Lecture 6 : 다차원 벡터의 구조와 그 활용

다차원 벡터는 실전에서 매우 흔히 사용되는 가장 대표적인 자료구조이다. 먼저 다차원 배열의 물리적인 구조를 이해해야 한다. 우리에게 보이는 다차원 배열의 구성의 전형적인 모습은 **`int x[5][6][7]`**과 같은 형식이다. 이 배열의 차원은 3차원이며 각 차원은 서로 독립적으로 indexing될 수 있다. 그러나 우리가 관리하는 메모리는 단순히 1차원 linear 구조이므로 실제 다차원 구조는 구현될 수 없다. 따라서 다차원 배열은 궁극적으로 1차원 배열에 저장되고 탐색 된다. 이 과정을 잘 이해하는 것이 올바른 자료구조를 구성하는 가장 중요한 요건이 된다.

다차원 배열에서 표현하는 것은 사용자를 위하여 편의를 제공해주는 것이지 실제 이것이 컴파일되어 수행되면 배열의 index를 기초로 실제 배정된 메모리 주소를 계산하게 된다. 이 과정이 컴파일 과정에 수행된다.

크기와 각 원소의 위치가 고정된 다차원 배열과 다르게 다차원 벡터는 각 원소 하나 하나가 벡터가 되고 각 벡터에 대하여 추가 원소의 삽입, 삭제가 가능하므로 다차원 배열에 비해서 훨씬 복잡한 구조를 가지고 있다. 이것은 C 혹은 C++의 pointer를 이용해서 직접 구현하는 것은 상당히 까다로운 작업이며 그 과정에 많은 오류가 발생할 수 있으므로 이런 작업이 필요한 경우에는 반드시 다차원 벡터를 사용해야 한다. 다차원 벡터는 기본 벡터의 원소가 프로그래밍 언어에서 기본적으로 제공하는 atom원소 {int float, double..}이 아닌 STL의 다른 container를 원소로 가지고 있는 구조이다.

이 장에서 여러분이 명심해야 하는 것은 다차원 벡터를 어떻게 사용하는가에 집중하는 것도 있겠지만 어떤 문제에 다차원 벡터를 활용하는 것이 유리하고 왜 그것이 유리한지를, 자료구조에 접근하는 basic function 관점으로 이해하고 있어야 한다는 것이다.

6.1 벡터를 다차원으로 구성하기

```
- vector < int > X ; // 1차원 vector
- vector < vector<string>  Book ;
// vector의 원소가 벡터
```

Book	**vector<string>**
0	There's a lady who's sure
1	All that glitters is gold $$
2	And she's buying a stairway to heaven
3	When she gets there she knows
4	If the stores are all closed

6.2 자신이 설정한 Class나 Type으로 Vector 구현하기

```
- vector < myty > X ;
```

0	1	2	3	4	5	6
myty	myty	myty	myty	myty	myty	myty

6.3 다차원(Multi-dimensional) 배열의 물리적 구성

- 배열의 한 원소가 다른 배열이 경우
- 실제 컴파일 후 메모리 맵을 구성할 때 "행 우선", "열 우선" 구현이 있다.
- `int x[10][8][7]`에서 `*(x+61)`은 어느 위치 원소와 동일한가 ?

- 다차원 배열의 두 가지 형식 a) 규칙적인 구성 b) 불규칙적인 구성

다차원 배열을 실제 쓸 때 주의해야 할 사항

```
int a[100], b[100] ;
for( i=1; i++ ; i < 200)  a[i]= i*i ;  // segment error ?
```

Q) Reference와 Pointer의 차이를 설명하시오

```
int i ;
int &x  = t ;
int *p ;              p = &i ;  *p= 300 ;
```

6.3 다차원(Multi-dimensional) 배열이 사용되어야 하는 경우

- 자료구조 2019년도 출석부
- 2차원 (행렬)
- 3차원 (tensor)
- 전국 기상측정 지점의 매 5분 간격의 동안의 기온
- 3차원 반도체
- 5분 간격의 환율 (달러: 원화, 유로:원화)
- 소설, 전쟁과 평화
- 소설, 노인과 바다
- 약국의 약장
- 양궁 선수 선발전 기록

6.4 Rule of Declaration Precedence in C, C++ (대단히 중요한 규칙)

Modifier	Significance
*	indicates a pointer
( )	indicates a function
[ ]	indicates an array

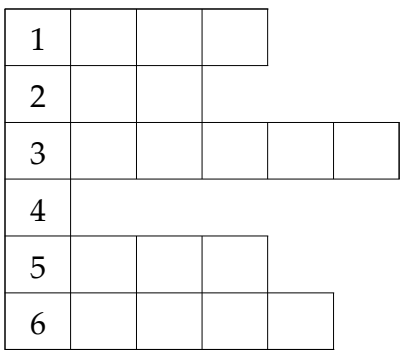
a. 배열 [ ] 이나 함수 ( ) modifiers가 포인터(*) 해석보다 먼저 적용된다.
b. 개체를 하나로 묶은 괄호가 가장 높은 우선순위를 가진다.
c. [ ]와 ( )의 경우에는 가까울수록 먼저 적용됨

6.5 다음을 해석하시오. (그림을 그려서 구조를 밝히시오.)

```
a. char *mybook[3][4], **lotte[3], ***what ;

b. double (*your)[5][6]

c. float *fun( ) ;

d. int (* fun)( ) ;

e. char *func[3]( ) ;

f. struct HOUSE **sign[3][4][4]
```

6.6. 다차원 배열을 REGUALR, IRREGULAR 로 선언할 수 있다. vector를 이용하면.

- int this[10][10] ; - vector < vector >



어떤 경우에 위와 같은 불규칙한 2차원 배열이 필요할까요?

example) 학생별 출석부 (결석한 날짜만 기록)

vector < vector < date > >

- 달 별 연차 사용자

```
typedef pair<int,int> Date;
typedef vector< Date > Student ;
int main() {

  Date x,y;
  Student  dsall ;
  array< Student,100 >  cbook ;    // 100명의 학생의 출력기록

  x = make_pair(9, 11) ;  // 9월 11일 결석
  dsall.push_back(x) ;
  x = make_pair(9, 28) ;   // 9월 28일 결석
  dsall.push_back(x) ;

  cout << "\n Size of cbook= " << sizeof(cbook) ;

  cbook[1] = dsall ;      // 1번 학생의 출력 기록
  cout << "\n Size of cbook= " << sizeof(cbook) ;

} // end of main()
```

6.7 다차원 배열의 응용 (희소행렬의 행렬 덧셈, 곱셈) : M[10000][10000]이 있다. 그런데 이 중에서 실제 값은 대략 1000-3000 여개에 불과하다. 이러한 경우 전체를 "쌩"으로 float M[10000][10000]으로 잡으면 안 된다. 따라서 희소행렬을 위한 좋은 자료구조가 필요하다.

(a) 어떤 응용에서 이런 문제가 실제 나타날까요?

(답) (예를 들어 전국의 10,000 by 10,000 개의 구역으로 나누고 몇 지점에서 측정된 값, 기온 습도 등을 중심으로 다른 지역의 기상상황을 구하고자 할 때 활용, Finite Element Method의 일종)

(b) 여러분이 구성한 자료구조에서 다음 Operation을 위한 작업의 수행시간은 ?

자료구조	M[i][j]=a	M[i][j]	zero?	M+N	M[i][*]	M[*][j]	M*N	Clear
M[ ][ ]								
여러분1								
여러분2								

6.8 다차원 배열을 array 자료구조로 사용하면 좋은 예, 나쁜 예

(a) 화투놀이

(b) 3차원 격자 공간 simulation

(c) 500 여종 휴대폰의 특성 분석표 (Characteristic Table)

(d) 각 가족별 정보

(e) 전국 교회당의 신자 관리

(f) 신용카드 사용자들의 월별, 일별 사용물품 정보관리

6.8 Column별 Row별 scan에 따른 수행 속도 차이 (2018+)

row 먼저 access	column 먼저 access

int myarray[n][m]
for(i=0; i < n ; i++) {
  for(j=0; j < m ; j++) {
  print myarray[i][j] ;
  }
}
	int myarray[n][m]
	for(j=0; j < m ; j++) {
	  for(i=0; i < n ; i++) {
	  print myarray[i][j] ;
	  }
	}

왜 속도 차이가 날까요? 만일 속도 차이가 보고된다면 어느 정도 크기에서 발생할까? (Virtual memory 구조와 Memory Swap을 이해하고 있어야 함.)

1	2	3	4	5	6	7									

1					2					3					4

- 프로그램에서 memory를 access하는 단계
  - user mode에서 system mode로 전환
  - context save
  - OS에서 메모리 확보
  - 다시 사용자 모드로 돌아 옴

6.9 다차원 배열의 한계 측정은 매우 중요한 작업이다. 이것은 compiler dependent하다.

```
int x[N][N][N][N] ;
```

- 확보할 수 메모리의 크기를 각 compiler에서 test를 해야 함.
- automatic 일 경우는 static 일 경우 구분하여 평가해보아야 함.

6.10 배열과 vector를 위한 오버로딩

Operator Overloading for vector add: va = vb + vc ;

```
template <typename T>

// 반드시 그 크기가 같아야 한다.
vector<T> operator+ (vector<T>& a, vector<T>& b) {
    assert(a.size() == b.size());

    vector<T> result;
    result.reserve(a.size());

    auto ita = a.begin();
    auto itb = b.begin();
    for(; ita < a.end(); ita++, itb++)
        result.push_back( *ita + *itb ) ;

    return result ;
}

int main() {

    vector <int> va   { 1, 2, 3,  4,  5} ;
    vector <int> vb   { 7, 8, 9, 10, 11} ;
    vector <int> vc ;

    vector <double> vda   { 1.0, 2.2, 3.4,  0.04,  5.3} ;
    vector <double> vdb   { 0.7, 8.2, 1.3, -3.2 ,  1. } ;
    vector <double> vdc ;

    vc  = va  + vb ;        // template의 장점
    vda = vda + vdb ;

    vec_out("vc", vc ) ;
    vec_out("vda", vda ) ;
    return 0;
}  // end of main( )
```

6.11 **Vector를 이용한 실전 연습**

문자와 단어로 구성된 Text(소설) 파일 T가 있다. 이 Text는 다음과 같은 계층 구조로 (hierarchical) 이루어져 있다. 이것은 나중에 배울 Tree 구조와 같다.

처리할 자료를 아래와 같은 형식적(formal)한 방법으로 표현하는 방법을 잘 익혀두어야 한다. 좋은 표현(good representation)은 좋은, 안정적인 코드를 만들기 위하여 필요한 요소이다. 따라서 문제가 주어지면 바로 코딩을[1] 하기 전에 반드시 아래와 같이 형식화 (formalization) 작업을 해야 한다. 이 과정에 수식화는 반드시 필요한 단계이다.

$T=< S_i >$ // 소설은 문장 s_i의 sequence(서열)로 이루어져 있다.

$S_i =< w_{i,j} >$ // 문장 s_i는 단어 $w_{i,j}$의 순서(sequence)로 구성되어 있다.

$w_{i,j} = < c_{i,j}^{(k)} >$ // 각 단어는 단위 문자의 순서로 이루어진 문자열이이다.

```
Taking a new step, uttering a new word is what they fear most.  #
But I am talking too much.  #
It's because I chatter that I do nothing.  #
Or perhaps it is that I chatter
It is not serious at all.  #
It's simply a fantasy to amuse myself; a plaything!   #
Yes, maybe it is a plaything.   #
because I do nothing.  #
I've learned to chatter this last month, lying for days together
in my den thinking ... of Jack the Giant-killer.  #
Why am I going there now?   #
Am I capable of that?   #
Is that serious?   #
```

지표	해당 데이터	지표	해당 데이터
S_3		S_6	
$w_{5,3}$		$w_{6,1}$	
$w_{5,22}$		$c_{3,4}^{(3)}$	
$c_{7,4}^{(8)}$		$\text{leng}(S_6)$	
$\text{leng}(w_{5,3})$		$S_2 < S_5$	

1) "코딩을 먼저 시작하는 사람이 가능 늦게 작업을 마친다." 일거리를 받자마자 바로 시작하면 안된다. 어떤 시험이든지 평가든지, 받자마자 바로 풀지말고 전체를 찬찬히 살펴보는 것이 매우 중요하다.