컴퓨터프로그래밍 및 실습

C++ 문자열/벡터

문자열 자료형(class) string – C++

- string 타입을 사용할 경우 헤더 파일 포함: #include<string>
- 문자열 상수는 이중인용부호로 둘러 쌓여 있다.
- 선언, 지정, 입출력에서 숫자형과 비슷하게 사용된다:

```
string name = "John"; // 선언과 초기화
name = "Carl"; // 지정문
cout << "Please enter your name:"; // 출력
cin >> name; // 입력(첫 번째 공백에서 중지함)
```

- Enter 키까지 입력되는 모든 문자를 읽으려면 getline을 사용

```
getline(cin, name);
```

–입력 문자들 예

Harry Hacker

string 멤버함수 – C++

- string 멤버함수

이름	목적
s.length()	문자열 s의 길이를 반환하는 함수
s.insert(pos, s2)	s의 pos 위치에 문자열 s2를 삽입
s.remove(pos, len)	s의 pos 위치부터 len 개 문자를 삭제
s.find(s2)	s에서 문자열 s2가 발견되는 첫번째 위치(인덱스)를 반환
s.find(pos, s2)	s의 pos위치부터 문자열 s2가 발견되는 첫번째 위치(인덱스)를 반환
s.substr(i, n)	문자열 s의 인덱스 i에서 시작하는 길이 n의 부분문자열을 반환
s.c_str()	s에서 문자열을 저장하는 배열의 주소를 반환
getline(f, s)	입력스트림 f로부터 string s를 입력

string 멤버함수 호출 – C++

- *string 변수이름.멤버함수이름*(인자$_1$,인자$_2$, ..., 인자$_n$)

- 멤버함수 호출 예

  name.length()

  name.substr(0, n – 1)

string 부분 문자열 추출 멤버함수 subsrt – C++

- 부분 문자열 추출 멤버 함수
  - substr(start, length)
- 예

```
string greeting = "Hello, World!\n";·
string sub = greeting.substr(0,4);
/* sub is "Hell" */
```

- 문자의 시작 위치는 0, 마지막 문자 위치는 length - 1

H	e	l	l	o	,		W	o	r	l	d	!	₩n
0	1	2	3	4	5	6	7	8	9	10	11	12	13

- string w = greeting.substr(7, 5);

H	e	l	l	o	,		W	o	r	l	d	!	₩n
0	1	2	3	4	5	6	7	8	9	10	11	12	13

string 문자열 접합(concatenation) – C++

- + 연산자 : 두 문자의 접합
- 예

```
string fname = "Harry";·
string lname = "Hacker";
string name = fname + " " + lname;
```

```
#include <iostream>
#include <string>
using namespace std;
int main()
{   cout << "Please enter your full name (first middle last): ";
    string first;
    string middle;
    string last;
    cin >> first >> middle >> last;
    string name1 = first.substr(0, 1) + "." + middle.substr(0, 1)+"." + last;
    cout << "Your name is " << name1 << "\n";
    return 0;
}
```

string 문자열 비교 – C++

- 두 문자열 비교 연산자: ==, < , > 연산자
- string name1 = “Kim";·
  string name2 = “Lee";

  if (name1 > name2)
  …

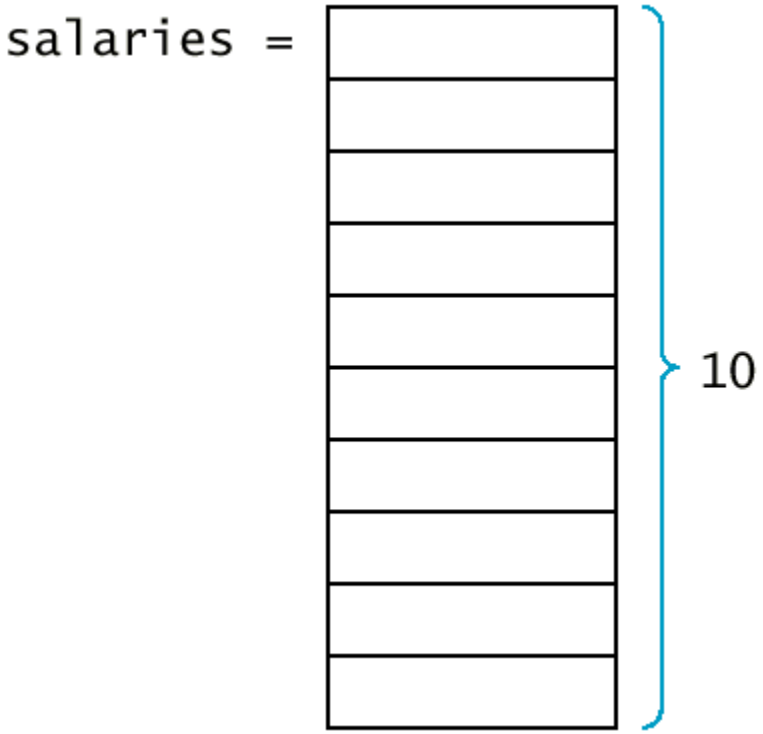
벡터 이용 예

- 급여 리스트를 읽어 최대 값을 출력하는 프로그램을 작성하라.

32000 54000 67500 29000 35000 80000 115000 44500 100000 65000

벡터

- 벡터는 같은 자료형의 원소들 모임이다
- vector<double> salaries(10);
- 위 벡터는 10개의 값을 저장하며, 각 값은 double 자료형이다.



벡터

- string 타입을 사용할 경우 헤더 파일 포함:

  #include <vector>

- 벡터 변수 정의

vector<*type_name*> *variable_name*;
vector<*type_name*> *variable_name*(*initial_size*);

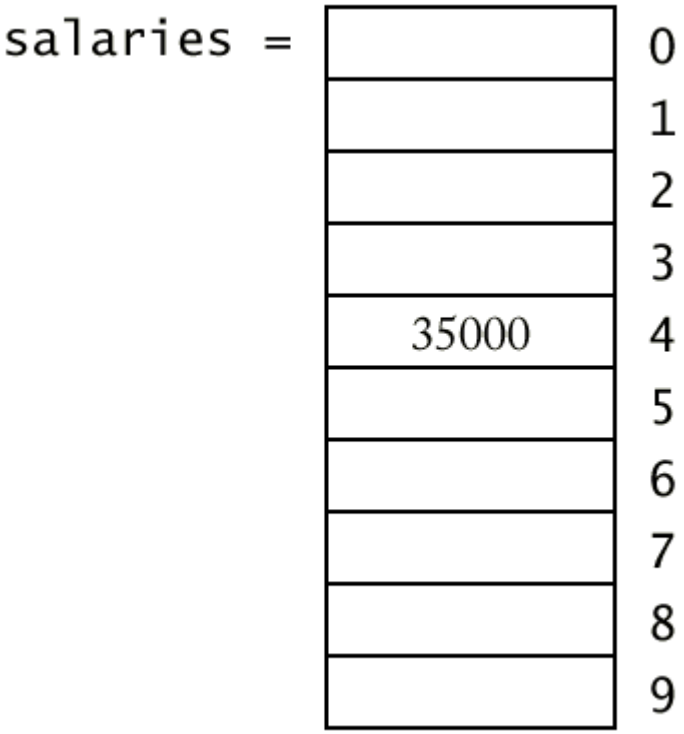
  - 벡터형의 변수를 정의한다. 벡터의 초기 크기를 지정할 수 있다.

- 예:

```
vector<int> scores;
vector<Employee> staff(20);
```

벡터

- 벡터 원소 지정: 첨자(index) 사용
  벡터변수_이름[정수수식]
- 첨자는 0부터 시작
- 예: salaries[4] = 35000;



벡터

- 벡터에 존재하지 않는 원소를 접근할 경우 오류
- vector<double> staff(10);

  cout << staff[10];

  /* 첨자범위는 0부터 9까지

- 크기인자가 주어지지 않으면 벡터크기는 0이다.

벡터

- 벡터 관련 함수
  - size() : 벡터 크기를 반환하는 함수
  - push_back(x): 벡터의 크기를 1 증가하고 벡터의 마지막에 원소 x를 저장한다. 즉 벡터의 마지막에 x를 추가한다.
  - pop_back(): 벡터의 마지막 원소를 삭제하고, 크기를 1 감소시킨다.

벡터 이용 예

```
#include <iostream>
#include <vector>
using namespace std;
 int main()
{
    vector<double> salaries;
    bool more = true;
    double highest
    int i;
    while (more)
    {
      double s;
      cout << "Please enter a salary, 0 to quit: ";
      cin >> s;
      if (s == 0)
          more = false;
      else
          salaries.push_back(s);
    }
    highest = salaries[0];
    for (i = 1; i < salaries.size(); i++)
            if (salaries[i] > highest)
                highest = salaries[i];
    cout << "highest value => " << highest  << "\n";

    return 0;
}
```

벡터 이용 예

- 함수의 인자가 벡터일 수 있다
- 벡터 인자 전달방법으로 값인자 및 참조인자 전달이 가능하다.
- 벡터의 원소 값을 변경할 경우 참조인자로 전달한다.

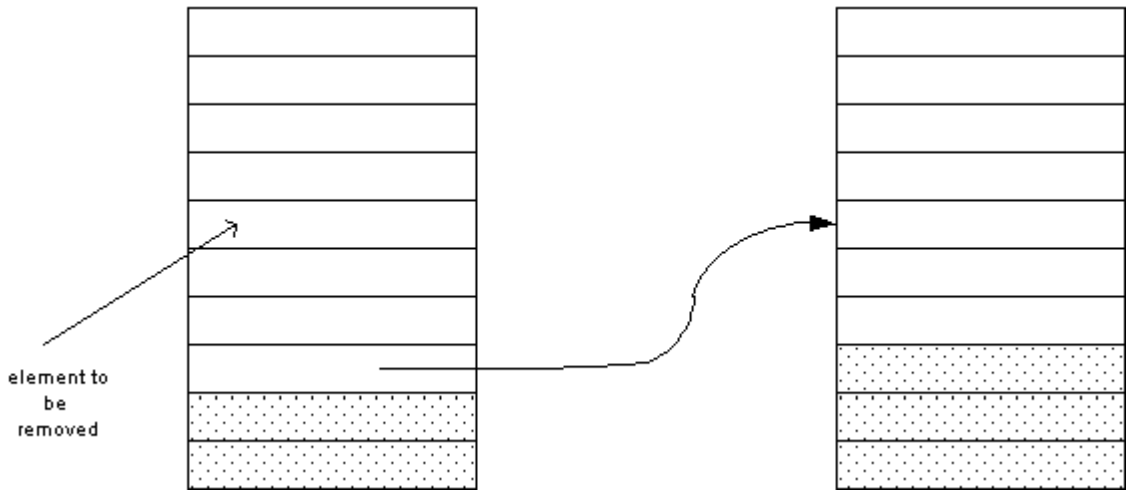
```
double average(vector<double> v)
{
    double sum = 0;
     int i;
    if (v.size() == 0) return 0;
    for (i = 0; i < v.size(); i++)
        sum = sum + v[i];
    return sum / v.size();
}
```

```
void raise_by_percent(vector<double>& v,
double p)
{
    int i;
    for (i = 0; i < v.size(); i++)
        v[i] =v[i] * (1 + p / 100);
}
```

벡터 이용 예

- 벡터에서 특정 위치 원소 삭제 예 (벡터에서 순서가 중요하지 않을 경우)

```
void erase(vector<string>& v, int pos)
{
    int last_pos = v.size() - 1;
    v[pos] = v[last_pos];
    v.pop_back();
}
```



벡터 이용 예

- 벡터에서의 특정 위치 원소 삭제 예 (벡터에서 순서가 중요할 경우)

```
void erase(vector<string>& v, int pos)
{
    int i;
    for (i = pos; i < v.size() - 1; i++)
        v[i] = v[i+1];
    v.pop_back();
}
```