실습 4 – 스택과 큐의 기본 연산

- 실습 목표
  - 스택의 개념을 이해하고, push(), pop() 연산을 구현할 수 있다.
  - 큐의 개념을 이해하고, add(), delete() 연산을 구현할 수 있다.
  - 원형 큐의 개념을 이해하고, add(), delete() 연산을 구현할 수 있다.

스택

- (4.1.) 스택의 구성
  - 정수 배열
  - 전역 변수: int stack[10], top = -1;

- push와 pop 함수를 구현
  - 함수원형 : void push(int item)
  - 함수원형 : int pop()

- main에서 여러 번 push(), pop() 연산을 호출하고, 각 연산이 실행된 후의 스택 상황을 관찰하여 관찰지에 작성

큐

- (4.2.) 큐의 구성
  - 정수 배열
  - 전역 변수: int queue[10], front = -1, rear = -1;

- add와 delete 함수를 구현
  - 함수원형 : void add_q(int item)
  - 함수원형 : int delete_q()

- main에서 여러 번 add(), delete() 연산을 호출하고, 각 연산이 실행된 후의 큐 상황을 관찰하여 관찰지에 작성

원형 큐

- (4.3.) 원형 큐의 구성
  - 정수 배열
  - 전역 변수: int queue[10], front = 0, rear = 0;

- add와 delete 함수를 구현
  - 함수원형 : void add_cq(int item)
  - 함수원형 : int delete_cq()

- main에서 여러 번 add(), delete() 연산을 호출하고, 각 연산이 실행된 후의 원형 큐 상황을 관찰하여 관찰지에 작성

기본 연산의 개선(1)

- 스택과 큐가 저장된 배열을 동적으로 할당
  - int *stack; (큐의 경우, int *queue;)로 구현
  - main에서 stack의 초기 크기로 malloc()
  - 이후, push(큐의 경우, add)에서 stackfull이 되면, realloc()으로 stack 크기를 현재의 두배로 증가
  - 현재 stack의 크기를 전역 변수(max_stack_size)에 저장

기본 연산의 개선(2)

- 스택, 큐, 원형 큐를 C++ 클래스로 정의하여 사용해 볼 것.
  - 배열과 top, max_size 등은 클래스의 private 변수로.
  - 배열의 초기 할당은 클래스의 생성자에서 구현

- 실습 숙제 1
  - 스택과 큐를 C++ 클래스로 구현
  - main 함수에서 스택과 큐를 생성하고, push, pop, add, delete 연산을 각 1000번씩 실행해서 잘 동작하는 지 확인할 것
  - 확장: template 기능을 추가하여 다양한 데이터 타입을 저장할 수 있는 stack과 queue를 구현해볼 것
    - 다음 실습 시간에 활용가능함