추상자료형(abstract data type, ADT)

〈추상자료형 정의〉

객체의 **명세**와 이들 **객체에 대한 연산의 명세**가 **객체의 표현**과 **연산의 구현**으로부터 **분리**된 방식으로 구성된 **데이터 타입**이다.

〈이석호 자료구조론〉

- **추상자료형 = 객체 + 연산** (**연산**에서는 **세부사항** 무시하고 **기능**을 강조)
- **추상자료형**은 사람마다 조금씩 다르게 정의할 수 있다.

- 추상자료형 정의는 참 **복잡**한 느낌을 받는다. 이해가 잘 안될 수도 있다.
- 해서, 추상자료형 예를 들어서 설명한다.

// 정수에 대한 사칙연산의 추상자료형

객체	컴퓨터가 가질 수 있는 **정수** 집합
연산 (함수)	add(a, b) ::= 두 정수 a, b에 대한 덧셈 sub(a, b) ::= 두 정수 a, b에 대한 뺄셈 mul(a, b) ::= 두 정수 a, b에 대한 곱셈 div(a, b) ::= 두 정수 a, b에 대한 나눗셈

- **추상자료형 = 객체 + 연산** (**연산**에서는 **세부사항** 무시하고 **기능**을 강조)

// 자연수에 대한 추상자료형

객체	**0**에서 시작하여 해당 컴퓨터가 가질 수 있는 **최대 정수(max_int)**까지 값
연산 (함수)	zero() ::= return **0**; is_zero(**x**) ::= if(**x**) return **false**; else return **true**; equal(**x, y**) ::= if(**x==y**) return **true**; else return **false**; add(**x, y**) ::= if(**x+y**<=**max_int**) return **x+y**; else return **max_int** sub(**x, y**) ::= if(**x**<**y**) return **0**; else return **x−y**; successor(**x**)::= if(**x**<=**max_int**) return **x+1** else return **x**;

// 스택(stack) S에 대한 추상자료형

객체	**0**개 이상의 원소를 가진 유한의 순서리스트 (여기서, 객체는 **데이터**를 의미)
연산 (함수)	create(**s**) ::= 스택 **s** 생성 is_empty(**s**) ::= 스택 **s**가 비어 있는지 검사 is_full(**s**) ::= 스택 **s**가 가득 찼는지 검사 push(**s, e**) ::= 스택 **s**의 맨 위에 요소 **e**를 삽입 pop(**s**) ::= 스택의 맨 위에 있는 요소를 삭제

예제 1	객체지향언어 Java에서 추상자료형

```
interface A                         // 추상자료형 - 추상 데이터 타입(ADT)
{
  public int add(int a, int b);     // 메소드 선언 - 덧셈
  public int sub(int a, int b);     // 메소드 선언 - 뺄셈
}
class B implements A                // 추상자료형을 상속받아서 구현
{
  public int add(int a, int b){ return a + b; };       // 메소드 구현
  public int sub(int a, int b){ return a - b; };       // 메소드 구현
}
public class Test
{
  public static void main(String ar[])
  {
    A p = new B();
    System.out.println(p.add(100, 2));  // 출력 : 102
    System.out.println(p.sub(100, 2));  // 출력 : 98
  }
}
```

- Java의 **interface**는 **추상자료형**의 한 종류이다.
- Java의 **interface**는 객체에 대한 **명세**만 기술한 것이다.(구현은 기술하지 않음)
- 객체에 대한 **구현**은 상속받은 **클래스**에서 기술한다.

- 이런 것을 간단하게 **객체**에 대한 **명세**와 **구현**을 **분리**한다. 라고 말한다.
- 즉, **명세**만 한 것을 **추상자료형**이라 말한다.

- **명세**는 영어단어로 **specification**이다.
- 영어단어 **specification**의 뜻은 **설명서**이다.
- 어떤 것에 대해 **명세**한다는 것은 **설명**해 둔 것이다.(보는 사람이 쉽게 이해할 수 있도록)
- **명세**한다는 것은 사람이 쉽게 이해할 수 있도록 **설명**해 둔 것이다.
- **명세**만한 것은 **컴퓨터**에서 실행될 수 없다.
- **명세**만한 것이 **컴퓨터**에서 실행되려면 **구현**해야 한다.
- **구현**은 **별도**로 하는 것이다.

예제 2	C에서 추상자료형

```
int add(int a, int b);     // 함수 선언(두 정수의 덧셈) - 추상 데이터 타입(ADT)
int sub(int a, int b);     // 함수 선언(두 정수의 뺄셈) - 추상 데이터 타입(ADT)
int add(int a, int b)              // 함수 구현
{
  return a + b;                    // 덧셈 연산
}
int sub(int a, int b)              // 함수 구현
{
  return a - b;                    // 뺄셈 연산
}
void main()
{
  printf("%d\n", add(100, 2));     // 출력 : 102
  printf("%d\n", sub(100, 2));     // 출력 : 98
}
```

- 추상자료형 개념은 특정 언어에만 적용되는 것이 아니다.
- 추상자료형은 각 언어가 가지는 특징을 이용하여 적용할 수 있다.
- 단지, C는 독자적인 **명칭**을 가진 **추상자료형**을 지원하지 않는다.
- 위의 C 코드에서 **함수**의 **기능**만 **기술**한 것이 **추상자료형**이다.(내용은 없는 것)
- C 코드에서 **추상자료형** 부분은 확장자 **.h**로 끝나는 헤더파일을 이용하여 처리할 수 있다.
- 함수 **내부**의 **구현**은 **기술하지 않은 것**을 **추상자료형(ADT)**이라 한다.
- 함수 내부의 자세한 구현은 별도로 기술하는 것이다.(함수 선언과 구현을 **따로** 한 것)
- 이런 것을 '**ADT**는 구현에 독립적이다' 라고 한다. ADT는 **사용설명서**와 같은 것이다.
- ADT를 통해서 데이터나 연산이 **무엇**(what)인지는 알 수 있다.
- ADT를 통해서 데이터나 연산이 **어떻게**(how) 동작하는지는 알 수 없다.
- 사용자는 ADT에 접근하여 **사용**할 수는 있지만, 직접 **변경**은 **불가능**하다.
- 여기서, '사용자는 직접 **변경**은 **불가능**하다' 라는 것을 잘 이해해야 한다.
- **예** : C에서 사용자는 함수 **printf()**를 **사용**할 수는 있지만, 직접 **변경**은 **불가능**하다는 것

주의할 핵심 내용	**데이터 타입(자료형)** = **객체** + 객체에 동작하는 **연산**의 집합

- **객체**(object)는 1, 2, 3, 4, … 같은 **값**을 나타내는 것이다.
- **연산**(operation)은 객체끼리 더하고(+), 빼고(-) 등을 **처리**하는 것이다.
- 여기서, 주의할 것은 **자료형**에는 **객체**만 있는 것이 아니라 그에 대한 **연산**이 **포함**되어 있다.

예제 3	객체지향언어 C++에서 추상자료형

```
#include <iostream>
using namespace std;
class A                          //클래스 선언부(추상자료형)
{  public:                       //C++에서 추상자료형은 키워드 class를 이용하여 정의함
    int add(int a, int b);       //메소드 선언 - 두 정수의 덧셈
    int sub(int a, int b);       //메소드 선언 - 두 정수의 뺄셈
};
int A::add(int a, int b)         //클래스 구현부 (외부로부터 숨겨짐, 정보은닉)
{
  return a + b;                  //메소드 구현 - 두 정수의 덧셈 연산 실시
}
int A::sub(int a, int b)         //클래스 구현부 (외부로부터 숨겨짐, 정보은닉)
{
  return a - b;                  //메소드 구현 - 두 정수의 뺄셈 연산 실시
}
int main()
{
  A p;
  cout<<p.add(100, 2)<<endl;     // 출력 : 102
  cout<<p.sub(100, 2)<<endl;     // 출력 : 98
  return 0;
}
```

- 클래스 **선언부**가 **추상자료형**에 해당한다.
- C++에서 **추상자료형**은 키워드 **class**로 선언한다.

- 추상자료형의 큰 특징 중 하나는 **사용자**와 **구현자**의 **분리**이다.
- 어떤 사용자가 추상자료형을 **선언**하고, 다른 사람이 선언된 것을 **구현**할 수 있는 것이다.
- 추상자료형은 **사용 관점**과 **구현 관점**을 명확히 **분리**하여 프로그램 개발을 유용하게 한다.
- 클래스와 추상자료형은 완전히 **같은 것**은 **아니다.**(클래스는 추상자료형 이외 다른 것이 포함됨)
- C++에서는 키워드 **class**를 이용하여 **추상자료형**을 선언할 수 있고, **일반 클래스**를 정의한다.

예제 4	C++에서 추상자료형과 클래스

```
class Phone                          //클래스 선언부(추상자료형)
{   public:
       void call();                  //메소드 선언 (전화 걸기)
       void send();                  //메소드 선언 (메시지 보내기)
};
void Phone::call()                   //클래스 구현부
{
  cout<<"전화를 걸다"<<endl;         //메소드 구현
}
void Phone::send()                   //클래스 구현부
{
  cout<<"메시지를 보내다"<<endl;     //메소드 구현
}

class StartPhone : public Phone      //자식클래스 : 부모클래스
{
  public: void wifi()
  {
    cout<<"인터넷을 한다"<<endl;     //메소드 정의
  }
};

int main()
{
  StartPhone *sp = new StartPhone();
  sp->call();                        // 출력 : 전화를 걸다
  sp->send();                        // 출력 : 메시지를 보내다
  sp->wifi();                        // 출력 : 인터넷을 한다.
  return 0;
}
```

• C++에서는 키워드 class를 이용하여 **추상자료형**을 선언할 수 있고, **일반 클래스**를 정의한다.

기출문제 분석

1. 추상자료형(abstract data type)에 대한 설명 중 가장 적절하지 않은 것은? [2021년 군무원 7급]

① 추상자료형은 자료(data)와 연산(operation)으로 구성된다.
② 추상자료형은 자료구조의 효율적인 구현 방법을 명시하여야 한다.
③ 추상자료형은 객체지향언어의 클래스(class)의 개념과 유사하다.
④ 추상자료형은 수학에서 대수구조(algebraic structure)의 개념과 유사하다.

♤ **추상자료형**

// 예제 : 스택(stack) S에 대한 추상자료형

객체	**0**개 이상의 원소를 가진 유한의 순서리스트 (여기서, 객체는 **데이터**를 의미)
연산 (함수)	create(**s**) ::= 스택 **s** 생성 is_empty(**s**) ::= 스택 **s**가 비어 있는지 검사 is_full(**s**) ::= 스택 **s**가 가득 찼는지 검사 push(**s, e**) ::= 스택 **s**의 맨 위에 요소 **e**를 삽입 pop(**s**) ::= 스택의 맨 위에 있는 요소를 삭제

- **추상자료형 = 객체 + 연산** (**연산**에서는 **세부사항** 무시하고 **기능**을 강조)
- **추상자료형**은 자료구조의 효율적인 구현 방법을 명시하지 않는다.

// 대수(algebra, 代數)

- **대수**(代數)는 직역하면, **수를 대신**한다는 의미이다.
- **대수**는 수를 대신하여 문자를 사용하여 **방정식**을 푸는 방법을 연구하는 학문에서 시작되었다.

대수학 연구분야	· 군론(group theory) – 군에 대한 연구 · 환론(ring theory) – 환에 대한 연구 · 체론(field theory) –체에 대한 연구 · 정수론(number theory) – 각종 수의 성질을 대상으로 연구 · 대수기하학(algebraic geometry) – 직교 좌표계의 점들 표현에 대한 연구 · 선형대수학(linear algebra) – 벡터, 선형변환, 행렬, 연립선형방정식 등을 연구 · 추상대수학(abstract algebra) – 군, 환, 체, 벡터공간, 대수학 등에 대해서 공부

- **추상대수학**은 덧셈, 곱셈 등이 정의된 집합(대수계)을 추상적으로 연구하는 학문이다.
- **대수구조**는 **원소의 집합**과 **연산**을 함께 묶어낸 수학적 개념이다.
- **원소의 집합**은 자연수, 정수, 실수 등이 될 수 있고**(원소의 집합은 클래스의 속성과 유사)**
- **원소의 연산**은 덧셈, 곱셈, 뺄셈 등이 될 수 있다.**(원소의 연산은 클래스의 메서드와 유사)**

정답 : ②

2. C 언어의 추상 데이터 타입(abstract data type)에 대한 설명으로 가장 옳지 않은 것은? [2022년 군무원 7급]

① 사용자들은 추상 데이터 타입 내부의 데이터에 직접 접근하여 사용하고 변경 가능하다.
② 사용자들은 추상 데이터 타입이 제공하는 연산만을 사용할 수 있다.
③ 만약 다른 사람이 추상 데이터 타입의 구현을 변경하더라도 인터페이스가 변경되지 않는다면 사용자들은 추상 데이터 타입을 같은 방식으로 사용할 수 있다.
④ 추상 데이터 타입은 데이터 타입 정의가 그 데이터 타입의 구현으로부터 분리된 데이터 타입을 말한다.

♤ **추상 데이터 타입 – 추상자료형**

```
// 먼저, 추상 데이터 타입에 대해 C 코드를 이용하여 예를 들어 설명한다.
 int sum(int a, int b);     // 함수 선언(두 정수의 합) - 추상 데이터 타입(ADT)
 void main()
 {
   printf("%d\n", add(1, 2));
 }
 int sum(int a, int b)      // 함수 구현
 {
   return a + b;            // 연산
 }
```

- 위의 코드에서 **추상적**이라는 것은 **함수**의 **기능**만 **기술**한 것을 말한다.(내용은 없는 것)
- 함수 **내부**의 **구현**은 **기술하지 않은 것**을 **추상 데이터 타입(ADT)**이라 한다.
- 함수 내부의 자세한 구현은 별도로 기술하는 것이다.(함수 선언과 구현을 **따로** 한 것)
- 이런 것을 'ADT는 구현에 독립적이다' 라고 한다. ADT는 **사용설명서**와 같은 것이다.
- ADT를 통해서 데이터나 연산이 **무엇**(what)인지는 알 수 있다.
- ADT를 통해서 데이터나 연산이 **어떻게**(how) 동작하는지는 알 수 없다.
- 사용자는 ADT에 접근하여 **사용**할 수는 있지만, 직접 **변경**은 **불가능**하다.
- 여기서, '사용자는 직접 **변경**은 **불가능**하다' 라는 것을 잘 이해해야 한다.
- **예** : C에서 사용자는 함수 **printf()**를 **사용**할 수는 있지만, 직접 **변경**은 **불가능**하다는 것

주의할 핵심 내용	**데이터 타입(자료형)** = **객체** + 객체에 동작하는 **연산**의 집합

- **객체**(object)는 1, 2, 3, 4, … 같은 **값**을 나타내는 것이다.
- **연산**(operation)은 객체끼리 더하고(+), 빼고(–) 등을 **처리**하는 것이다.
- 여기서, 주의할 것은 **자료형**에는 **객체**만 있는 것이 아니라 그에 대한 **연산**이 **포함**되어 있다.

정답 : ①