Data Type vs Abstract Data Type vs Object

2013210111 남세현

Data Type

C언어같은 고급언어가 나오면서, 우리는 기계 중심이 아니라 사람 중심으로 문제를 해결하기 시작합니다. 그 시도 중 하나는 바로 Data Type입니다.

Data Type이란, 수학적인 모델, 현실에 있는 것들을 모델화한 것 입니다. 그것으로 컴퓨터 작업에 필요한 정보를 담아둘 수 있습니다.

흔히 우리가 int로 알고있는 integer, Floatting Point Numbers, boolearn 등이 바로 그것입니다.

int a = 0x0FF1CE;

float f = 5.5f;

bool isDead = true;

Abstract Data Type

위 Data Type으로부터 더 나아가, 어떤 데이터의 구조, 모양새나 동작방식 등을 추상적으로 만들

말이 어렵지만, 쉽게 생각하면 많은 컴퓨터 과학자들이 컴퓨터 문제를 풀면서 자주 쓰이게 되는 자료구조(Structure)가 있을 것입니다. 자료구조들은 어떤 문제를 푸느냐에 따라 구현(Implementation)이 다를것이고, 어떤 시스템에서 동작하느냐에 따라도 많이 다를 것입니다.

그런 구체적인 것들을 제외한 것이 바로 ADT입니다. **가장 추상적으로 그 자료모델이 필요한 핵심만 짚어놓은 것** 입니다.

예시를 들면, **List는 ADT**입니다. 하지만 STL등에서 제공하는 **ArrayList, LinkedList 등은 자료구조**입니다. Map이란 것도 ADT라면, 그 Map의 철학을 이어받아 직접 구현한 HashMap, HeapMap 등은 자료구조입니다.

(이어서)

제가 이 수업을 들으면서 이해한 ADT의 특징 중 하나는 바로 Data Encapsulation입니다. Public 요소와 Private 요소를 적절히 분배해서, 사용자는 그 내부를 알지 못해도 상관없이 사용할 수 있도록 Interface를 제공하는 것이 ADT의 기본인 것 같습니다.

요즘 수업시간에 나오는 Stack을 예시로 들면,

(stack.h 파일에서)

```
Template<Typename T>
Class Stack
{
        ...
        public:
        void Push (T&);
        T Pop();
        T Top();
        private:
        std::list<T> m_List;
}
```

(stack.cpp 파일 생략)

이런 식으로 public 요소로 stack에 접근할 순 있지만 그 내부의 모양새나 implementation(cpp파일) 등은 숨겨놓음으로서 '**추상적**'으로 만들 수 있습니다. 그 새부 내용을 어떻게 만드느냐~ 가 바로 Data Structure라고 할 수 있겠습니다.

Object

Data Type에서 언급한 것 처럼, 고급 언어와 함께 문제 해결하는데 있어서의 패러다임이 기계 중심에서 사람 중심, 세상 중심으로 바뀌기 시작합니다.

세상에는 수 많은 개념들과 수 많은 '것'들이 있습니다. 마찬가지로 프로그래밍 할 때 단순히 데이터들의 집합으로 문제를 해결하는 것이 아니라, **'것'(객체, object)들의 집합으로 문제를 바라보겠다는 것입니다.**

객체가 어떻게 생겼고, 어떤 일을 하는지 정의를 하고, 그 객체들의 **실체(instance)**를 만들어가면

서 문제를 해결하는 것 입니다.

C++에서의 implementation으로 예시를 들면

```
Class printer
{
        ...
        Void DoPrint(...);
}
```

이런식으로 프린팅을 하는 객체 Printer를 정의할 수 있고

Printer printerA;

프린터A라는 이름으로 객체 Printer의 실체를 하나 만들 수 있으며,

printerA.DoPrint(...)

그 실체에게 행동을 시킴으로서 문제 해결을 해나갈 수 있는 것입니다.

유사점

세 개념 다 메모리를 어떻게 사용할 것인가 고민을 하며 나온 개념입니다.

실제로 메모리에 할당하여 문제를 해결하는데 핵심이 될 수 있습니다.

차이점

Data Type은 특별히 사용자가 **바꾸거나 할 수 없고, 그럴 만한 특별한 Implementation도 없습니다**.

ADT 역시 **특별히 바꿀 수 있는 것은 없습니다만, 그 내부의 Implementation을 바꿈으로서 천차만별**이 될 수 있음이 Data Type과 차이점입니다.

Object는 ADT와 비슷한 점을 가지고 있지만, **수학적, 컴퓨터과학적 모델이 아니라 정말 다양한 분야, 다양한 문제에 적용할 수 있는 점**이 다릅니다(예를 들면, 게임의 캐릭터 오브젝트 등).