110507

Nguyễn Hồ Thiên Thanh 21110645

Phan Minh Thuận 21110667

**HỌC KỲ 1 - NĂM HỌC: 2022-2023**

**DANH SÁCH NHÓM**

**HỌC KÌ I, NĂM HỌC 2022 – 2023**

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Họ và tên | MSSV | Tỉ lệ hoàn thành |
| 1 | Nguyễn Minh Khánh | 21110507 | 100% |
| 2 | Lê Nam Hưng | 21110645 | 100% |
| 3 | Phan Minh Thuận | 21110667 | 100% |

***Nhận xét của giáo viên***

*Ngày 31 tháng 12 năm 2022*

**Giáo viên chấm điểm**

**GV.TS. Huỳnh Xuân Phụ****ng**

**Đồ án cuối kì môn cấu trúc dữ liệu và giải thuật, đề tài 1 bổ sung:**

- Viết chương trình mô phỏng dạng hình ảnh cho các cấu trúc dữ liệu stack, queue, dequeue, link list, circle list với các phép toán insert, delete.

- Cài đặt thuật toán, viết chương trình minh họa hình ảnh từng bước thực thi cho thuật toán Topological Sort.

----------------------------------------------------------------------------------------------------

**Chi tiết các nội dung đã làm trong đồ án:**

1. Stack

+ Từ dòng 1 đến 4: khai báo các thư viện.

+ Từ dòng 6 đến 9: tạo một cấu trúc dữ liệu có tên là NodeStack, char so[3] là mảng 3 kí tự, x1, y1 là tọa độ x,y để vẽ, stt là index của số trong mảng.

+ Từ dòng 11 đến 15: khai báo các biến cần dùng trong quá mô phỏng, nS, nTop, nStack là số phần tử để dùng cho các hàm phía sau, x1Khung\_Stack, x2Khung\_Stack, y1Khung\_Stack, y2Khung\_Stack là tọa độ x,y để vẽ từng phần tử trong mảng, mang\_Stack dùng để chứa các phần tử trong mảng, danhDau[] đánh dấu lại các phần tử trong quá trình Thêm để dồn lên, dồn xuống, Xóa trong quá trình mô phỏng.

+ Từ dòng 17 đến 34: hàm con NhapSoStack() dùng để nhập các phần tử cho mảng, char temp[5] là mảng có 5 kí tự dùng để nhập.

+ Từ dòng 36 đến 61: hàm con CapNhatManHinhStack() dùng để hiển thị màn hình mô phỏng với các lệnh outtextxy(), drawText(), rectangle(), line().

+ Từ dòng 63 đến 95: hàm con DiChuyenVaoStack() dùng để thêm phần tử vào Stack trong quá trình mô phỏng.

+ Từ dòng 97 đến 114: hàm con DiChuyenRaStack() dùng để xoá phần tử ra khỏi Stack trong quá trình mô phỏng.

+ Từ dòng 116 đến 122: hàm con SoOTop()dùng để trả về chỉ số của các phần tử ở ngoài cùng ở Top trong mảng lưu rồi so sánh để Xóa phần tử khỏi Stack.

+ Từ dòng 123 đến 160: hàm con MoPhongStack() dùng để chạy chương trình mô phỏng Stack ra màn hình console.

2. Queue

+ Từ dòng 1 đến 4: khai báo các thư viện.

+ Từ dòng 6 đến 9: tạo một cấu trúc dữ liệu có tên là NodeQueue, char so[3] là mảng 3 kí tự, x1, y1 là tọa độ x,y để vẽ, stt là index của số trong mảng.

+ Từ dòng 11 đến 15: khai báo các biến cần dùng trong quá mô phỏng, nQ, nFront, nQueue là số phần tử để dùng cho các hàm phía sau, x1KhungQ, x2KhungQ, y1KhungQ, y2KhungQ là tọa độ x,y để vẽ từng phần tử trong mảng, mangQ dùng để chứa các phần tử trong mảng, danhDau[] đánh dấu lại các phần tử trong quá trình Thêm để dồn lên, dồn xuống, Xóa trong quá trình mô phỏng.

+ Từ dòng 17 đến 34: hàm con NhapSoQueue() dùng để nhập các phần tử cho mảng, char temp[5] là mảng có 5 kí tự dùng để nhập.

+ Từ dòng 36 đến 61: hàm con CapNhatManHinhQueue() dùng để hiển thị màn hình mô phỏng với các lệnh outtextxy(), drawText(), rectangle(), line().

+ Từ dòng 63 đến 95: hàm con DiChuyenVaoQueue() dùng để thêm phần tử vào Queue trong quá trình mô phỏng.

+ Từ dòng 97 đến 115: hàm con DiChuyenRaQueue() dùng để xoá phần tử ra khỏi Queue trong quá trình mô phỏng.

+ Từ dòng 117 đến 123: hàm con SoOFront()dùng để trả về chỉ số của các phần tử ở ngoài cùng ở Front trong mảng lưu rồi so sánh để xóa phần tử khỏi Queue.

+Từ dòng 125 đến 140: hàm con DonLenFront() dùng để dồn phần tử lên đầu.

+ Từ dòng 143 đến 181: hàm con MoPhongQueue() dùng để chạy chương trình mô phỏng Queue ra màn hình console.

3. Dequeue

+ Từ dòng 1 đến 4: khai báo các thư viện.

+ Từ dòng 6 đến 9: tạo một cấu trúc dữ liệu có tên là NodeDequeue, char so[3] là mảng 3 kí tự, x1, y1 là tọa độ x,y để vẽ, stt là index của số trong mảng.

+ Từ dòng 11 đến 15: khai báo các biến cần dùng trong quá mô phỏng, nQ, nFront, nDequeue là số phần tử để dùng cho các hàm phía sau, x1KhungQ, x2KhungQ, y1KhungQ, y2KhungQ là tọa độ x,y để vẽ từng phần tử trong mảng, mang\_Dequeue dùng để chứa các phần tử trong mảng, danhDau[] đánh dấu lại các phần tử trong quá trình Thêm để dồn lên, dồn xuống, Xóa trong quá trình mô phỏng.

+ Từ dòng 17 đến 33: hàm con NhapSoDequeue() dùng để nhập các phần tử cho mảng, char temp[3] là mảng có 3 kí tự dùng để nhập.

+ Từ dòng 35 đến 64: hàm con CapNhatManHinhDequeue() dùng để hiển thị màn hình mô phỏng.

+ Từ dòng 66 đến 97: hàm con DiChuyenVaoDequeueTheoRear() dùng để Thêm phần tử vào ở “cuối” Dequeue trong quá trình mô phỏng.

+ Từ dòng 99 đến 130: hàm con DiChuyenVaoDequeueTheoFront() dùng để Thêm phần tử vào ở “đầu” Dequeue trong quá trình mô phỏng.

+ Từ dòng 132 đến 154: hàm con SoORear() và SoOFront() dùng để trả về chỉ số của các phần tử ở ngoài cùng Rear và Front trong mảng lưu rồi so sánh để Xóa phần tử khỏi Dequeue.

+ Từ dòng 156 đến 222: hàm con DonLenFront() dùng để dồn phần tử lên đầu(Front), DonXuongRear() dùng để dồn phần tử xuống cuối(Rear) của Dequeue.

+ Từ dòng 224 đến 258: hàm con DiChuyenRaDequeueTheoFront() dùng để Xóa phần tử từ đầu(Front) và DiChuyenRaDequeueTheoRear() dùng để xóa phần tử từ cuối(Rear) của Dequeue.

+ Từ dòng 260 đến 314: hàm con MoPhongDequeue() dùng để chạy chương trình mô phỏng Dequeue.

4. Link list

+ Từ dòng 1 đến 5: khai báo các thư viện.

+ Từ dòng 7 đến 11: tạo một cấu trúc dữ liệu có tên là NodeLink, char so[3] là mảng 3 kí tự, x1, y1 là tọa độ x,y để vẽ, stt là index của số trong mảng.

+ Từ dòng 13 đến 17: khai báo các biến cần dùng trong quá mô phỏng, nL, nNode, nLink là số phần tử để dùng cho các hàm phía sau, xNull, yNull là tọa độ x,y để vẽ chữ Null, char sNull[] để vẽ ra chữ Null, mang\_Link dùng để chứa các phần tử trong mảng, danhDau[] đánh dấu lại các phần tử trong quá trình Thêm để dồn lên, dồn xuống, Xóa trong quá trình mô phỏng.

+ Từ dòng 19 đến 35: hàm con NhapSoLink() dùng để nhập các phần tử cho mảng, char temp[3] là mảng có 3 kí tự dùng để nhập.

+ Từ dòng 37 đến 60: hàm con CapNhatManHinhLink() dùng để hiển thị màn hình mô phỏng.

+ Từ dòng 62 đến 68: hàm con SoOLink() dùng để trả về chỉ số của các phần tử trong mảng lưu rồi so sánh để Xóa phần tử khỏi Linklist.

+ Từ dòng 70 đến 168: hàm con DiChuyenVaoLink() dùng để Thêm phần tử vào Link trong quá trình mô phỏng.

+ Từ dòng 170 đến 263: hàm con DiChuyenRaLink() dùng để Xóa phần tử từ khỏi Linklist.

+ Từ dòng 265 đến 295: hàm con MoPhongLink() dùng để chạy chương trình mô phỏng Linklist.

5. Circle list

+ Từ dòng 1 đến 5: khai báo các thư viện.

+ Từ dòng 7 đến 12: tạo một cấu trúc dữ liệu có tên là NodeCircle, char so[3] là mảng 3 kí tự, x1, y1 là tọa độ x,y của Node để vẽ, stt là index của số trong mảng.

+ Từ dòng 13 đến 17: khai báo các biến cần dùng trong quá mô phỏng, nC, nnode, nCircle là số phần tử để dùng cho các hàm phía sau, xnull, ynull là tọa độ x,y đầu tiên của Node khi vào CircleList, mang\_Circle dùng để chứa các phần tử trong mảng, danhDau\_C[] đánh dấu lại các phần tử trong quá trình Thêm để dồn lên, dồn xuống, Xóa trong quá trình mô phỏng.

+ Từ dòng 18 đến 34: hàm con NhapSoCircle() dùng để nhập các phần tử cho mảng, char temp[3] là mảng có 3 kí tự dùng để nhập.

+ Từ dòng 35 đến 66: hàm con CapNhatManHinhCircle() dùng để hiển thị màn hình mô phỏng.

+ Từ dòng 68 đến 187: hàm con DiChuyenVaoCircle() dùng để Thêm phần tử vào CircleList trong quá trình mô phỏng.

+ Từ dòng 189 đến 198: hàm con SoOCircle() dùng để trả về chỉ số của các phần tử trong mảng lưu rồi so sánh để Xóa phần tử khỏi Linklist

+ Từ dòng 199 đến 298: hàm con DiChuyenRaLink() dùng để Xóa phần tử từ khỏi CircleList.

+ Từ dòng 299 đến 335: hàm con MoPhongCircle() dùng để chạy chương trình mô phỏng CircleList.

6. Topological sort mô phỏng

+ Từ dòng 1 đến 4: khai báo các thư viện.

+ Từ dòng 6 đến 12: tạo một cấu trúc dữ liệu có tên là NodeTopo, char so[3] là mảng 3 kí tự, x1, y1 là tọa độ x,y của Node để vẽ và các biến.

+ Từ dòng 14 đến 18: khai báo các biến cần dùng trong quá mô phỏng, nT, nOut, nHuong dùng cho các hàm phía sau, mang\_Topo dùng để chứa các phần tử trong mảng.

+ Từ dòng 19 đến 111: hàm con NhapSoTopo() dùng để nhập các phần tử cho mảng.

+ Từ dòng 113 đến 135: hàm con CapNhatManHinhTopo() dùng để hiển thị màn hình mô phỏng.

+ Từ dòng 137 đến 199: hàm con MoPhongTopo() dùng để chạy chương trình mô phỏng Topological sort.

7. Topological sort

+ Từ dòng 1 đến dòng 4: Khai báo các thư viện

+ Từ dòng 7 đến dòng 11: Tạo một cấu trúc tên là TOPO: có SUC là biến lưu giá trị của Node và next trỏ tới Node sau.

+ Từ dòng 13 đến 17: Tạo một cấu trúc tên là Node: có COUNT là biến lưu giá trị các Node trước hướng vào Node đây, TOP là một TOPO.

+ Từ dòng 19 đến dòng 22: Khai báo các biến: TOPO\* P, mảng QLINK[20] có giá trị ban đầu = 0, mang A chứa các NODE, giá trị n là số lượng Node là hướng.

+ Từ dòng 24 đến 31: Nhập số lượng Node và hướng, gán giá trị COUNT của các Node ban đầu bằng không và trỏ về NULL.

+ Từ dòng 33 đến 43: Nhập các hướng và tăng đầu vào(COUNT) của các NODE có giá trị k lên 1. Tạo một TOPO có giá trị là P, gán giá trị của P->SUC là k và P->next là A[j].TOP và gán A[j].TOP = P vì hướng nhập là từ j đên k. Để ghi lại các hướng ban đầu khi nhập vào.

+ Từ dòng 46 đến 55: Duyệt các giá trị COUNT của các Node nếu bằng 0 thì cho giá trị của Node vào mảng QLINK.

+ Từ dòng 57 đến 75: In ra thứ tự Topological sort. Lấy giá trị đầu của mảng F = QLINK[0], và in ra F, và tạo vòng lặp, sau đó giảm giá trị đầu vào(COUNT) của các Node có Node này hướng vào rồi duyệt xem các giá trị COUNT của các Node nếu bằng 0 thì cho giá trị của Node vào mảng QLINK.