6장 객체 배열과 벡터

2020. 9. 28

순천향대학교 컴퓨터 공학과

내용

- 객체 배열이란?
- 벡터(vector)
- 동적 객체 배열
- STL 알고리듬 함수
- Array 클래스

객체 배열이란?

- 객체 배열이란 배열 요소가 객체인 배열을 말함
- 배열 요소 객체 생성시 기본 생성자가 호출됨
- 기본 생성자가 정의되지 않으면 오류 발생

문법 6.1 객체 배열 선언

```
클래스_이름  배열_이름[배열_크기];
```

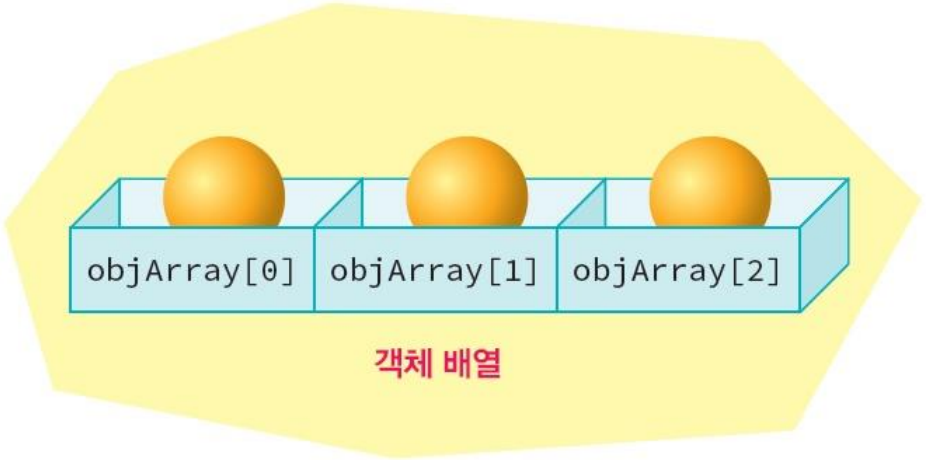


그림 6.1 객체 배열

예제

```
#include <iostream>
using namespace std;

class Circle
{
public:
        int x, y;
        int radius;
        Circle() : x{ 0 }, y{ 0 }, radius{ 0 } {  }
        Circle(int x, int y, int r) : x{ x }, y{ y }, radius{ r } {    }
        void print() {
          cout << "반지름: " <<
                  radius << " @(" << x
                  << ", " << y <<")" << endl;
        }
};
```

```
int main(void)
{
        Circle objArray[10];// 객체 배열

        for (Circle& c: objArray) {
                c.x = rand()%500;
                c.y = rand()%300;
               c.radius = rand()%100;
        }
        for (Circle c: objArray)
                c.print();

        return 0;
}
```

예제

```
#include <iostream>
using namespace std;

class Circle
{
public:
        int x, y;
        int radius;
        Circle() : x{ 0 }, y{ 0 }, radius{ 0 } {  }
        Circle(int x, int y, int r) : x{ x }, y{ y }, radius{ r } {     }
        void print() {
          cout << "반지름: " <<
                  radius << " @(" << x
                  << ", " << y <<")" << endl;
        }
};
```

```
int main(void)
{
        Circle objArray[10];

        for (Circle& c: objArray) {
                c.x = rand()%500;
                c.y = rand()%300;
                c.radius = rand()%100;
        }
        for (Circle c: objArray)
                c.print();

        return 0;
}
```

What if '&' is removed?

객체 배열 요소 접근

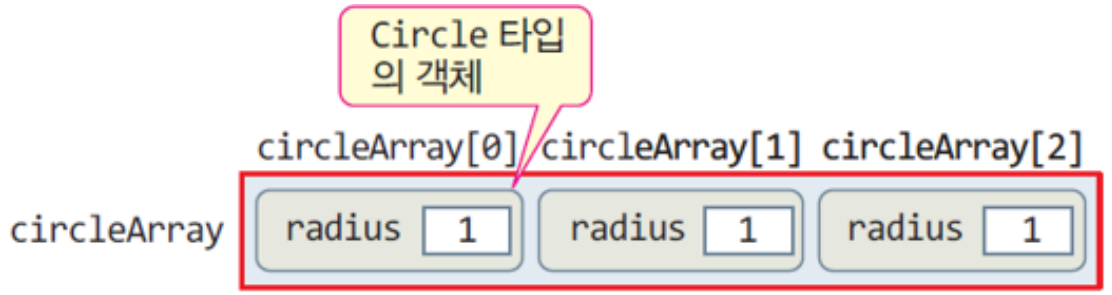
- 배열 요소 객체 접근은 배열 요소 참조 구문과 동일함

```
#include <iostream>
using namespace std;

class Circle {
   int radius;
public:
   Circle() {  radius = 1; }
   Circle(int r)  { radius = r; }
   void setRadius(int r)  { radius = r; }
   double getArea();
};

double Circle::getArea() {
   return 3.14*radius*radius;
}
```

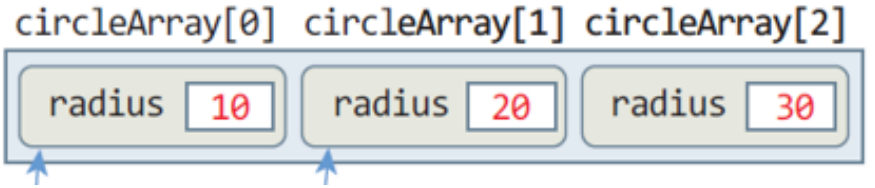
```
Int main() {
   Circle circleArray[3];

   // 배열의 각 원소 객체의 멤버 접근
   circleArray[0].setRadius(10);
   circleArray[1].setRadius(20);
   circleArray[2].setRadius(30);

   for(int i=0; i<3; i++) // 배열의 각 원소 객체의 멤버 접근
      cout << "Circle " << i << "의 면적은 " << circleArray[i].getArea() << endl;

}
```

```
int main() {
   Circle circleArray[3];

   // 배열의 각 원소 객체의 멤버 접근
   circleArray[0].setRadius(10);
   circleArray[1].setRadius(20);
   circleArray[2].setRadius(30);

   for(int i=0; i<3; i++) // 배열의 각 원소 객체의 멤버 접근
      cout << "Circle " << i << "의 면적은 " << circleArray[i].getArea() << endl;

}
```

(1) Circle circleArray[3];



(2)
```
circleArray[0].setRadius(10);
circleArray[1].setRadius(20);
circleArray[2].setRadius(30);
```



객체 배열 생성과 기본 생성자

- 다음 두 코드에서 문제가 발생하는가?

```
#include <iostream>
using namespace std;

class Circle {
   int radius;
public:
   double getArea() {
      return 3.14*radius*radius;
   }
};

int main() {
   Circle circleArray[3];
}
```

```
#include <iostream>
using namespace std;

class Circle {
   int radius;
public:
   Circle(int r) { radius = r; }
   double getArea() {
      return 3.14*radius*radius;
   }
};

int main() {
   Circle waffle(15);
   Circle circleArray[3];
}
```

객체 배열, 생성 및 소멸

- 객체 배열 선언
  - 기본 타입 배열 선언과 형식 동일
    - int n[3]; // 정수형 배열 선언
    - Circle c[3]; // Circle 타입의 배열 선언
- 객체 배열 생성
  - 객체 배열을 위한 공간 할당
  - 배열의 각 원소 객체마다 생성자 실행
    - c[0]의 생성자, c[1]의 생성자, c[2]의 생성자 실행
    - **매개 변수 없는 생성자 호출**
  - 매개 변수 있는 생성자를 호출할 수 없음
    - Circle circleArray[3](5); // 오류
- 배열 소멸
  - 배열의 각 객체마다 소멸자 호출. 생성의 반대순으로 소멸
    - c[2]의 소멸자, c[1]의 소멸자, c[0]의 소멸자 실행

객체 배열 초기화

- 객체 배열 요소를 서로 다르게 초기화하려면, 배열 요소별로 생성자를 호출한다.

```
Circle objArray[10] = {
            Circle(100, 100, 30),
            Circle(100, 200, 50),
            Circle(100, 300, 80),
            ...
};
```

예제: 객체 배열 초기화


```
#include <iostream>
using namespace std;

class Circle {
   int radius;
public:
   Circle() { radius = 1; }
   Circle(int r)  { radius = r; }
   void setRadius(int r)  { radius = r; }
   double getArea();
};

double Circle::getArea() {
   return 3.14*radius*radius;
}

int main() {
   Circle circleArray[3] = { Circle(10), Circle(20), Circle() }; // Circle 배열 초기화

   for(int i=0; i<3; i++)
      cout << "Circle " << i << "의 면적은 " << circleArray[i].getArea() << endl;
}
```

Lab: 책들을 저장해 보자.

- 여러 권의 책을 저장할 수 있는 객체 배열 books를 생성하여 보자.

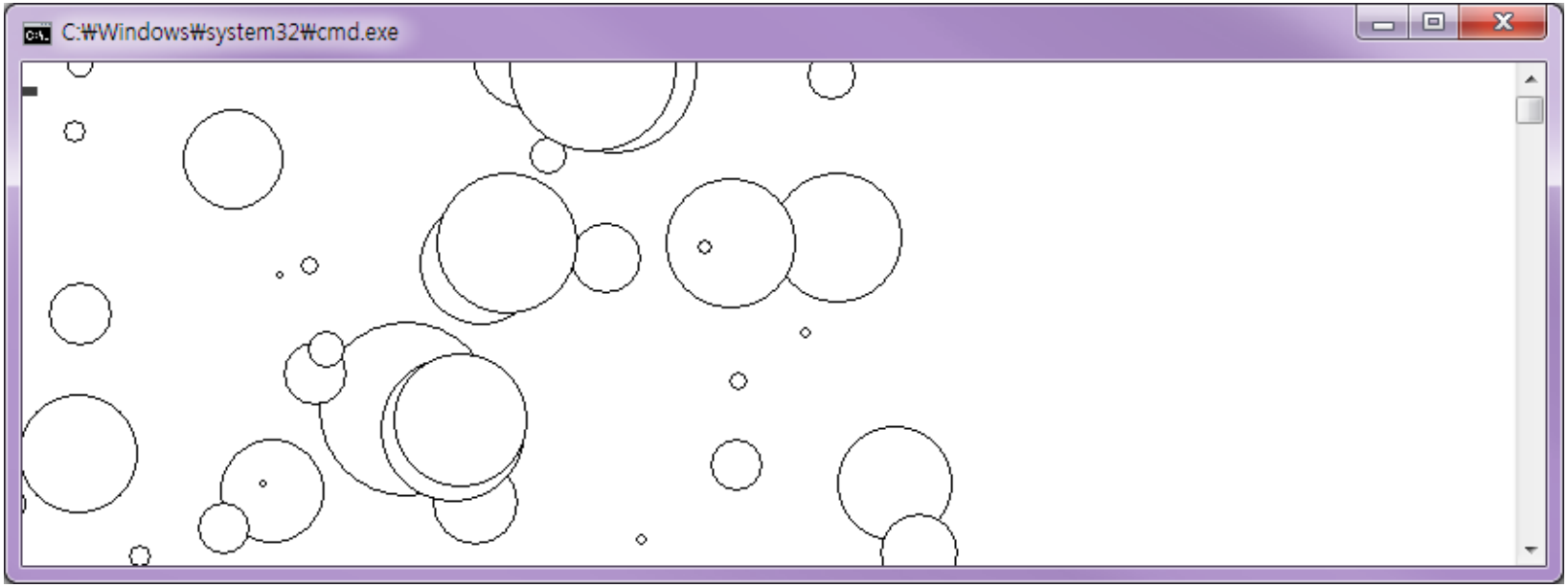
Book class


```
class Book
{
        string title;
        int price;

  public:
        Book(string name, int price) : title{ name }, price{ price } { }
        void print() {
            cout << "제목:" << title << ", 가격:" << price << endl;
        }
};
```

```
class Book
{
        string title;
        int price;
  public:
        Book(string name, int price) : title{ name }, price{ price } { }
        void print() {
            cout << "제목:" << title << ", 가격:" << price << endl;
        }
int main(void)
{
        Book books[2] = {
                Book("어서와 C++", 25000),
                Book("어서와 C  ", 22000)
        };

        cout << "소장하고 있는 책 정보" << endl;
        cout << "======================" << endl;

        for (Book& b : books)
                b.print();
        cout << "======================" << endl;

        return 0;
}
```

Lab: 원들을 저장해보자.

- 10개의 원을 저장할 수 있는 배열을 선언하고 사용자가 키 'c'를 누르면 각각의 원의 위치와 반지름을 난수로 초기화한 후에 화면에 그린다. 사용자가 키 'q'를 누르면 프로그램을 종료한다.



Solution: Circle class

```
#include <windows.h>
#include <conio.h> // for console input/output
#include <iostream>
using namespace std;

class Circle {
public:
        int x, y;
        int radius;
        Circle() : x{ 0 }, y{ 0 }, radius{ 0 } {  }
        Circle(int x, int y, int r) : x{ x }, y{ y }, radius{ r } {     }
        void draw()
        {
                int r = radius/2;
                HDC hdc = GetWindowDC(GetForegroundWindow());
                Ellipse(hdc, x-r, y-r, x+r, y+r);
        }
};
```

Solution: main()

```
int main(void)
{
	Circle objArray[10];

	while(true){
		for (Circle& c: objArray) {
			c.x = rand()%500;
			c.y = rand()%300;
			c.radius = rand()%100;
			c.draw();
		}
		char ch = _getch(); // 콘솔로부터 한 개 문자 입력
		if (ch == 'q') break;
	}
	return 0;
}
```

벡터

- 벡터는 C++ STL(Standard Template Library)에서 제공
  - **STL**에서 제공되는 컨테이너**(container) (**리스트**,** 스택**,** 큐 등을 포함**)**
  - 템플릿 클래스 형태로 제공

- 벡터(vector)는 동적 배열이다.
- 컴파일 시간에 배열의 크기를 미리 결정할 필요가 없다.
- 배열 표기법으로 임의 요소 접근

벡터의 선언

문법 6.1 벡터 선언

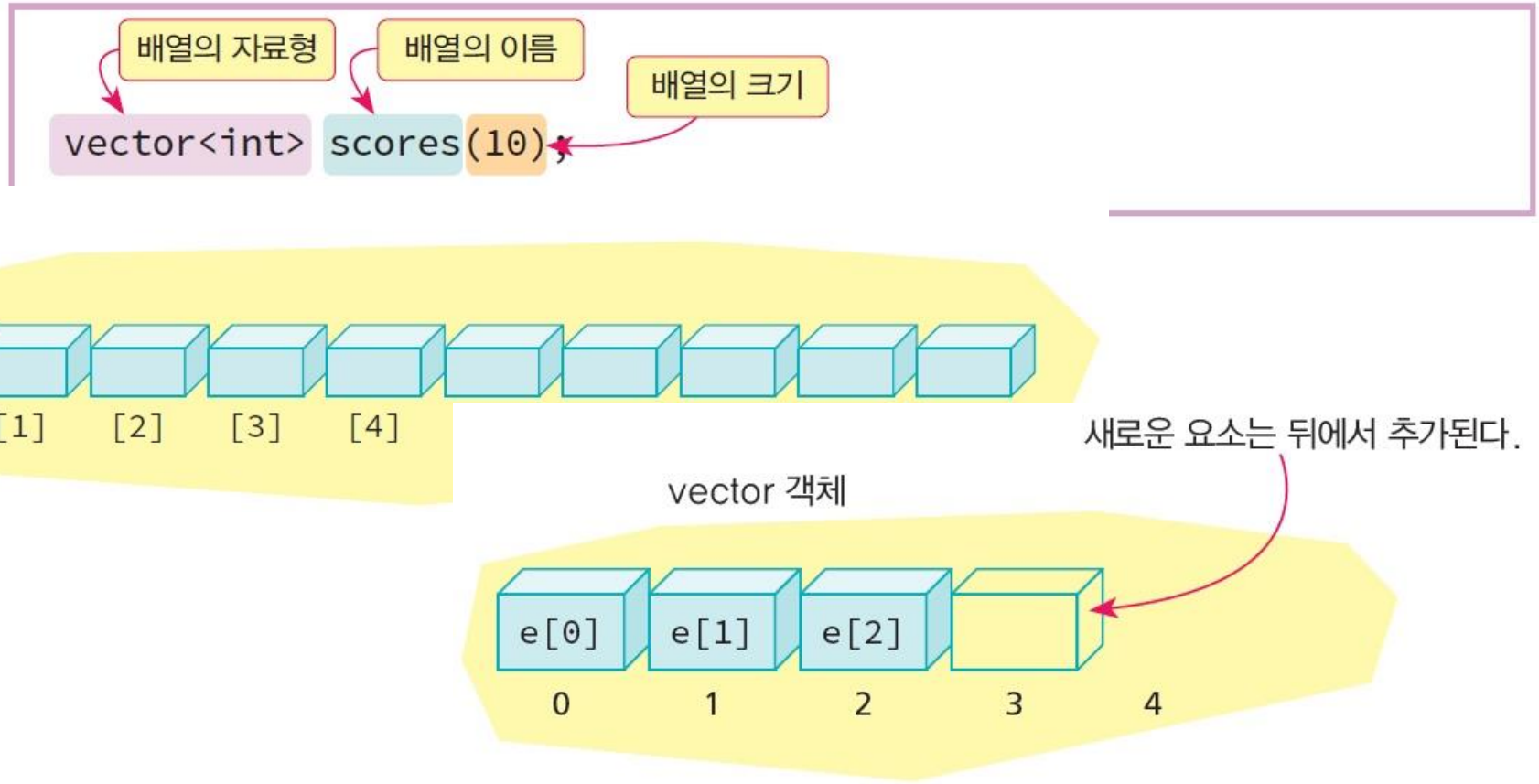


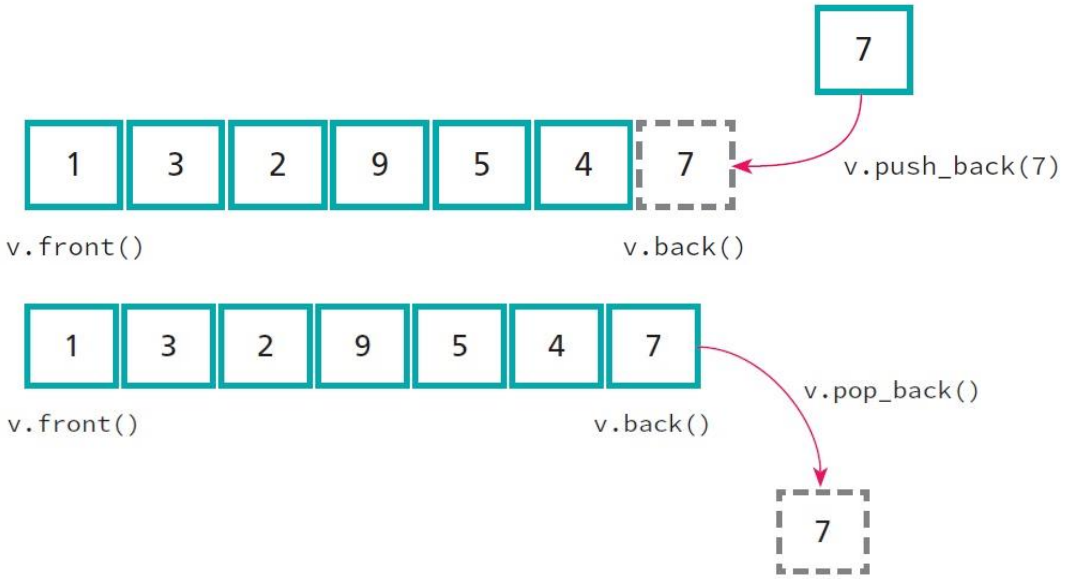
그림 6.2 벡터

벡터의 사용

```
#include <vector>
#include <iostream>
using namespace std;

int main(void)
{
	vector<int> fibonacci { 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89 };

	for (auto& number : fibonacci)
		cout << number << ' ';

	cout << endl;
	return 0;
}
```

벡터 연산: push_back(), pop_back()

- push_back()
  - 공백 벡터부터 시작하여 항목을 하나씩 추가하여 벡터 크기 확대
  - 비효율적: 벡터 크기 증가 시마다 벡터 이동 요구



- pop_back()
  - push_back()의 반대 기능
  - 삭제 요소 반환하지 않음에 주의

벡터 연산

- front()
  - 벡터의 첫번째 요소 참조
- back()
  - 벡터의 마지막 요소 참조

벡터의 사용 예

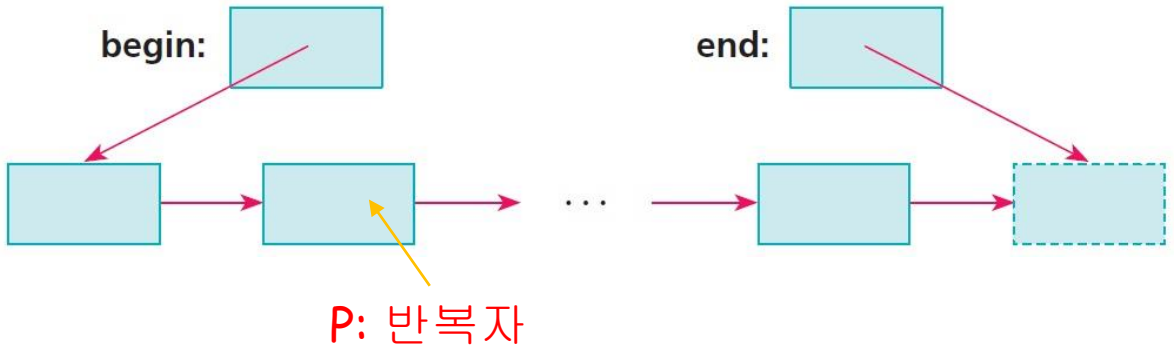
```
int main(void)
{
        vector<int> v1;

        v1.push_back(10);
        v1.push_back(20);
        v1.push_back(30);
        v1.push_back(40);
        v1.push_back(50);

        cout << "v1 = ";
        for (auto& e : v1) {
                cout << e << " ";
        }
        cout << endl;
        return 0;
}
```

벡터의 사용 예(2)

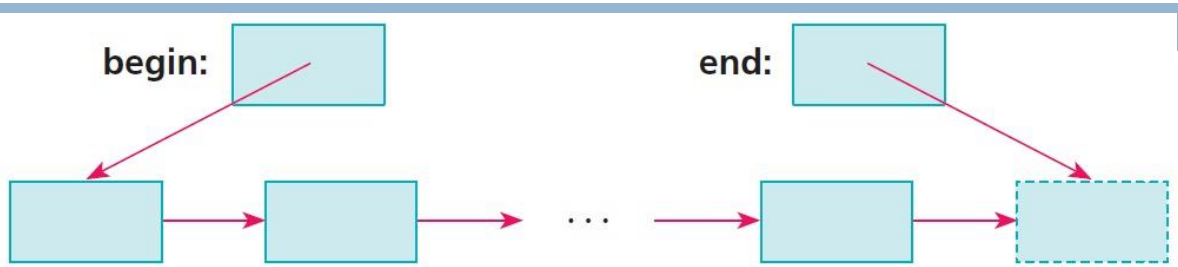
```
int main(void) {
        vector<int> v;

        for (int i = 0; i < 10; ++i) {
                v.push_back(i);
        }

        cout << "현재의  v =  ";
        for (auto& e : v)
                cout << e << " ";
        cout << endl;
        cout << "삭제 요소 =  ";

        while (v.empty() != true) {
                cout << v.back() << " ";
                v.pop_back();
        }
        cout << endl;
}
```

벡터 반복자

- 반복자 (iterator)는 컨테이너 원소들을 하나씩 순회 접근하는데 사용되는 컨테이너 원소에 대한 포인터이다.
- 반복자를 이용하여 벡터 요소의 위치 표현
  - begin()은 벡터의 첫번째 요소 식별
  - end()는 벡터의 벡터 끝을 하나 넘는 요소 식별



예제



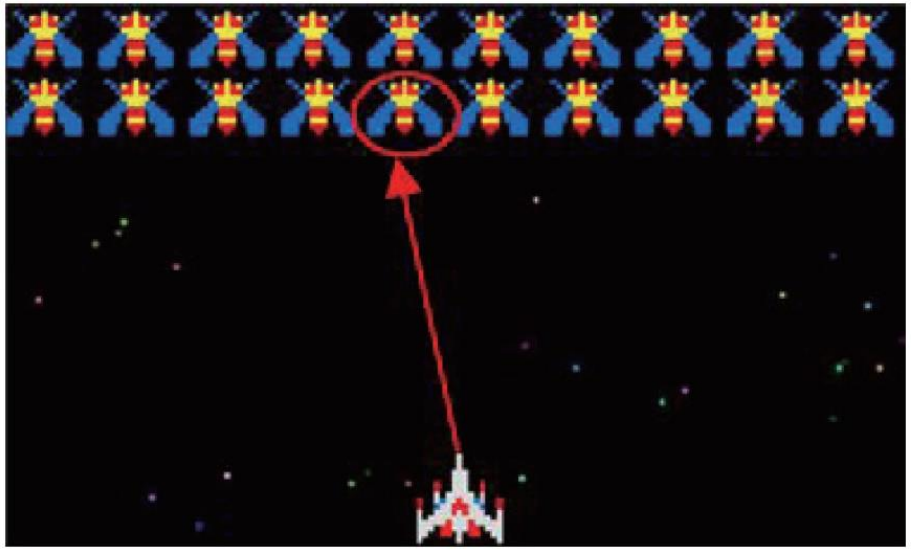

```
vector<int> v;
…
vector<int>::iterator p; // 벡터의 반복자 생성

for (p = v.begin(); p != v.end(); p++)
        cout << *p << endl;
```

반복자로부터 요소 참조

```
vector<int> v;
…

for (auto p = v.begin(); p != v.end(); ++p)
        cout << *p << endl;
```

벡터의 중간에서 삭제

- 벡터의 중간 요소 삭제 가능: erase() 이용



```
v.erase(v.begin()+i); // i번째 요소 삭제
```

벡터 연산자

- 벡터 복사: = (한 벡터를 다른 벡터에 복사)
- 벡터 비교: ==, != ('=='는 벡터 요소의 개수와 값이 모두 일치할 경우에만 true를 반환)

```
int main(void)
{
        vector<int> v1{ 1, 2, 3, 4, 5 };
        vector<int> v2(v1); // v1을 복사하여 v2를 생성: v2 = v1

        if (v1 == v2) {
                cout << "2개의 벡터가 일치합니다. " << endl;
        }
        return 0;
}
```

벡터를 이용한 객체 배열

```
#include <iostream>
#include <vector>
using namespace std;

int main(void)
{
	vector<string> vec;

	vec.push_back("MILK");
	vec.push_back("BREAD");
	vec.push_back("BUTTER");
	for (auto e : vec) {
		cout << " " << e;
	}
	cout << endl;
	return 0;
}
```

```
class Circle
{
public:
        int x, y;
        int radius;
        Circle() : x{ 0 }, y{ 0 }, radius{ 0 } {  }
        Circle(int x, int y, int r) : x{ x }, y{ y }, radius{ r } {     }
        void print() {
            cout << "반지름: " << radius << " @(" << x << ", " << y <<")" << endl;
        }
};
```

```
int main(void) {
        vector<Circle> objArray;

        for (int i = 0; i < 10; i++) {
                Circle obj{ rand()%300, rand()%300, rand()%100 };
                objArray.push_back(obj);
        }
        for (Circle c : objArray)
                c.print();

        return 0;
}
```

STL 알고리듬

- STL 컨테이너(벡터 포함) 사용시 다양한 STL 알고리듬(템플릿 함수)을 사용 가능하다.
  - 컨테이너 원소에 대한 복사, 검색, 삭제, 정렬 등의 기능 구현

copy	merge	random	rotate
equal	min	remove	search
find	move	replace	sort
max	partition	reverse	swap

STL 알고리즘 sort 함수 이용

```
#include <iostream>
#include <algorithm>  // STL 알고리듬 사용을 위해서 필요함
#include <vector>
using namespace std;

class Person {
private:
        string name;
        int age;
public:
        Person(string n, int a) {
                name = n;
                age = a;
        }
        string get_name() { return name; }
        int get_age()     { return age; }
        void print()      {
                cout << name << " "  << age << endl;
        }
};
```

```
bool compare(Person &p, Person &q)
{ return p.get_age() < q.get_age(); }

int main()
{
        vector<Person> list;

        list.push_back(Person("Kim", 30));
        list.push_back(Person("Park", 22));
        list.push_back(Person("Lee", 26));

        sort(list.begin(), list.end(), compare);
                // STL 알고리듬에서 제공하는 정렬 함수
                // 반복자를 이용한 정렬 범위 지정
                // 비교 함수 compare() 전달

        for (auto& e : list) {
                e.print();
        }
        return 0;
}
```

Lab: 성적평균 계산하기

- 학생들의 평균 성적을 계산하는 예제에서 학생이 몇 명인지 알 수 없다고 하자. 동적 배열인 벡터를 이용하여서 작성해 보자.

```
#include <iostream>
#include <vector>
using namespace std;

int main() {
        vector<int> scores;       // 이 선언문의 의미는?
        int i, sum = 0;

        while (true) {
                int score;
                cout << "성적을 입력하시오(종료는 -1) : ";
                cin >> score;
                if (score == -1) break;
                scores.push_back(score);
        }

        for (auto& value : scores) {
                sum += value;
        }
        double avg = (double)sum / scores.size();
        cout << "성적 평균=" << avg << endl;

        return 0;
}
```

Lab: 영화정보 저장

- 벡터를 이용하여 영화에 대한 정보를 저장했다가 다음과 같이 출력하는 프로그램을 작성해보자.

Movie class

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;

class Movie
{
private:
        string title;
        double rating;
public:
        Movie(string t = "", double r = 0.0) { title = t; rating = r; }
        void print_movie() { cout << title << ": " << rating << endl; }
};
```

main()

```
int main(void)
{
	vector<Movie> movies;

	movies.push_back(Movie("titinic", 9.9));
	movies.push_back(Movie("gone with the wind", 9.6));
	movies.push_back(Movie("terminator", 9.7));

	for (auto& e : movies)
		e.print_movie();

	return 0;
}
```

array 클래스

- vector는 빈번한 객체 복사와 소멸로 인해 비효율적
- C++11는 std::array를 새롭게 제공
  - 배열 크기는 미리 결정되어야 함
  - size(): 배열 크기
  - fill(): 배열 모든 요소를 동일한 값으로 채운다
  - empty(): 배열이 비어 있는지 검사
  - at(): 배열 요소 접근으로 []과 동일함
  - front(): 배열의 첫번째 요소
  - back(): 배열의 마지막 요소

예제

```
#include <iostream>
#include <array>
using namespace std;

int main()
{
        array<int, 3> list{ 1, 2, 3 };

        for (int i = 0; i<list.size(); ++i)
                ++list[i];

        for (auto& elem : list)
                cout << elem << " ";
        cout << endl;

        return 0;
}
```

배열 크기 명세해야 함