01 프로그래밍 프로젝트

01 [추상 자료형] 배열은 C언어에서 기본적으로 제공하는 중요한 자료 구조중의 하나이다. 배열이 제공하는 기능을 추상 자료형으로 만들어 보자. 배열에서 핵심적인 것은 인덱스 위치에 데이터를 저장하고 추출하는 것이다. 따라서 배열을 추상 데이터 타입으로 만든다면 다음과 같을 것이다.

배열

- 객체: 〈인덱스, 요소〉 쌍의 집합
- 연산:
- retrieve(i) ::= 배열 A의 i번째 요소 반환.
- store(i, item) ::= 배열 A의 i번째 위치에 item 저장.

(1) 위의 추상 데이터 타입의 연산들만을 이용하여 배열을 하나 생성하고, 배열에 1부터 5까지의 정수를 저장한 다음, 다시 배열에 저장된 정수들을 추출하여 합을 구하는 프로그램을 작성하라. 주의할 점은 반드시 추상 데이터 타입에서 제공된 연산만을 사용하여야 한다. 하나의 예는 다음과 같다. 빈칸을 채워라

```
#define SIZE 5
main()
{
     int i, sum=0;
     for(i=0;i<SIZE;i++)
          _____;       // 배열에 값 저장
     for(i=0;i<SIZE;i++)
          sum += _____;   // 배열에서 값 추출
}
```

(2) 정수 배열을 이용하여 배열 추상 자료형의 연산들을 store와 retrieve를 구현해보라.

```
int iarray[SIZE];
void store(int i, int item)
{
     _____
}
int retrieve(int i)
{
     _____
}
```

이들 함수들을 사용하여 (1)의 프로그램과 함께 컴파일하여 수행하여 보라.

(3) 구현과 분리된 추상자료형의 장점을 알기 위하여 추상 자료형의 구현을 다음과 같이 변경시킨 다음에 (1)의 프로그램과 함께 컴파일하여 수행시켜보라. (1)의 프로그램을 변경할 필요가 있는가?

```
float farray[SIZE];
void store(int i, int item)
{
```

```
        farray[i] = (float)item;
    }
    int retrieve(int i)
    {
        return (int)farray[i];
    }
```

02 [수행 시간 측정] 1부터 n까지의 합을 구하는 프로그램을 다음과 같이 3가지 방법으로 알고리즘을 만들어보고 각 알고리즘들을 실제로 구현하여 각 알고리즘의 실제수행 시간을 측정하여 이론적인 분석과 같게 나오는지를 조사해보라.

- 알고리즘 A: $sum = n(n+1)/2$공식사용
- 알고리즘 B: $sum = 1 + 2 + \cdots + n$
- 알고리즘 C: $sum = (1) + (1+1) + (1+1+1) + \cdots, (1+1+\cdots+1)$

수행 시간측정을 쉽게 하기 위하여 n을 상당히 크게 하고 중간에 고의적인 시간지연함수를 넣거나 같은 합산을 여러 번 반복하라. 각 n값에 대하여 수행 시간을 측정하여 표로 나타내라.

03 [시간복잡도] 아래의 프로그램 A는 $O(n)$이고 프로그램 B는 $O(n^2)$이지만 n이 작을 때는 프로그램 B가 더 빠를 수도 있다. 두 개의 프로그램을 구현하여 수행 시간을 측정하여 n이 어느 정도로 커져야 프로그램 B가 시간이 더 걸리는지를 실험하라.

```
// 프로그램 A
    mul = 1;
    for(i=0;i<n;i++)
        for(j=1;j<1000;j++)
            mul = mul *j;

// 프로그램 B
    mul = 1;
    for(i=0;i<n;i++)
        for(j=1;j<n;j++)
            mul = mul *j;
```