Part

01

자료구조 시작하기

Chapter 01

자료구조

*** 학습목표**

- 자료구조의 의미와 중요성을 알아본다.
- 자료구조에서 다루는 내용을 알아본다.
- 컴퓨터 내부의 2진수 코드 체계를 알아본다.
- 자료의 형태에 따른 자료 표현 형식을 알아본다

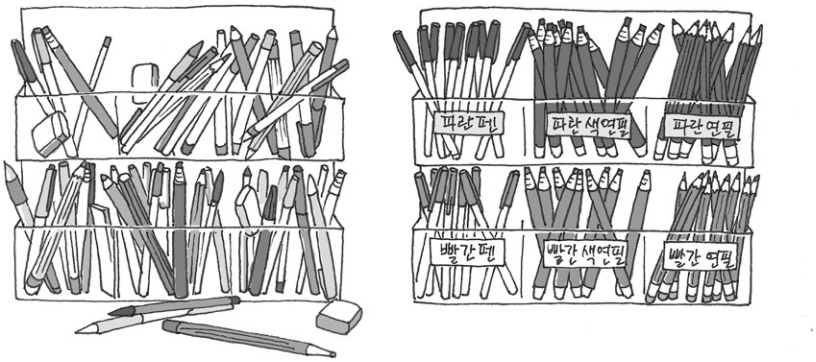
자료구조는 컴퓨터에서 사용할 자료를 더 효율적으로 저장하고 처리하기 위해서 자료의 특성과 사용 용도에 따라 분류하고 정리하는 것, 즉 구조화하는 것입니다. 공부하려고 책상에 앉았는데 필요한 것을 찾느라 시간만 보내서 짜증난다면, 우선 책상 정리부터 해보세요. 책상이 본인 개성에 맞도록 정리 정돈되어 구조화된 책상이라면 공부 효율도 더 올라가겠죠?

Section

01 자료구조 개요

우리는 인터넷, TV, 책 등 매일 엄청난 양의 자료 속에서 살아가고 있다. 사회가 정보화, 다양화될수록 우리가 사용할 수 있는 자료 또한 더욱 풍부해지고 다양해진다. 정보의 홍수라고 할 만큼 자료가 방대한 지금은 얼마나 많은 자료를 가지고 있느냐 보다는 가지고 있는 자료를 얼마나 효율적으로 사용하느냐가 중요한 시대가 되었다.

예를 들어, 문구점에서 필기구를 진열하는 방법을 살펴보자. 종류와 색상에 관계없이 판매하는 모든 종류의 필기구를 같은 상자 안에 넣어둘 수도 있고, 종류별·색상별로 분류해서 진열할 수도 있다. 원하는 필기구를 찾을 때 어떤 방법이 더 효과적일까? 단순하게 생각해봐도 종류별·색상별로 분류하는 방법이 더 효과적이다. 이와 마찬가지로 자료를 효율적으로 표현하고 저장, 처리하기 위해 정리하는 것이 자료구조이다.

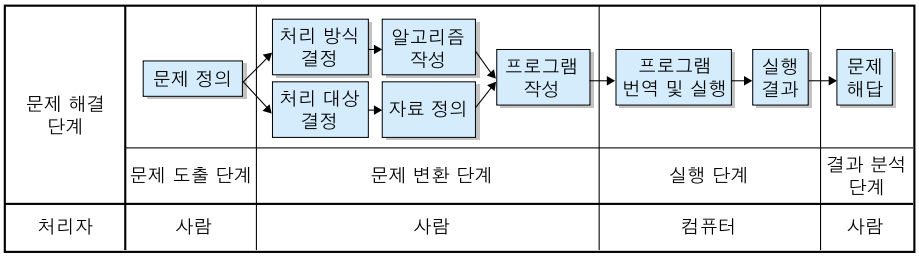


[나쁜 자료구조] [좋은 자료구조]

[그림 1-1] 자료구조의 적용 예

그렇다면 컴퓨터에서는 자료구조가 어떻게 사용되고 있을까? 컴퓨터에서는 자료를 효과적으로 표현하고 효율적으로 저장, 처리할 수 있도록 논리적인 구조로 설계하고 분석하여 프로그램에서 사용한다. 컴퓨터는 자료를 처리하는 장치이므로 컴퓨터 관련 분야에서 자료구조는 기본적이고도 필수적인 개념이다.

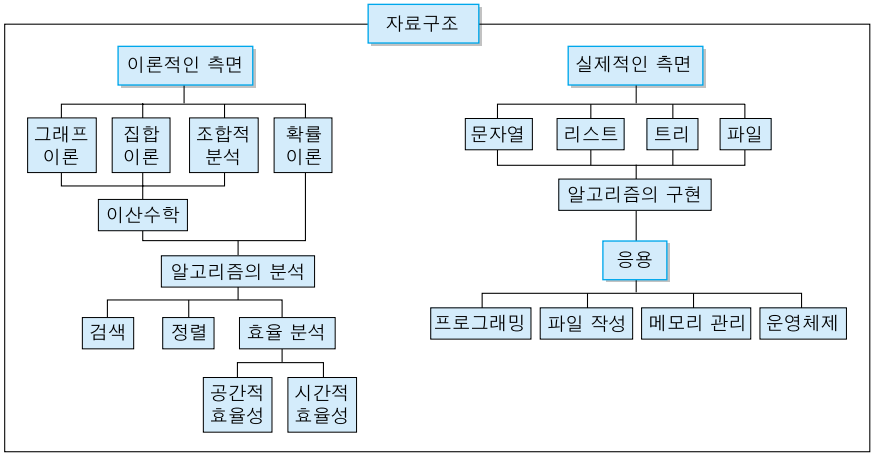
"모든 일은 컴퓨터가 하는데 왜 우리가 자료구조를 공부해야 하는가?"라는 의문을 가질 수 있다. 이런 의문을 풀기 위해 컴퓨터가 문제를 해결하는 과정을 살펴보자.



[그림 1-2] **컴퓨터의 문제 해결 과정**

컴퓨터가 효율적으로 자료를 처리하려면 [그림 1-2]와 같이 문제 도출 단계와 문제 변환 단계에서 문제를 자료구조 측면에서 분석하고 구성해야 한다. 사용자(프로그래머)는 문제를 좀 더 효율적으로 해결하기 위해 문제를 정의하고 처리 방식을 결정하여 알고리즘을 작성하고, 자료를 정의해야 한다. 이러한 과정에 자료구조에 대한 개념과 활용 능력이 필요하다.

자료구조에서 다루는 내용은 [그림 1-3]과 같다.



[그림 1-3] **자료구조의 내용**

자료구조는 이론적인 측면과 효율적인 측면, 실제적인 측면에서의 응용을 모두 다루어야 한다. 이론적인 측면에서는 그래프 이론, 집합 이론, 조합 이론의 이산수학과 확률 이론을 기본

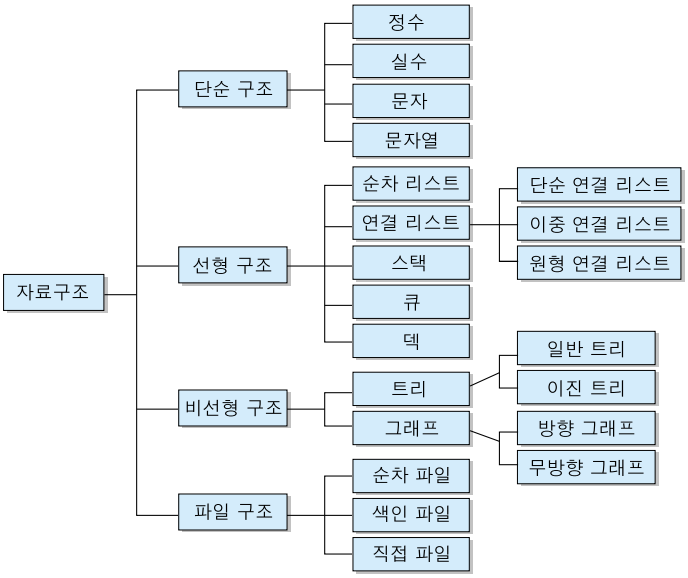
으로 알고리즘을 분석하여 검색 · 정렬 방법 등을 결정하고, 효율적인 측면에서는 공간적 효율성과 시간적 효율성을 기준으로 최상의 상태를 결정한다. 이렇게 결정한 내용에 따라 자료를 문자열, 리스트, 트리, 파일 등의 구조로 실제적으로 표현하고 알고리즘을 구현하여, 프로그램과 파일 작성 및 메모리 관리, 운영체제 등에 사용한다.

성공적인 시스템을 개발하는 고급 개발자가 되기 위해서는 자료의 특성을 이해하고, 분석하여 최적의 알고리즘을 개발하는 능력이 필요하다. 이러한 능력을 키우기 위해서 자료구조에 대한 학습이 필요하다.

Section

2 자료구조의 분류

자료를 형태에 따라 분류하면 프로그래밍 언어에서 배운 정수, 실수, 문자, 문자열 등의 데이터 타입에 해당하는 '단순 구조'와 자료 간의 연결 관계가 1:1 관계를 가지고 있는 '선형 구조', 1:다 또는 다:다 관계의 '비선형 구조', 그리고 '파일 구조'로 나눌 수 있다. 표현할 자료의 특성과 양, 자료의 주된 사용 방법과 수행하는 연산의 종류, 구현에 필요한 기억 공간 용량을 고려하여 가장 효율적인 자료구조를 선택해야 한다.



[그림 1-4] 자료구조의 형태에 따른 분류

■ 선형 구조

자료 간의 연결 관계가 1:1 관계를 갖는 구조로 순차 리스트(Sequential List)와 연결 리스트(Linked List), 스택(Stack), 큐(Queue), 덱(Deque) 등이 있다.

순차 리스트는 자료의 논리적인 순서와 기억 장소에 저장되는 물리적 순서가 일치하는 구조이며, 연결 리스트는 물리적인 순서에 상관없이 포인터를 사용하여 연결한 논리적인 순서를 갖는 구조이다. 스택과 큐, 덱은 자료의 삽입·삭제 위치에 대해 제한 조건이 있는 선형 구조이다.

■ 비선형 구조

자료 간에 선형 구조가 아닌 계층 구조나 망 구조를 갖는 자료구조로 트리와 그래프가 있다.

■ 파일 구조

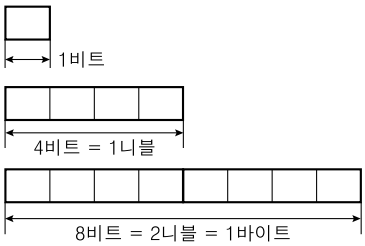
서로 관련 있는 필드들로 구성된 레코드의 집합인 파일에 대한 자료구조로, 보조 기억 장치에 데이터가 실제로 기록되는 자료구조이다. 파일의 구성 방식에 따라 순차 파일과 색인 파일, 직접 파일 등이 있다.

Section

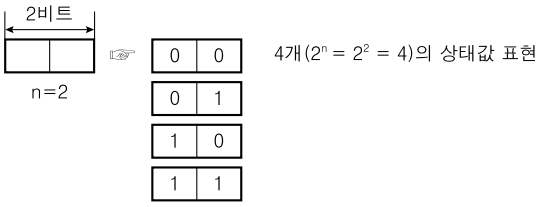
03 자료의 표현

컴퓨터에서는 자료를 표현하기 위해서 1과 0(ON과 OFF, 참(True)과 거짓(False))의 조합으로 구성된 2진수 코드를 사용한다. 숫자·문자·그림·소리 등 다양한 형식의 자료가 컴퓨터 내부에서는 오직 1과 0의 2진수 코드 형태로 표현되어 처리 및 저장된다.

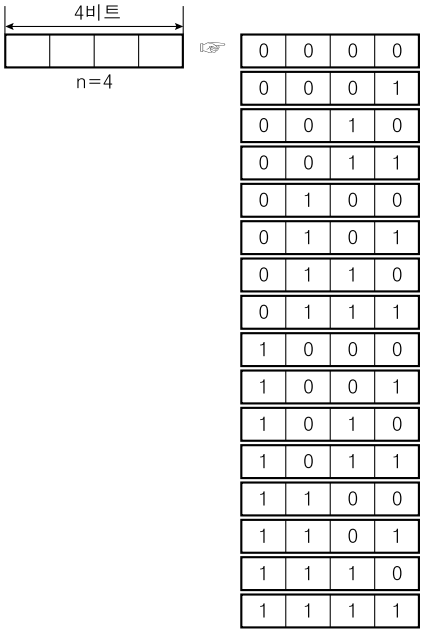
한 자리의 1 또는 0을 표현하는 단위를 비트(Bit)라고 하며, 이는 디지털 시스템에서 자료를 표현하는 최소 단위이다. 4개의 비트 그룹을 니블(Nibble), 8개의 비트 그룹을 바이트(Byte)라고 한다. n개의 비트로 2^n개의 상태를 표현할 수 있다.



[그림 1-5] 비트, 니블과 바이트



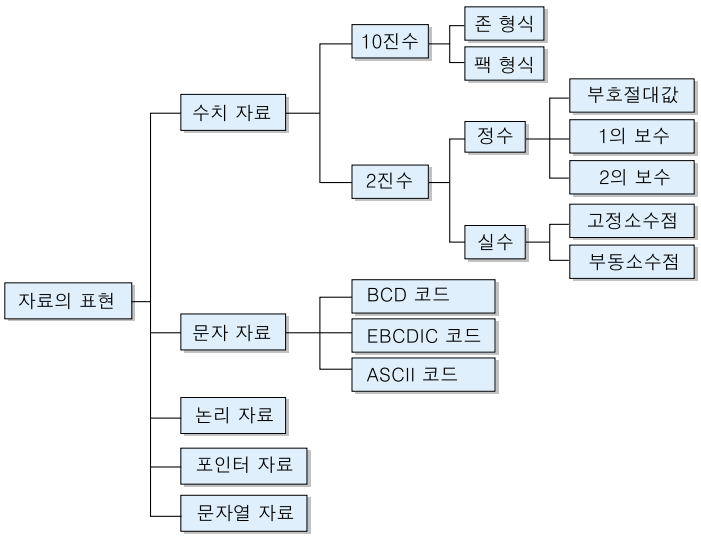
[그림 1-6] n개의 비트로 2^n개의 상태 표현(n=2인 경우)



0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

[그림 1-7] n개의 비트로 2^n개의 상태 표현(n=4인 경우)

컴퓨터 내부에서 표현할 수 있는 자료는 [그림 1-8]과 같으며, 1과 0의 2진수 조합으로 표현된다.



[그림 1-8] 컴퓨터 내부에서 자료를 표현하는 방법

1. 수치 자료의 표현

10진수의 표현

■ 10진수의 존(Zone) 형식 표현

10진수 한 자리를 표현하기 위해서 1바이트(8비트)를 사용하는데, 상위 4비트의 존 영역과 하위 4비트의 수치 영역으로 구성된다.

존 영역				수치 영역			
				8	4	2	1
x	x	x	x	x	x	x	x

[그림 1-9] 존 형식의 구조

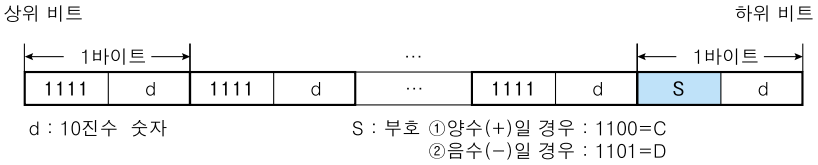
존 영역은 항상 '1111'로 표시한다. 수치 영역은 표현할 10진수 한 자리의 값에 대한 2진수 값을 표시한다. 10진수 값을 2진수로 변환하기 위해서 4비트의 수치 영역은 각 비트 자리마다 2진수의 자리값을 갖는다. 최상위 비트는 2^3=8, 그 다음 비트는 2^2=4, 그 다음 비트는 2^1=2, 마지막 최하위 비트는 2^0=1의 자리값을 갖는다. 4비트의 2진수로 10진수 0~15를 표현할 수 있는데 10에서 15까지의 두 자릿수 10진수는 16진수에서 사용하는 A에서 F까지의 문자를 사용하여 한 자리로 간략히 표현한다.

[표 1-1] 4비트의 2진수에 대한 10진수 표현

4비트의 2진수				10진수 변환	10진수
0	0	0	0	0 × 8 + 0 × 4 + 0 × 2 + 0 × 1	0
0	0	0	1	0 × 8 + 0 × 4 + 0 × 2 + 1 × 1	1
0	0	1	0	0 × 8 + 0 × 4 + 1 × 2 + 0 × 1	2
0	0	1	1	0 × 8 + 0 × 4 + 1 × 2 + 1 × 1	3
0	1	0	0	0 × 8 + 1 × 4 + 0 × 2 + 0 × 1	4
0	1	0	1	0 × 8 + 1 × 4 + 0 × 2 + 1 × 1	5
0	1	1	0	0 × 8 + 1 × 4 + 1 × 2 + 0 × 1	6
0	1	1	1	0 × 8 + 1 × 4 + 1 × 2 + 1 × 1	7
1	0	0	0	1 × 8 + 0 × 4 + 0 × 2 + 0 × 1	8
1	0	0	1	1 × 8 + 0 × 4 + 0 × 2 + 1 × 1	9
1	0	1	0	1 × 8 + 0 × 4 + 1 × 2 + 0 × 1	10 = A

4비트의 2진수				10진수 변환	10진수
1	0	1	1	1 × 8 + 0 × 4 + 1 × 2 + 1 × 1	11 = B
1	1	0	0	1 × 8 + 1 × 4 + 0 × 2 + 0 × 1	12 = C
1	1	0	1	1 × 8 + 1 × 4 + 0 × 2 + 1 × 1	13 = D
1	1	1	0	1 × 8 + 1 × 4 + 1 × 2 + 0 × 1	14 = E
1	1	1	1	1 × 8 + 1 × 4 + 1 × 2 + 1 × 1	15 = F

여러 자리의 10진수를 표현할 경우에는 10진수의 자릿수만큼 존 형식을 연결해서 사용하고, 부호는 최하위 바이트의 존 영역([그림 1-10]의 'S' 영역)에 나타낸다.



[그림 1-10] 존 형식의 10진수 표현 형식

+213과 -213을 존 형식으로 나타내보자.

① +213

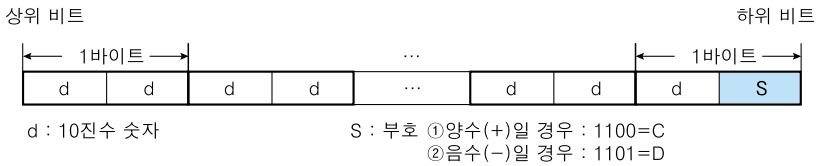
1111	0010	1111	0001	1100	0011
F	2	F	1	C(+)	3

② -213

1111	0010	1111	0001	1101	0011
F	2	F	1	D(-)	3

■10진수의 팩(Pack) 형식 표현

숫자만 사용하는 10진수 연산에서 존 형식을 사용하면 최하위 바이트에서 부호를 표시하는 존 영역 이외의 다른 존 영역은 항상 '1111'을 저장하므로 기억 공간이 낭비되고 처리가 지연된다. 이러한 문제를 해결하기 위해 팩(Pack) 형식에서는 1바이트에 10진수 2자리를 표현한다. 그리고 최하위 바이트의 하위 4비트([그림 1-11]의 'S' 영역)에 부호를 표시한다.



[그림 1-11] 팩 형식의 10진수 표현 형식

+213과 −213을 팩 형식으로 나타내보자.

① +213

0010	0001	0011	1100
2	1	3	C(+)

② −213

0010	0001	0011	1101
2	1	3	D(-)

2진수의 정수 표현

2진수는 일정한 길이의 n비트로 표현하는데, 최상위 비트(MSB: Most Significant Bit)인 첫 번째 1비트는 부호를 나타내기 위해 사용하고 나머지 (n−1)비트는 2진수 값을 표현하기 위해 사용한다.

■ 부호절대값 형식 표현

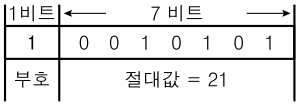
최상위 비트에 부호를 표시하고 나머지 비트에 표현할 2진수의 절대값을 표시한다. 부호가 양수(+)인 경우에는 최상위 비트를 0으로 하고, 음수(−)인 경우에는 최상위 비트를 1로 한다. 1바이트를 사용하여 2진수를 표현한다면 첫 번째 1비트는 부호비트가 되고, 나머지 7비트는 2진수의 절대값을 표현한다.

+21과 −21을 각각 부호절대값 형식으로 표현해보자.

① +21

1비트	7 비트
0	0 0 1 0 1 0 1
부호	절대값 = 21

② −21



■ 1의 보수 형식 표현

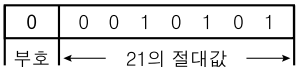
양수(+)는 부호절대값 형식으로 표현하고, 음수(−)는 2진수를 1의 보수(1's Complement) 형식으로 변환하여 표현한다. 1바이트를 사용해서 2진수를 표현하는 경우에 1의 보수를 만드는 방법은 전체 8비트를 1로 변환한 값에서 절대값을 뺀다.

−21을 1의 보수 형식으로 표현하면 다음과 같다.

```
  1 1 1 1 1 1 1 1
- 0 0 0 1 0 1 0 1   ☜ −21의 절대값
-------------------
  1 1 1 0 1 0 1 0   ☜ 21의 1의 보수 = −21
```

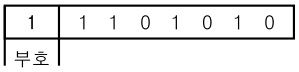
+21과 −21을 1의 보수 형식으로 표현해보자.

① +21



② −21

```
  1   1 1 0 1 0 1 0   ☜ 21의 1의 보수
```



■ 2의 보수 형식 표현

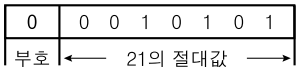
양수(+)는 부호절대값 형식과 같이 표현하고, 음수(−)는 2진수를 2의 보수(2's Complement)로 변환하여 표현한다. 1의 보수에 1을 더하면 2의 보수가 된다.

−21을 2의 보수 형식으로 표현하면 다음과 같다.

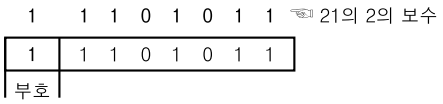
```
  1 1 1 1 1 1 1 1
- 0 0 0 1 0 1 0 1   ☜ −21의 절대값
-------------------
  1 1 1 0 1 0 1 0   ☜ 21의 1의 보수
+               1
-------------------
  1 1 1 0 1 0 1 1   ☜ 21의 2의 보수 = −21
```

+21과 −21을 2의 보수 형식으로 표현해보자.

① +21



② −21



2진수의 3가지 표현 방법인 '부호절대값 형식 표현'과 '1의 보수 형식 표현', '2의 보수 형식 표현'에서 양수(+)에 대한 표현 방법은 모두 같다는 것을 알 수 있다. 음수(−)를 표현하는 방법에 있어서 (n−1)비트의 절대값으로 표현하고 최상위 비트에 음수 부호를 나타내는 1을 표시하느냐, n비트의 1의 보수로 표현하느냐, n비트의 2의 보수로 표현하느냐의 차이만 있다.

- **부호절대값 형식**
  음수(−) 표현은 부호비트만 다르게 하여 사람이 사용하는 음수 형태로 표현해서 편리해 보이지만, 컴퓨터의 입장에서 보면 부호와 절대값을 따로 구분해서 처리해야 한다. 그리고 덧셈기와 뺄셈기를 따로 구현해야 하기 때문에 회로가 복잡해져서 실제로는 사용하지 않는다.
- **1의 보수 형식**
  부호와 절대값을 따로 계산하지 않아도 되고 음수 간의 덧셈으로 뺄셈을 처리할 수 있어 뺄셈기를 구현할 필요가 없다. 따라서 부호절대값 형식보다 회로가 간단해지지만 +0(00000000)과 −0(11111111)이 존재하는 논리적 오류가 발생한다.
- **2의 보수 형식**
  −0을 제거하여 덧셈 연산에 대한 처리를 더 단순화시킨다. 따라서 2의 보수 형식이 음수를 표현하는 방법 중에서 가장 단순하고 효율적인 회로를 구성할 수 있어 컴퓨터 시스템에서 실제로 사용하는 표현 형식이다.

2진수의 실수 표현

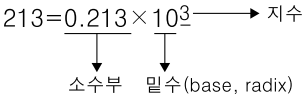
실수와 정수와의 차이점은 소수점의 유무이다. 2진수 실수의 표현 방법은 소수점의 위치가 고정되어 있는 고정소수점 표현 방식과 소수점의 위치가 고정되어 있지 않고 변하는 부동소수점 표현 방식이 있다.

■ **고정소수점 표현 방식**

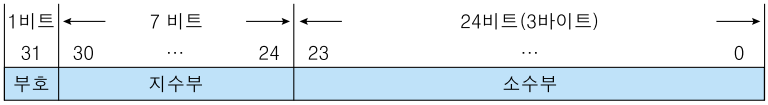
고정소수점 표현 방식은 소수점이 항상 최상위 비트의 왼쪽 밖에 고정되어 있는 것으로 취급하는 것으로 고정소수점 방식으로 표현된 00010101은 0.00010101의 실수를 의미한다.

■ **부동소수점 표현 방식**

부동소수점 표현 방식은 고정소수점 표현 방식에 비해 표현 가능한 값의 범위가 크고, 아주 작은 값이나 아주 큰 값을 표현할 수 있다. 부동소수점 표현 방식은 부호와 지수, 소수의 3부분으로 구분된다.



일반적으로 컴퓨터에서 실수 자료형은 4바이트(32비트)나 8바이트(64비트)로 표현한다. 4바이트를 사용한 부동소수점의 표현 형식을 보면 [그림 1-12]와 같다.



[그림 1-12] **4바이트 부동소수점 표현 형식**

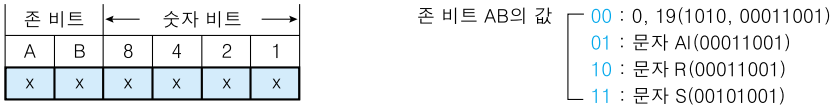
부호비트는 양수(+)인 경우 0, 음수(-)인 경우 1로 표시한다.

2 문자 자료의 표현

컴퓨터 내부에서는 문자 자료도 1과 0의 2진수 조합으로 표현한다. 문자에 대한 2진 코드를 정의해놓은 문자 코드들이 있는데, 그중에서 주로 사용하는 BCD 코드와 EBCDIC 코드, ASCII 코드를 알아보자.

BCD 코드

BCD(Binary-Coded Decimal) 코드는 6비트를 사용하며, 상위 2비트의 존 비트와 하위 4비트의 숫자 비트로 구성된다.



[그림 1-13] BCD 코드 구성

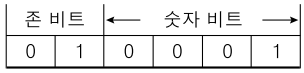
다음 BCD 코드표에서 문자는 존 비트와 숫자 비트의 조합으로 0부터 9까지의 10진수 숫자와 영어 대문자와 특수 문자를 표현할 수 있다.

[표 1-2] BCD 코드표

존 비트		숫자 비트				표현 문자	존 비트		숫자 비트				표현 문자	존 비트		숫자 비트				표현 문자	존 비트		숫자 비트				표현 문자
0	0	0	0	0	1	1	0	1	0	0	0	1	A	1	0	0	0	0	1	J							
0	0	0	0	1	0	2	0	1	0	0	1	0	B	1	0	0	0	1	0	K	1	1	0	0	1	0	S
0	0	0	0	1	1	3	0	1	0	0	1	1	C	1	0	0	0	1	1	L	1	1	0	0	1	1	T
0	0	0	1	0	0	4	0	1	0	1	0	0	D	1	0	0	1	0	0	M	1	1	0	1	0	0	U
0	0	0	1	0	1	5	0	1	0	1	0	1	E	1	0	0	1	0	1	N	1	1	0	1	0	1	V
0	0	0	1	1	0	6	0	1	0	1	1	0	F	1	0	0	1	1	0	O	1	1	0	1	1	0	W
0	0	0	1	1	1	7	0	1	0	1	1	1	G	1	0	0	1	1	1	P	1	1	0	1	1	1	X
0	0	1	0	0	0	8	0	1	1	0	0	0	H	1	0	1	0	0	0	Q	1	1	1	0	0	0	Y
0	0	1	0	0	1	9	0	1	1	0	0	1	I	1	0	1	0	0	1	R	1	1	1	0	0	1	Z
0	0	1	0	1	0	0																					

코드표에서 영문자 A에 대한 BCD 코드를 찾아보자. A가 있는 자리의 존 비트 '01'과 숫자 비트 '0001'을 연결한 '01 0001'이 A에 대한 BCD 코드가 된다.

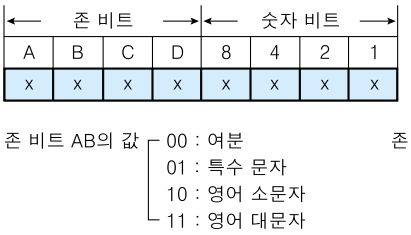
A = 01 0001



EBCDIC 코드

EBCDIC 코드(Extended Binary-Coded Decimal Interchange Code)는 IBM사에서 개발한 코드로 8비트를 사용하며 상위 4비트의 존 비트와 하위 4비트의 숫자 비트로 구성된다.

존 비트와 숫자 비트의 조합으로 0부터 9까지의 10진수 숫자와 영어 대문자와 소문자, 특수 문자까지 나타낼 수 있다.



[그림 1-14] EBCDIC 코드 구성

다음 EBCDIC 코드표에서 열은 상위 4비트의 존 비트를 나타내고, 행은 하위 4비트의 숫자 비트를 나타낸다. 열과 행의 비트를 연결하면 8비트의 EBCDIC 코드를 구할 수 있다

[표 1-3] EBCDIC 코드표

상위 / 하위	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	NUL	DLE	DS		SP	&	-						{	}	₩(\)	0
0001	SOH	DC1	SOS				/		a	j	~		A	J		1
0010	STX	DC2	FS	SYN					b	k	s		B	K	S	2
0011	ETX	TM							c	l	t		C	L	T	3
0100	PF	RES	BYP	PN					d	m	u		D	M	U	4
0101	HT	NL	LF	RS					e	n	v		E	N	V	5
0110	LC	BS	ETB	UC					f	o	w		F	O	W	6
0111	DEL	IL	ESC	EOT					g	p	x		G	P	X	7
1000	GE	CAN							h	q	y		H	Q	Y	8
1001	RLF	EM							i	r	z		I	R	Z	9
1010	SMM	CC	SM		¢	!		:								
1011	VT	CU1	CU2	CU3	.	$	,	#								
1100	FF	IFS		DC4	<	*	%	@								
1101	CR	IGS	ENQ	NAK	(	)	_	'								
1110	SO	IRS	ACK		+	;	>	=								
1111	SI	IUS	BEL	SUB	|	¬	?	"								

※ 코드의 의미

ACK	Acknowledge	EOT	End of Transmission	PN	Punch On
BEL	Bell	ESC	Escape	RES	Restore
BS	Backspace	ETB	End of Transmission Block	RS	Reader Stop
BYP	Bypass	ETX	End of Text	SI	Shift In
CAN	Cancel	FF	From Feed	SM	Set Mode
CC	Cursor Control	FS	Field Separator	SMM	Start of Manual Message
CR	Carriage Return	HT	Horizontal Tab	SO	Shift Out
CU1	Customer Use 1	IFS	Interchange File Separator	SOH	Start of Heading
CU2	Customer Use 2	IGS	Interchange Group Separator	SOS	Start of Significance
CU3	Customer Use 3	IL	Idle	SP	Space
DC1	Device Control1	IRS	Interchange Record Separator	STX	Start of Text
DC2	Device Control2	IUS	Interchange Unit Separator	SUB	Substitute
DC4	Device Control4	LC	Lower Case	SYN	Synchronous
DEL	Delete	LF	Line Feed	TM	Tape Mark
DLE	Data Link Escape	NAK	Negative Acknowledge	UC	Upper Case
DS	Digital Select	NL	New Line	VT	Vertical Tab
EM	End of Medium	NUL	Null	¢	Cent Sign
ENQ	Enquire	PF	Punch off	¬	Logical NOT

코드표에서 영문자 A에 대한 EBCDIC 코드를 찾아보자. A가 있는 자리의 열이 '1100'이고, 행이 '0001'이므로 이를 연결한 '1100 0001'이 A에 대한 EBCDIC 코드가 된다.

A = 11000001

← 존 비트 →				← 숫자 비트 →			
A	B	C	D	하위 비트			
1	1	0	0	0	0	0	1

ASCII 코드

ASCII 코드(American Standard Code for Information Interchange)는 7비트를 사용하며 상위 3비트의 존 비트와 하위 4비트의 숫자 비트로 구성된다. 존 비트와 숫자 비트의 조합으로 0부터 9까지의 10진수 숫자와 영어 대문자와 소문자, 특수 문자를 나타낼 수 있다.

← 존 비트 →			← 숫자 비트 →			
			8	4	2	1
x	x	x	x	x	x	x

[그림 1-15] ASCII 코드 구성

다음 ASCII 코드표에서 열은 상위 3비트의 존 비트를 나타내고, 행은 하위 4비트의 숫자 비트를 나타내므로 열과 행의 비트를 연결하면 7비트의 ASCII 코드를 구할 수 있다. EBCDIC 코드와 유사하지만, 일반적으로 ASCII 코드를 더 많이 사용한다. 데이터 통신용으로 ASCII 코드를 사용할 경우에 패리티 비트를 최상위 비트로 추가하여 8비트 형식으로 사용하기도 한다.

[표 1-4] ASCII 코드표

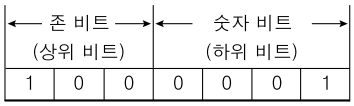
하위 \ 상위	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	″	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	END	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	′	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	₩ (\)	l	|
1101	CR	GS	−	=	M	]	m	}
1110	SO	RS	·	>	N	∧	n	~
1111	SI	US	/	?	O	_	o	DEL

※ 코드의 의미

GS	Group Separator	RS	Record Separator	US	Unit Separator

코드표를 보고 영문자 A에 대한 ASCII 코드를 찾아보자. A가 있는 자리의 열이 '100' 이고, 행이 '0001' 이므로 이를 연결한 '100 0001' 이 A에 대한 ASCII 코드가 된다.

A = 1000001



초기 IBM 컴퓨터 시스템에서는 BCD 코드를 사용하다가 더 많은 문자 코드를 표현할 수 있는 EBCDIC 코드로 대체되었고, 현재는 미국 표준 코드인 ASCII 코드를 일반적으로 사용하고 있다. BCD 코드는 세븐 세그먼트를 이용하여 숫자를 10진 코드로 출력해야 하는 전자회로와 마이크로 프로세서에서 사용하고 있다.

3. 논리 자료의 표현

논리 자료는 논리값을 표현하기 위한 자료 형식이다. 논리값이란 참(True)과 거짓(False)의 두 가지 상태 중에서 하나를 표시하는 것으로 1비트로 표현이 가능하지만 일반적으로 컴퓨터 내부에서는 1바이트나 1워드(word)를 논리값 표현 단위로 사용한다.

1바이트를 사용하여 논리 자료를 표현하는 방법은 다음의 3가지가 있다.

방법 1) 참인 경우에는 최하위 비트를 1로 표시하여 00000001로 표현하고,
거짓인 경우에는 00000000으로 표현한다.
방법 2) 참인 경우에는 전체 비트를 1로 표시하여 11111111로 표현하고,
거짓인 경우에는 00000000으로 표현한다.
방법 3) 참인 경우에는 하나 이상의 비트를 1로 표시하고,
거짓인 경우에는 00000000으로 표현한다.

4. 포인터 자료의 표현

포인터(Pointer) 자료는 메모리의 주소를 표현하기 위한 자료 형식으로, 자료를 저장하고 있는 변수나 특정 위치에 대한 메모리 주소를 저장하고 주소 연산에 사용한다.

포인터 자료를 사용하면 복잡한 자료구조 연산을 메모리에서의 주소 연산만으로 처리할 수 있다.

5. 문자열 자료의 표현

문자열(String) 자료는 하나의 문자만 표현할 수 있는 문자 자료와 다르게 여러 문자로 이루어진 문자 그룹을 하나의 자료로 취급하여 메모리에 연속적으로 저장하는 자료 형식이다. 하나의 문자열 자료는 여러 개의 부분문자열을 포함할 수 있다.

부분문자열을 포함하는 문자열 자료를 메모리에 저장하는 방법은 다음의 3가지가 있다.

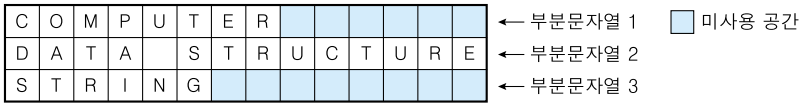
방법 1) 부분문자열 사이에 구분자를 사용해서 저장하는 방법
방법 2) 가장 긴 부분문자열의 길이에 맞추어 고정 길이로 저장하는 방법
방법 3) 부분문자열을 연속 저장하고 각 부분문자열에 대한 포인터를 사용하는 방법

{COMPUTER, DATA STRUCTURE, STRING}과 같이 3개의 부분문자열로 구성된 문자열 자료구조를 메모리에 저장하는 방법을 살펴보자.

구분자를 사용하는 방법은 표현하기는 간단하지만, 필요한 부분문자열을 찾기 위해서 문자를 비교하여 구분자를 식별하는 연산과 처리 시간이 필요하다. 다음은 세미콜론(;)을 구분자로 사용하여 메모리에 저장하는 방법이다.

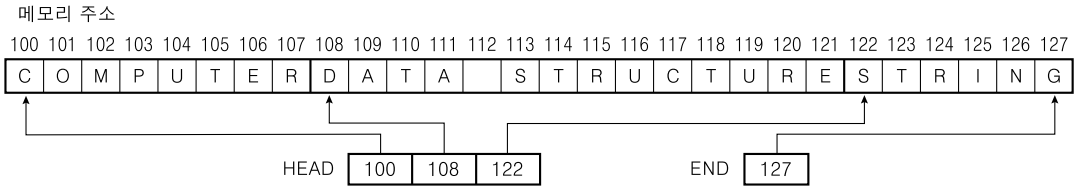
[그림 1-16] 구분자를 사용하여 문자열 저장하는 방법(방법 1)

고정 길이 저장 방법은 부분문자열 중 가장 긴 부분문자열의 길이를 고정 길이로 이용한다. 부분문자열마다 저장되는 고정 길이가 같기 때문에 부분문자열을 찾기는 쉬우나 부분문자열 간의 길이 차이가 클 경우에는 미사용 공간이 많아진다.



[그림 1-17] 고정 길이로 문자열 저장하는 방법(방법 2)

포인터를 사용하는 방법은 부분문자열에 대한 주소와 문자열의 마지막 주소를 포인터에 저장하여 포인터 주소 연산으로 부분문자열을 찾는다.



[그림 1-18] 포인터를 사용하여 문자열 저장하는 방법(방법 3)

[표 1-5]는 문자열을 표현하는 3가지 방법에 대해 메모리 이용률과 부분문자열 탐색 시간을 비교한 자료이다.

[표 1-5] 문자열 표현 방법 비교

비교 항목 \ 방법	구분자를 사용하는 방법	고정 길이로 저장하는 방법	포인터를 사용하는 방법
메모리 이용률	문자열 길이 + 구분자 길이 ☞ 효율적	가장 긴 부분문자열 길이 × 부분문자열의 개수 ☞ 비효율적	문자열 길이 + 포인터 저장 공간 ☞ 효율적
부분문자열 탐색 시간	문자 비교 연산 시간 + 구분자 식별 시간 ☞ 비효율적	문자 비교 연산 시간 ☞ 효율적	문자 비교 연산 시간 + 포인터 주소 연산 시간 ☞ 효율적

요약

1 자료구조는 다양한 자료를 효율적으로 표현해 저장하고 처리하여 사용할 수 있도록 하는 것이다. 컴퓨터에서 자료를 효과적으로 표현하고 표현한 자료를 좀 더 효율적으로 저장, 처리할 수 있도록 논리적인 구조로 만들어 프로그램적으로 처리하는 것이 자료구조이다.

2 자료를 형태에 따라 분류해보면 프로그래밍 언어에서 배운 정수, 실수, 문자, 문자열 등의 데이터 타입에 해당하는 단순 구조와 자료 간에 1:1의 관계가 있는 선형 구조, 1:다 또는 다:다의 관계를 갖는 비선형 구조, 그리고 파일 구조가 있다.

3 표현할 자료의 특성과 양, 자료의 주된 사용 방법과 수행하는 연산의 종류, 구현에 필요한 기억 공간 용량을 고려하여 가장 효율적인 자료구조를 선택해야 한다.

4 컴퓨터에서는 자료를 표현하기 위해서 1과 0(ON과 OFF, 참(True)과 거짓(False))의 조합으로 구성한 2진수 코드를 사용한다. 숫자·문자·그림·소리 등의 다양한 형식의 자료가 컴퓨터 내부에서는 오직 1과 0의 2진수 코드 형태로 표현되어 처리되고 저장된다.

5 10진수의 표현 방법은 존(Zone) 형식과 팩(Pack) 형식이 있다.

6 2진수 정수의 표현 방법은 부호절대값 형식 표현과 1의 보수 형식 표현, 2의 보수 형식 표현이 있다. 양수(+)에 대한 표현은 3가지 방법이 모두 같고, 음수(−)에 대한 표현만 차이가 있다.

7 2진수 실수의 표현 방법은 고정소수점 표현 방식과 부동소수점 표현 방식이 있다. 부동소수점 표현 방식은 표현 가능한 값의 범위가 고정소수점 표현 방식보다 크고, 아주 작은 값이나 아주 큰 값을 표현할 수 있다.

8 컴퓨터 내부에서는 문자 자료 역시 1과 0의 2진수 조합으로 표현한다. 문자에 대한 2진 코드를 정의해놓은 문자 코드들이 있는데, 그중에서 주로 사용하는 코드는 BCD 코드와 EBCDIC 코드, ASCII 코드가 있다.

9 논리 자료는 0과 1의 논리값을 표현하기 위한 자료 형식이다.

요약

10 포인터 자료는 메모리의 주소를 표현하기 위한 자료 형식으로서 메모리의 주소를 저장하고 주소 연산에 사용한다.

11 문자열 자료는 여러 문자로 이루어진 문자 그룹을 하나의 자료로 취급하여 메모리에 연속적으로 저장하는 자료 형식이다. 문자열에 포함된 부분문자열을 표현하는 방법으로 구분자를 사용하는 방법, 고정 길이를 정하여 사용하는 방법, 포인터를 사용하는 방법이 있다.

연습문제

1 비선형 구조와 선형 구조가 옳게 짝지어진 것은? (2007년 기출문제)

① 스택(Stack)
② 큐(Queue)
③ 트리(Tree)
④ 연결 리스트(Linked List)
⑤ 그래프(Graph)

가. 비선형 자료구조 : ①,②,⑤
선형 자료구조 : ③,④

나. 비선형 자료구조 : ③,⑤
선형 자료구조 : ①,②,④

다. 비선형 자료구조 : ①,②,③
선형 자료구조 : ④,⑤

라. 비선형 자료구조 : ③
선형 자료구조 : ①,②,④,⑤

2 다음 중 선형 자료구조가 아닌 것은 무엇인가? (2003년, 2004년, 2006년 기출문제)

가. 리스트　　나. 그래프　　다. 스택　　라. 큐

3 컴퓨터에서 정보를 표현할 수 있는 최소 단위는 무엇인가?

4 6개의 비트를 가지고 서로 다른 상태값을 표현할 때 최대 몇 개의 값을 표현할 수 있는가?

5 $(1001)_2$을 10진수로 변환하면 얼마인가?

6 8비트 부호절대값 형식으로 +62와 −62를 표현하시오.

7 8비트 1의 보수 형식으로 +62와 −62를 표현하시오.

8 10진수 516을 존 형식과 팩 형식으로 각각 표현하시오.

9 다음의 정수를 표현하는 방법 중에서 같은 크기의 비트 수를 사용할 때 표현 범위가 가장 큰 것은 무엇인가?

가. 2의 보수 방법　　나. 부호절대값 방법　　다. 팩 형식　　라. 존 형식

10 2의 보수 표현 방법에서 8비트의 기억 공간에 정수를 표현할 때 표현 범위는? (2006년 기출문제)

가. $-2^7 \sim +2^7$
나. $-2^8 \sim +2^8$
다. $-2^7 \sim +2^7 - 1$
라. $-2^8 \sim +2^8 - 1$

연습문제

11 2의 보수 표현이 1의 보수 표현보다 더 널리 사용되고 있는 주요 이유는? (2006년, 2010년 기출문제)

가. 음수 표현이 가능하다.
나. 10진수 변환이 더 용이하다.
다. 보수 변환이 더 편리하다.
라. 표현할 수 있는 수의 개수가 하나 더 많다.

12 2진수 $(1001011)_2$의 2의 보수는? (2003년 기출문제)

가. 0110100 나. 1110100 다. 1110101 라. 0110101

13 10진수 634를 BCD 코드로 표현하였을 때 옳은 것은? (2005년, 2006년 기출문제)

가. 0110 0011 0100
나. 0110 0011 0011
다. 0011 0011 0100
라. 0011 0011 0011

14 EBCDIC 코드를 사용하여 영어 대문자 A~I를 표현할 때 EBCDIC 코드의 존 비트(상위 4비트)의 값은 무엇인가?

15 ASCII 문자에 해당하지 않는 것은? (2006년 기출문제)

가. 제어 문자 나. 영문자 다. 로마 문자 라. 아라비아 문자

16 자료에 관한 설명 중 옳은 것은? (2002년 기출문제)

가. EBCDIC 코드는 데이터 통신용으로 널리 쓰이며, 특히 소형 컴퓨터용으로 쓰인다.
나. ASCII 코드는 IBM사에서 개발한 것으로 대형 컴퓨터용으로 쓰인다.
다. 자료의 가장 작은 단위를 비트(Bit)라 하며, Bit는 Binary Digit의 약자이다.
라. 부동소수점 방식의 특징은 적은 비트(Bit)를 차지함과 동시에 정밀도가 낮다는 것이다.

17 문자 자료와 문자열 자료의 차이를 설명하시오.

18 다음의 자료 형식 중에서 가장 적은 비트 수로 표현할 수 있는 것은 무엇인가?

가. 실수형 자료 나. 정수형 자료 다. 문자형 자료 라. 논리형 자료