

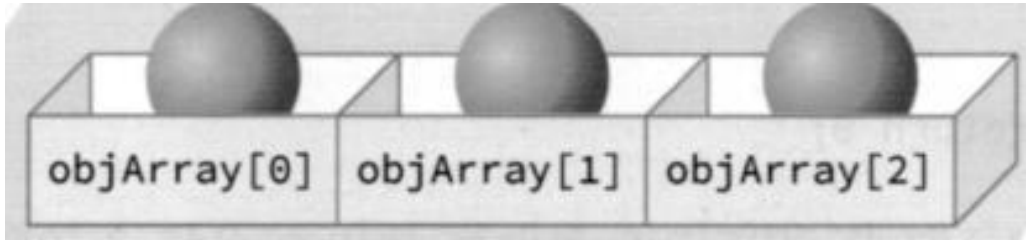
객체 배열

객체 배열

■ 객체를 요소로 가지는 배열

- 클래스명 배열_이름[배열_크기];

- Circle objArray[3];



- objArray[0].calcArea();

객체 배열

■ chapter06/ex01_object_array.cpp] 객체 배열

```
#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;
    }
};
```

객체 배열

■ chapter06/ex01_object_array.cpp] 객체 배열

```
int main() {  
    Circle objArray[10]; // 10개의 요소가 디폴트 생성자에 의해 생성  
  
    for(Circle& c; objArray) {  
        c.x = rand()%500;  
        c.y = rand()%300;  
        c.radius = rand()%100;  
    }  
  
    for(Circle c; objArray) {  
        c.print();  
    }  
  
    return 0;  
}
```

객체 배열

■ 객체 배열의 초기화

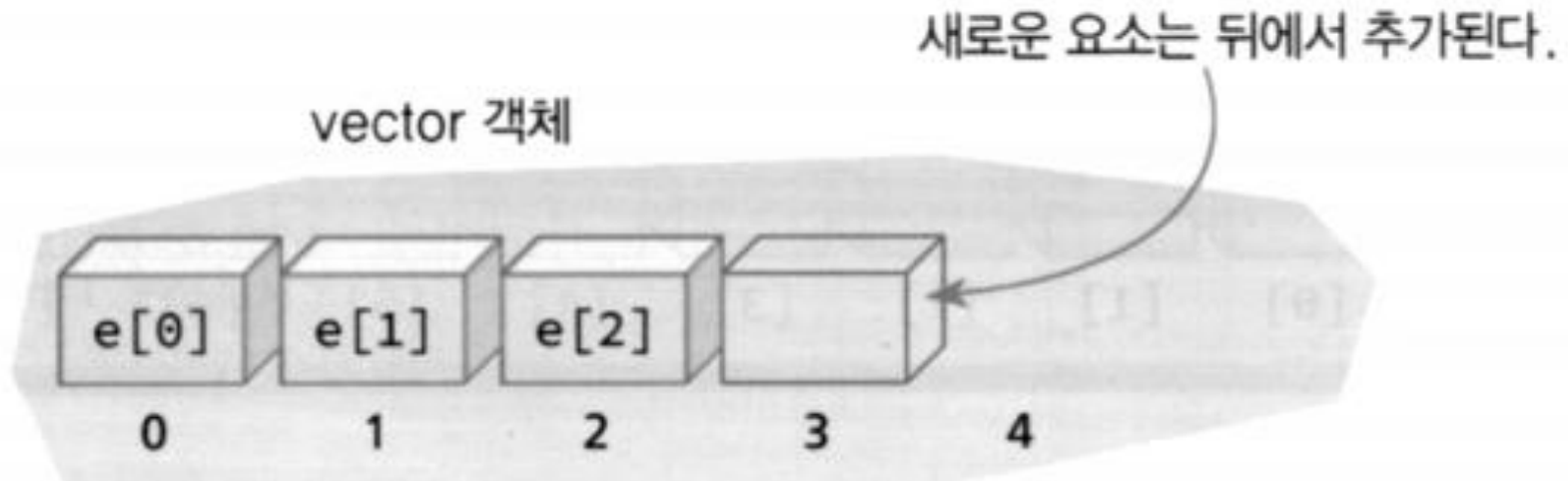
```
Circle objArray[10] = {  
    Circle(100, 100, 30),  
    Circle(100, 200, 50),  
    Circle(100, 300, 80),  
    ...  
};
```

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

벡터

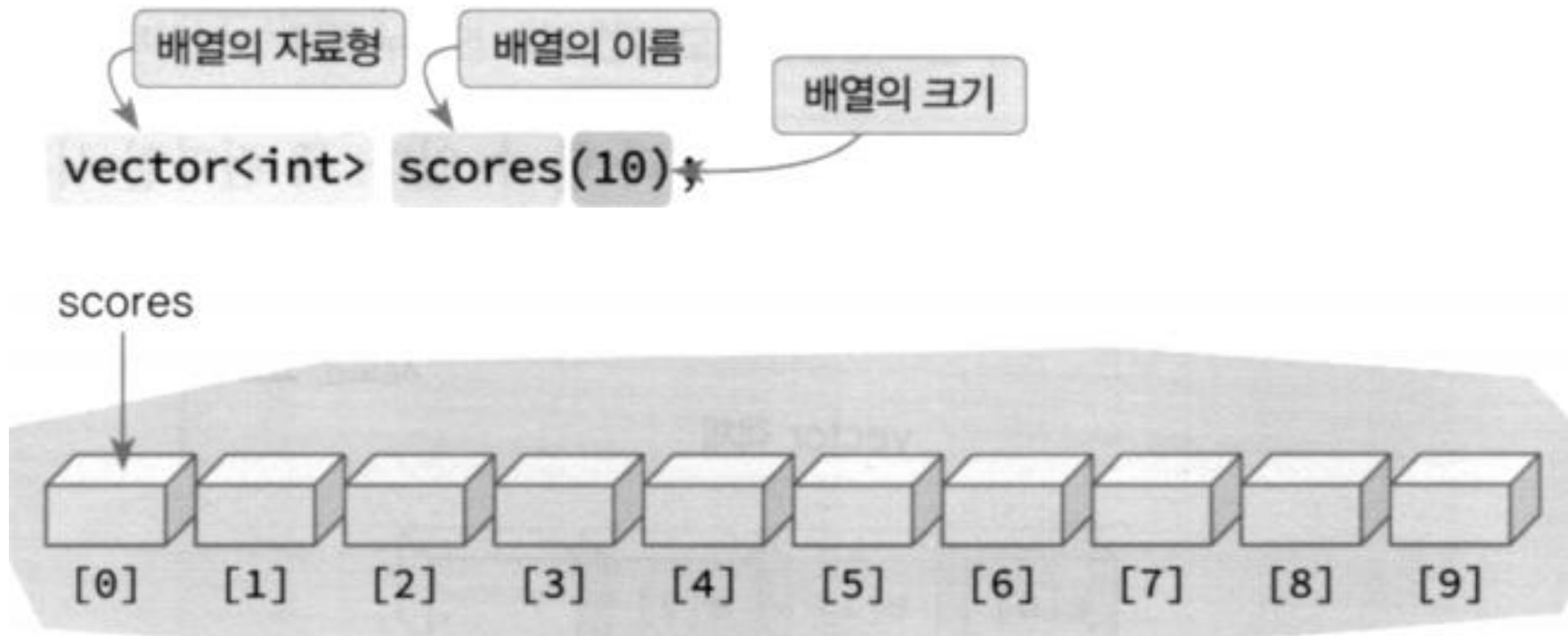
■ 벡터

- 배열은 크기가 고정되어 있는 단점이 있음
- 벡터는 동적으로 크기를 자동 조정
- `#include <vector>`



벡터

■ 벡터의 기초

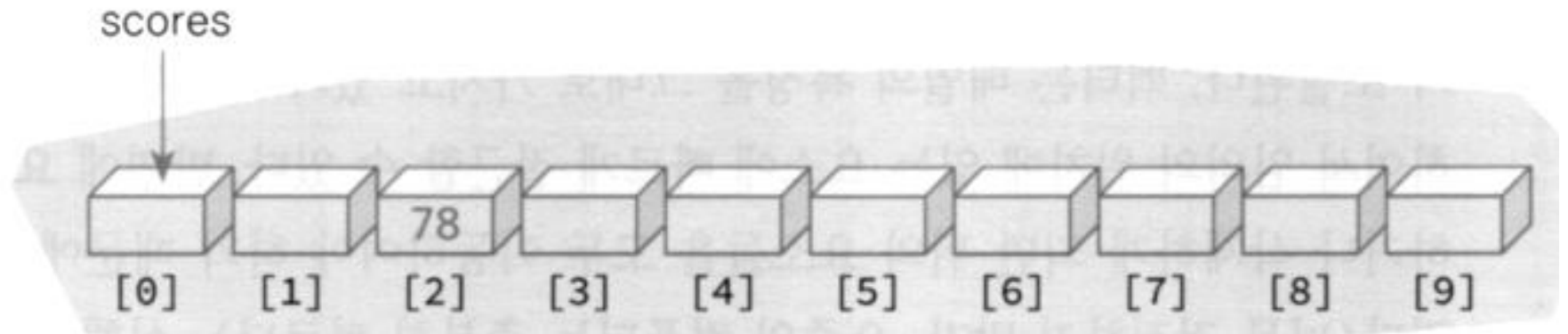


벡터

■ 벡터의 기초

```
scores[2] = 78;
```

```
cout << score[2] << endl;
```



벡터

■ chapter06/ex02_vector.cpp] 벡터의 기초

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

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

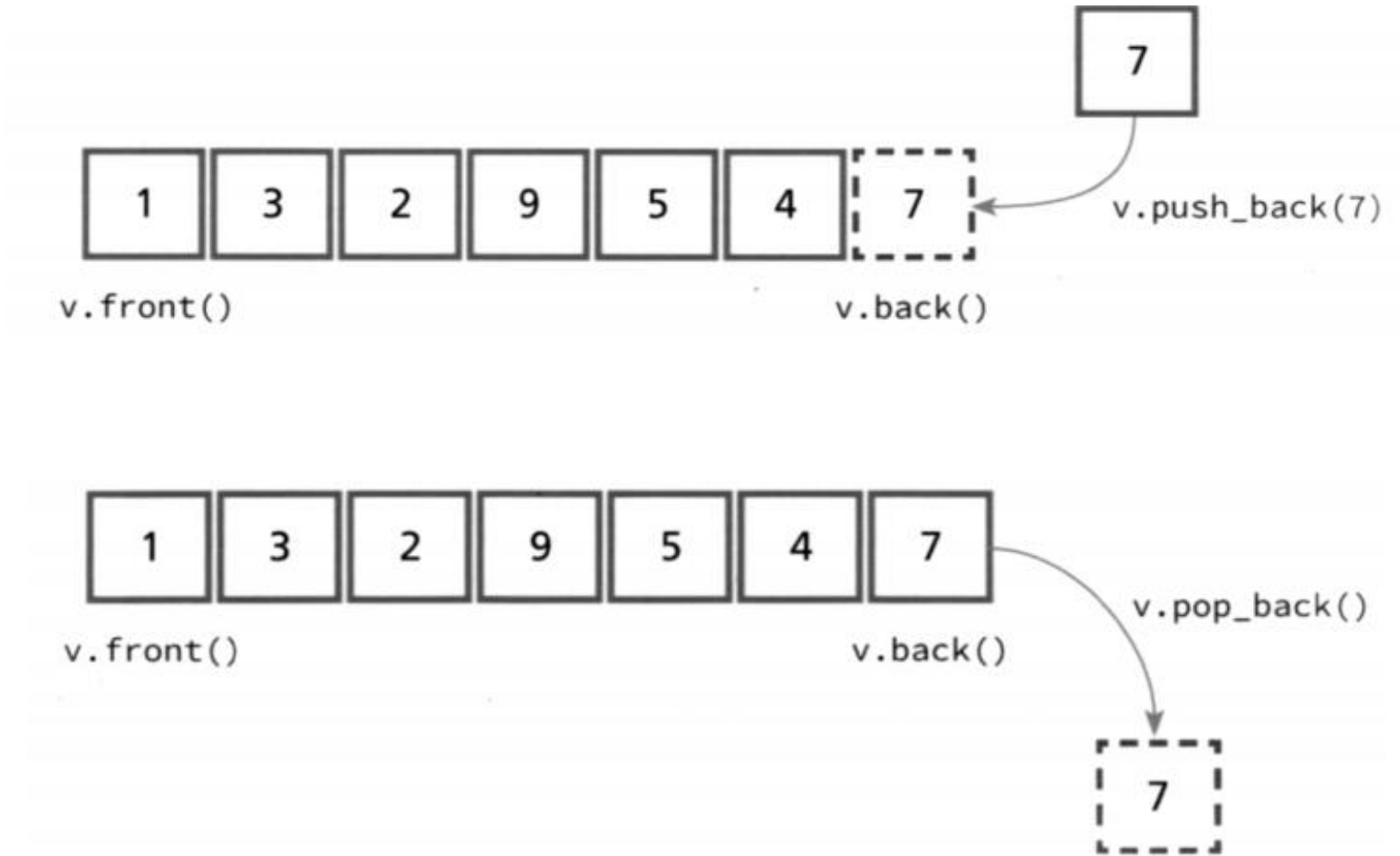
    for(auto& number; fibo) {
        cout << number << ' ';
    }

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

```
for (int i = 0; i < fibonacci.size(); i++)
    cout << fibonacci[i] << ' ';
```

벡터

■ push_back()과 pop_back()



■ chapter06/ex03_vector_op.cpp] push_back()과 pop_back()

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

int main() {
    vector<int> v;
    v.push_back(10);
    v.push_back(20);
    v.push_back(30);
    v.push_back(40);
    v.push_back(50);

    for(auto& e; v) {
        cout << e << ' ';
    }
    cout << endl;
    return 0;
}
```

■ chapter06/ex04_vector_op2.cpp] push_back()과 pop_back()

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

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

    for(auto& e; v) {
        cout << e << ' ';
    }
    cout << endl;
```

■ chapter06/ex04_vector_op2.cpp] push_back()과 pop_back()

```
cout << "삭제" << endl;

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

return 0;
}
```

벡터

■ 벡터에서 요소의 위치



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

■ 중간에서 삭제하는 방법

- `v.erase(v.begin()+i);`

벡터

■ chapter06/ex05_vector_op3.cpp] 벡터와 연산자

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

int main() {
    vector<int> v1{1, 2, 3, 4, 5};
    vector<int> v2(v1);

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

    return 0;
}
```

■ chapter06/ex06_object_save.cpp] 객체의 저장

```
#include <vector>
#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;
    }
};
```


■ chapter06/ex06_object_save.cpp] 객체의 저장

```
int main() {  
  
    vector<Circle> objArray;  
  
    for(int i=0; i<10; i++) {  
        Circle obj{rand()%300, rand()%300, rand()%100};  
        objArray.push_back(obj);  
    }  
  
    for(auto &c: objArray) {  
        c.print();  
    }  
    return 0;  
}
```