1. 자료구조에 대해 정의하시오.

자료구조란, 효율적으로 자료에 접근하고 수정할 수 있도록 자료를 관리하고 저장하는 방법으로 array, linked list, stack, que, tree, graph 등 다양한 종류가 있으며, 처리하고자 하는 데이터와 상황에 적합한 구조를 선택하기 위해서 필수적인 학문이다.

1. 자료구조의 필요성에 대해 설명하시오.

자료를 어떻게 처리하는지에 따라 자료를 삭제하고 추가하는 연산 속도가 달라지며, 연산속도 뿐만이 아니라, 수행하는 알고리즘이 차지하는 메모리를 최소화할 수 있기 때문에 자료구조는 알고리즘과 함께 컴퓨터 과학에서 기본적으로 배워야 할 학문이다.

1. Primitive와 non-primitive 자료구조에 대해 설명하시오.

Primitive 자료구조는 기본적으로 제공되는 자료구조라고 할 수 있으며 integer, pointer, float, character, double 등이 있으며, non-primitive 자료구조는 여러 데이터를 목적에 맞게 효과적으로 저장하는 자료구조로 primitive 자료구조를 조합하여 만든 array, linked list, stack, que, tree, graph 등이 있다. Non primitive 자료구조는 그 안에서 linear 자료구조와 non-linear 자료구조로 분류할 수가 있는데, array, linked list, stack, que 등이 linear 자료구조이고, tree, graph 와 같은 구조가 non-linear 자료구조라고 할 수 있다. Linear 자료구조는 자료들이 서로 앞뒤로 1:1의 연결을 가지는 반면에, non-linear 자료구조는 1:n 또는 n:n 등의 계층적 구조를 가진다.

1. 알고리즘의 시간 복잡도 및 공간 복잡도에 대해 설명하시오.

알고리즘의 시간 복잡도는 알고리즘이 실행되는 시간에 대한 지표로 빠를수록 좋은 알고리즘이라고 할 수 있다. 대표적인 표기법으로는 Big-O 표기법이 있으며, 이는 worst case complexity를 기준으로 표기하며, 예를 들자면, 배열에서 한번의 for loop순환을 하는 알고리즘의 경우 O(n)과 같이 표기한다. Worst case 시나리오에 대해서 설명하자면, 앞의 예시의 경우, 크기가 n인 배열에서 가장 마지막에 열어본 주소가 우리가 찾고자 하는 element이거나 배열에 찾고자 하는 element가 없는 경우 수행횟수가 가장 많은 worst case이며 이때가 n이기 때문에 O(n)으로 표기를 하는 것이다. Big O 표기법 외에도 average case, best case를 기준으로 하는 표기법이 존재하지만 big O 표기법이 가장 많이 쓰인다.

알고리즘의 공간 복잡도는 알고리즘이 메모리에서 차지하는 공간을 나타내는 지표로 적은 공간을 차지할수록 좋은 알고리즘이다. 기술의 발전으로 RAM의 용량이 증가하며 시간 복잡도에 비해서 중요성이 최근에는 상대적으로 떨어지는 효율성 지표이다.

1. 배열의 크기를 초과하는 배열 element에 값을 할당하면 C 프로그램에서 어떤 일이 발생합니까? 예를 들어 설명하시오.

JS나 Python과 같이 array의 크기를 백그라운드에서 자동으로 처리해주는 언어와는 다르게 C는 사용자가 하나하나 지정해야 하는 차이점이 존재한다. 이와 같은 특성으로 인해서, C에서는 앞에서의 다른 프로그램보다 사용자가 엄밀하게 알고리즘을 다룰 수 있으며 더 효율적인 코드를 짤 수 있다는 장점과 함께 이로 인해서 생기는 문제에 대한 주의가 필요하다는 단점이 있다. 이로 인해서 생기는 문제인 Index out of range는 C 컴파일러가 스스로 error로 잡지 못하고 배열의 인덱스 할당 범위를 벗어나도 일정 범위의 메모리에 값을 할당해버리기 때문에 일어나는 문제이다. 이로 인해서 메모리 오류나 다른 데이터를 손상시킬 수 있다. 예를 들자면, 크기가 5 인 array 가 base address가 1000인 메모리 공간에 저장이 되었다고 해보자. int arr[5]로 integer array를 5의 크기로 선언해준 뒤에 arr[5] = num; 과 같이 배열의 크기를 초과하는 배열 element에 값을 할당하면 1000~1004에서 선언된 array의 다음 주소인 1005에 접근을 하기 때문에 컴파일러나 컴퓨터에 따라서 여기에 저장된 값의 변화로부터 다양한 오류가 생길 수가 있다.

1. Row and Column major order는 무엇인지, 메모리에 어떻게 표현되는지 설명하시오.

2차원 배열이라도 메모리에 저장될 때는 1차원의 array형태로 저장이 되는데, 이 때의 순서를 결정하는 것이 row and column major order이다. Row major order의 경우는 row 단위로 하나의 row를 저장하고 다음 row를 그 옆에 저장하는 방법이며, column major order는 마찬가지로 column 단위로 하나의 column을 저장하고 다음 column을 옆에 저장하는 방법이다.

1. *n*개의 요소로 구성된 배열 Arr에서 주어진 요소 *k*를 삭제하는 알고리즘/의사 코드를 작성하시오. (요소 *k*가 항상 Arr에 존재한다고 가정)

int\* arr; int k; int n; int index; // index = -1과 같이 설정한다면 배열에 k가 없는 경우를 index == -1 로 지정해서 if 문을 통해서 다룰 수 있다

for (int i=0; i<n; i++) // 배열을 순환하게 하는 for loop

if (arr[i]==k) // 요소 k를 찾았다면 이 경우에는 항상 존재하므로

index = i ; // index 에 이때의 i를 저장

break; // 순환을 멈춤

// 마지막 요소가 k 인 경우

If (index == n-1)

n--; // 단순하게 마지막 요소를 삭제

// 마지막 요소가 k가 아닌 경우

// array에서는 삭제를 위해서 arr[n] = arr[n+1]로 값을 덮어씌워야 하므로

// for loop를 한번도 이용

for (int i = index; i<n-1; i++) // 마지막 요소가 아닌 경우

arr[i]=arr[i+1]; // for loop를 이용해서 한칸씩 옆으로 밀어넣는다

n--; // 마지막 요소 삭제

/\* self-feedback

이 의사코드에서는 for loop를 2번 사용하였으므로 이를 개선하기 위해서는 첫번째 for loop에서 k값의 확인과 동시에 if 문을 통해서 2번째 for loop에서의 밀어넣는 작업을 수행한다면 array를 1번만 순환하며, 삭제 작업을 수행할 수 있을 것 같다. 또한 scanf를 통해서 k를 입력받아 변수로 저장하면 더 유동적인 함수를 만들 수 있을 것 같다.

\*/