# 객체지향 프로그래밍: 함수와 문자열(3장)

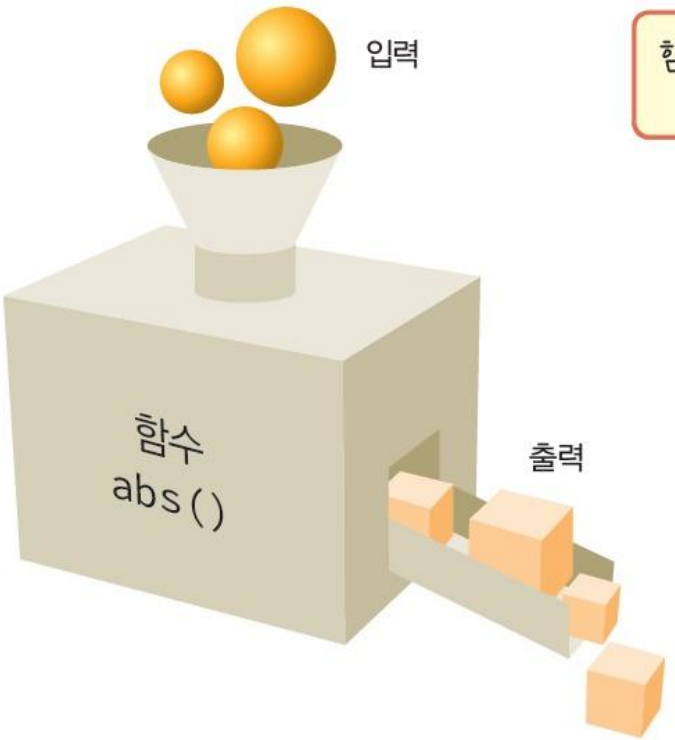
2020. 9. 7

순천향대학교 컴퓨터 공학과

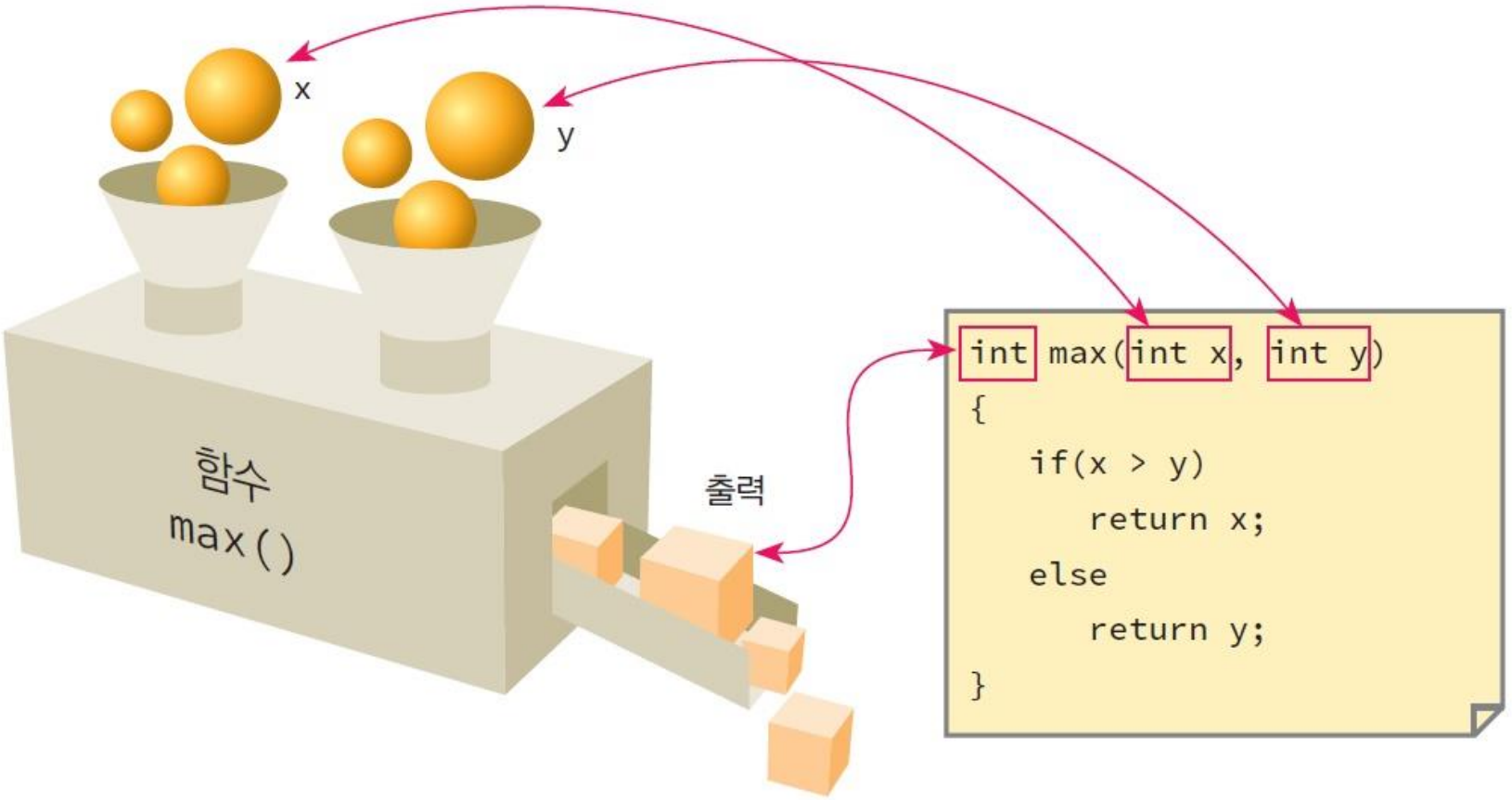
# 내용

1. 함수 개념
2. 매개변수 전달 방법
3. 참조 변수
4. 중복 함수
5. 디폴트 매개 변수
6. 인라인 함수
7. string 클래스
8. auto

# 함수란?




# 함수 선언



# 함수 호출

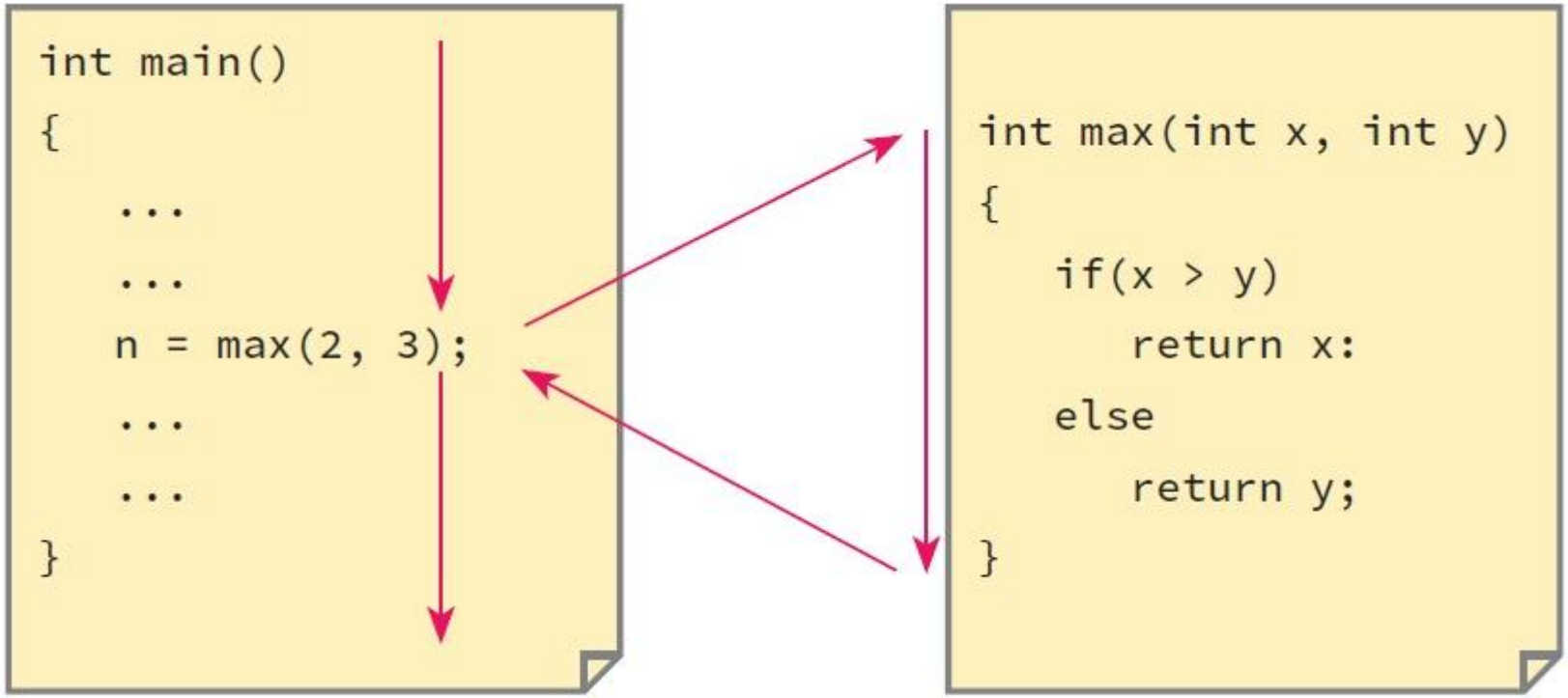



그림 3.1 함수 호출

# 함수 원형 선언

- 함수를 정의하기 전에 호출하고자 한다면 함수 원형(function prototype)을 미리 선언하는 것이 필요
- 함수 원형은 함수의 이름, 매개변수 타입, 반환 타입으로 구성

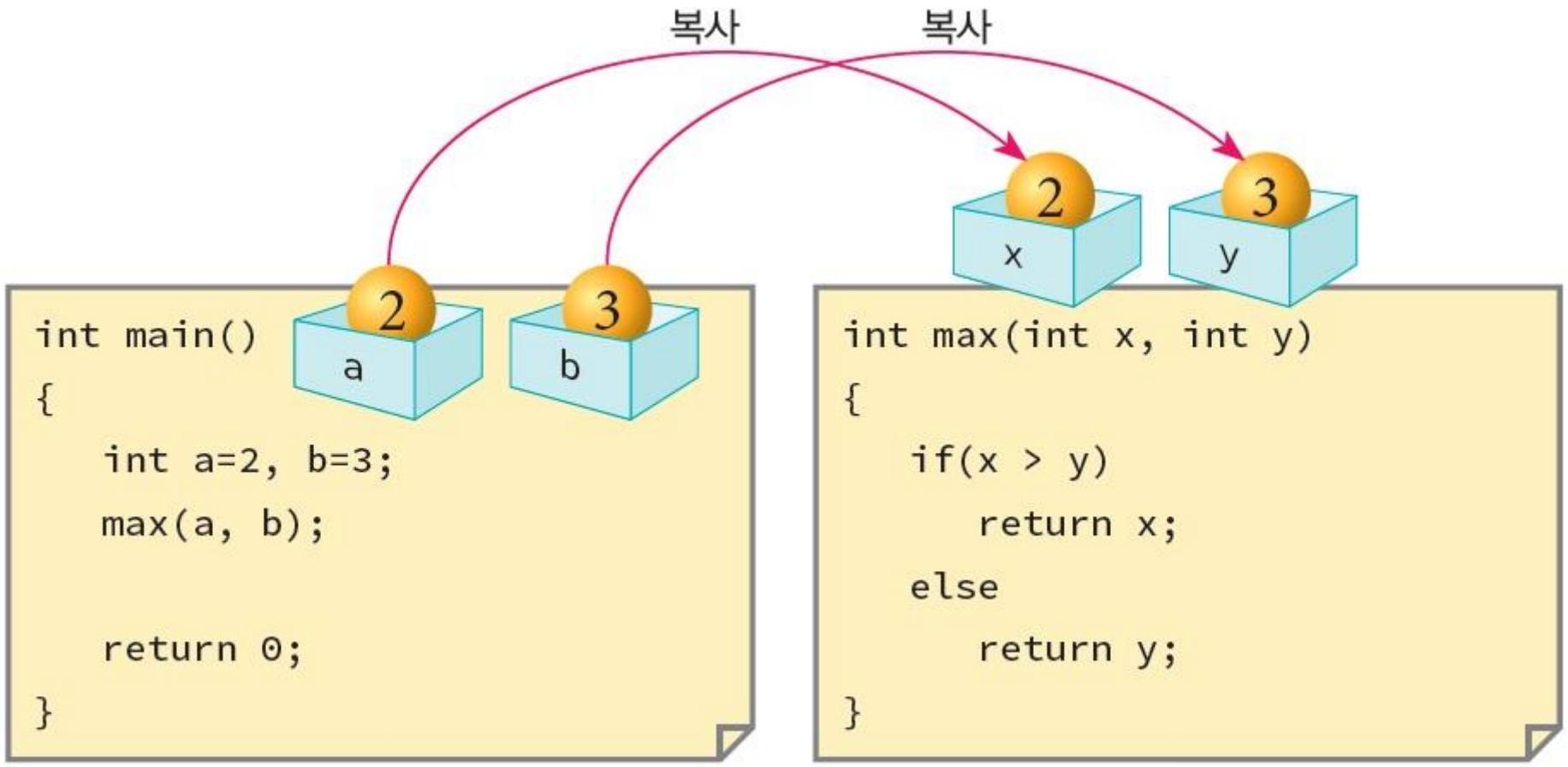
```
int square(int n); // 함수 원형 선언

int main()
{
        int result;

        result = square(5);
        printf("%d \n", result);
}
```

```
int square(int n)
{
        return(n * n);
}
```

# 매개변수 전달 방법

- 값으로 호출하기(call-by-value)
  - 인수의 값을 복사하여 전달
- 참조로 호출하기(call-by-reference)
  - 인수의 주소를 전달
  - C++는 참조 변수를 제공

# 값으로 전달하기


복사
복사
2
3
a
b
2
3
x
y
int main()
{
int a=2, b=3;
max(a, b);
return 0;
}
int max(int x, int y)
{
if(x > y)
return x;
else
return y;
}

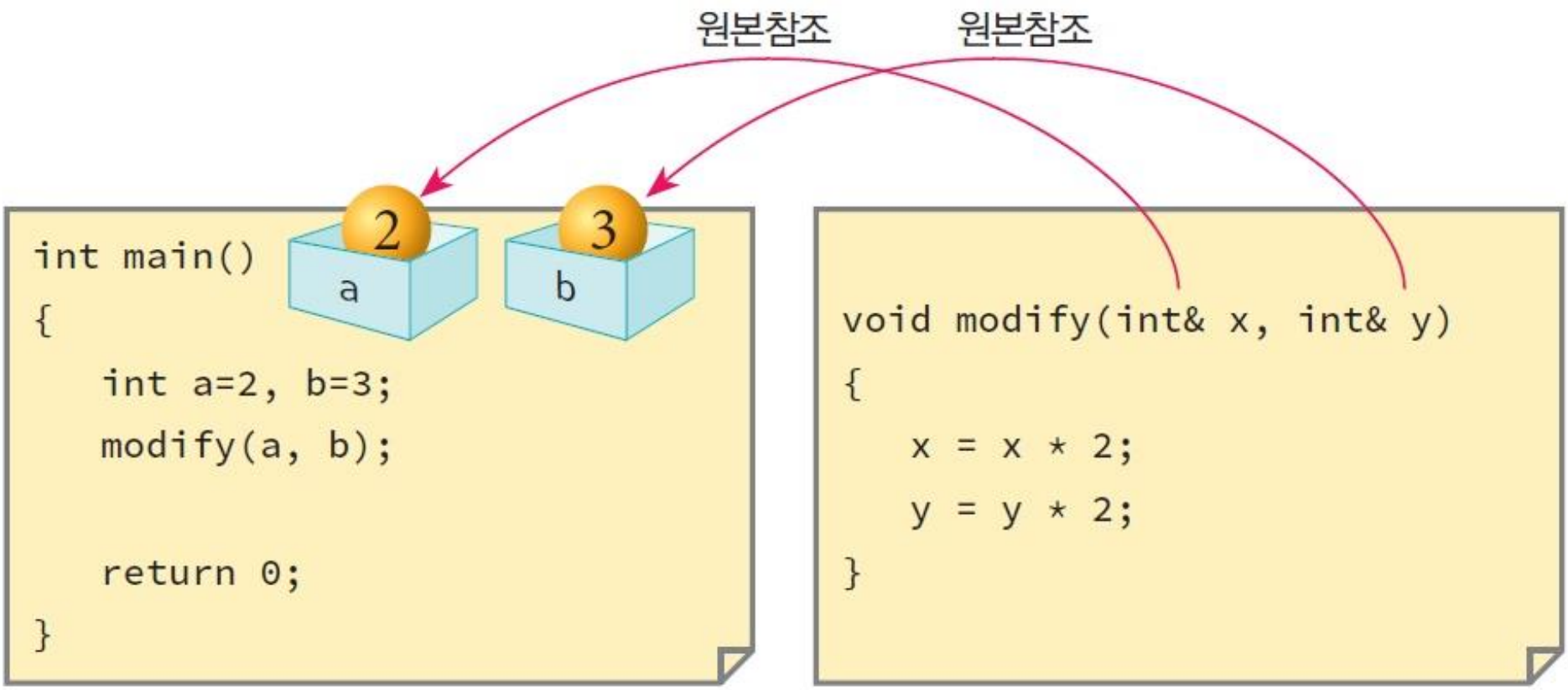
# 참조 변수

- C++에서 지원되는 특징
- 참조 변수는 이미 선언된 변수에 대한 별명
- 선언시 반드시 특정 변수로 초기화해야 함 (이후에 다른 변수 참조하는 것으로 변경 불가)

```
int n = 10;
int& ref = n; // ref는 참조 변수임


...
ref  = ref + 20;
cout << ref <<n << endl; // 출력 값은?
```

# 참조로 전달하기

- 매개변수로 참조 변수 이용



# Lab: 참조 변수 이용 swap() 함수

- 다음 프로그램에서 swap() 호출 후에 a, b의 값이 교환되게 하고자 한다. C++의 참조 타입을 이용하여 swap() 함수를 작성하라.

```
int main()
{
        int a = 100, b = 200;

        printf("a=%d b=%d\n", a, b);

        swap(a, b);
        printf("a=%d b=%d\n", a, b);

        return 0;
}
```

# 중복 함수

- 동일한 이름의 함수를 여러 개 정의하는 것을 **중복(된) 함수**(overloaded functions)라고 한다.
  - 중복 함수들간에는 시그너쳐가 달라야 한다.
  - 함수 시그너쳐는 매개변수의 개수, 타입, 순서로 구성
  - 언어에 따라서 반환 값 타입이 시그너처에 포함되기도 함

```
int square(int i) {
        return i*i;
}

double square(double i) {
        return i*i;
}
```

# 중복 함수

- 프로그램상의 중복 함수 호출은 그 시그너쳐에 기반하여 해당 함수 버전으로 매핑됨

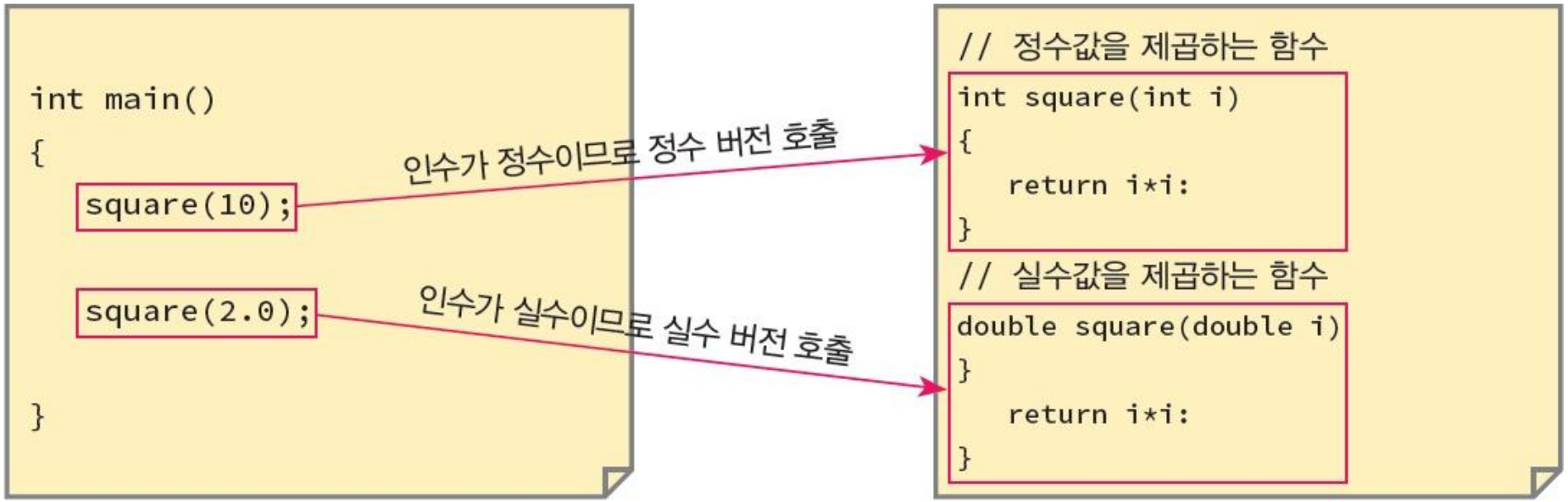



그림 3.3 중복 함수의 개념

# 디폴트 매개변수

- 인수를 전달하지 않아도 디폴트 값이 미리 설정되어 있는 매개변수를 디폴트 인수 (default argument)라고 한다.

```
void display(char c = '*', int n = 10)
{
        for (int i = 0; i < n; i++)
                cout << c;
        cout << endl;
}


…
display();
display('#');
display('#', 3);
```

# 디폴트 매개변수 (2)

- 디폴트 매개변수는 뒷부분에 위치해야 한다
  - 디폴트 매개변수 다음에 일반 매개변수가 오면 안된다.

```
int fun(double dvalue = 0.0, int prec); // 오류임
```

```
void func(int p1 = 100, int p2 = 200, int p3) {…}

func(10, 20); // 매개변수가 어디로 전달되는가?
```

# 예제: 디폴트 매개변수

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

int sum(int x, int y, int z = 0, int w = 0)
{
        return x + y + z + w;
}

int main()
{
        cout << "sum(10, 15)=" << sum(10, 15) << endl;
        cout << "sum(10, 15, 25)=" << sum(10, 15, 25) << endl;
        cout << "sum(10, 15, 25, 30)=" << sum(10, 15, 25, 30) << endl;

        return 0;
}
```

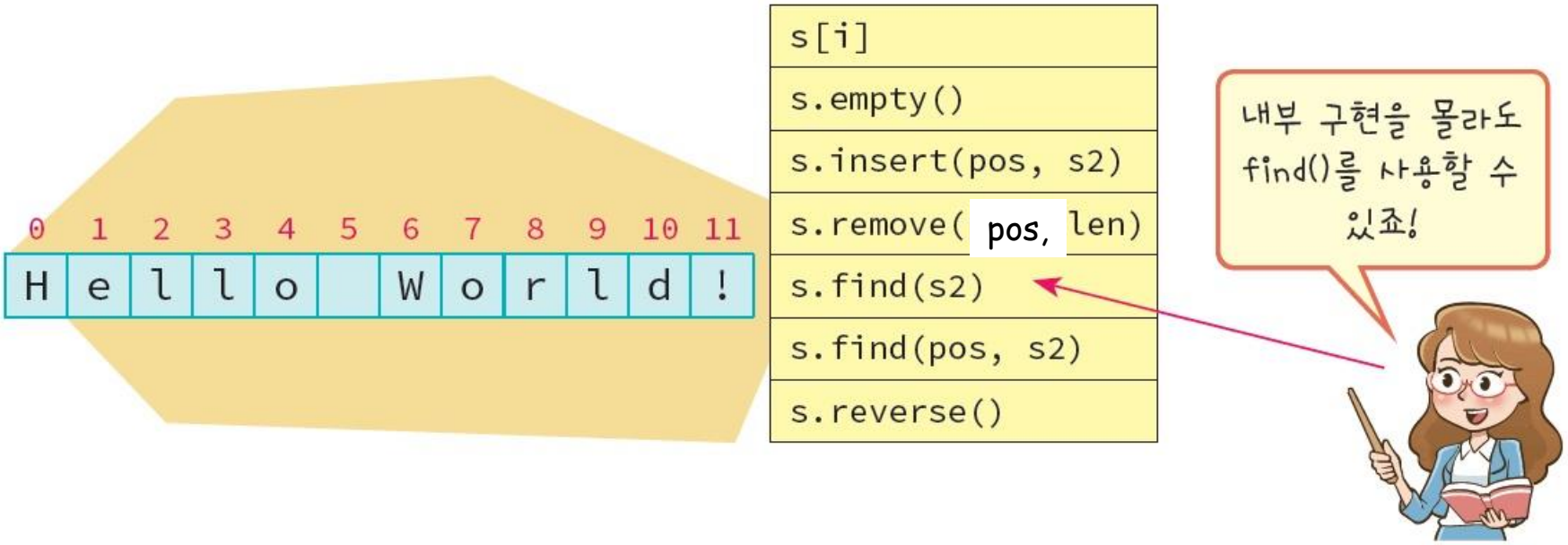
# 인라인 함수

- 함수 호출시에 부담(오버헤드)이 초래
- C++ 인라인 함수는 함수 호출을 회피하게 함
  - 인라인 함수는 함수 이름 앞에 inline이 붙는다
  - 컴파일러는 인라인 함수를 함수 호출로 처리하지 않고, 함수의 코드를 호출한 곳에 복사한다.

```
inline double square(double i)
{
        return i*i;
}
```

# string 클래스

- string 클래스

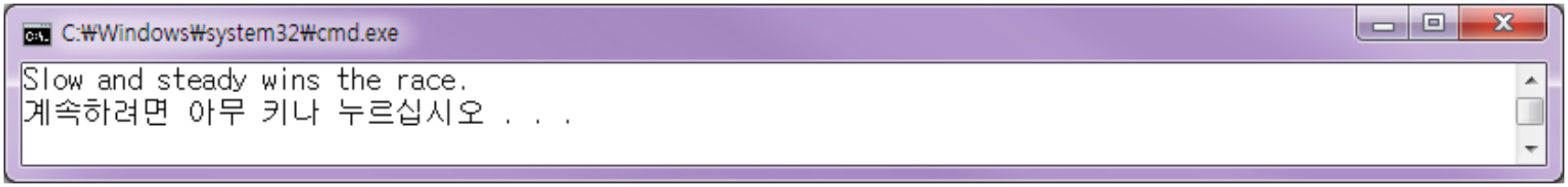


# 문자열 초기화

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

void main()
{

        string s;                           // string 객체 s를 생성한다.
        string s = "Hello World!";          // string 객체를 생성하고 초기화
        string s{ "Hello World!" };         // string 객체를 생성하고 초기화
}
```

# 문자열의 결합

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

int main()
{
        string s1 = "Slow", s2 = "steady";
        string s3 = "the race.";
        string s4;

        s4 = s1 + " and " + s2 + " wins " + s3;
        cout << s4 << endl;
        return 0;
}
```



# 문자열 비교

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

void main()
{
	string s1 = "Hello", s2 = "World";
	if( s1 == s2 )
		cout << "동일한 문자열입니다." << endl;
	else
		cout << "동일한 문자열이 아닙니다." << endl;
	if( s1 > s2 )
		cout << "s1이 앞이 있습니다. " << endl;
	else
		cout << "s2가 앞이 있습니다. " << endl;

}
```

# 문자열 입력

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

int main()
{
        string s1, addr;

        cout << "이름을 입력하시오 : ";
        cin >> s1;          // 문자열에 공백을 포함하면 그 이전까지만 입력
        cin.ignore();       // 엔터키를 없애기 위하여 필요하다.

        cout << " 주소를 입력하시오 : " ;
        getline(cin, addr); // 한 줄 전체 입력 (문자열에 공백 포함 가능)

        cout << addr << " 의 " << s1 << " 씨 안녕하세요? " << endl;

        return 0;
}
```

# string 클래스 멤버 함수 사용

| 멤버 함수 | 설명 |
|---|---|
| s[i] | i번째 원소 |
| s.empty() | s가 비어있으면 true 반환 |
| s.insert(pos, s2) | s의 pos 위치에 s2를 삽입 |
| s.remove(pos, len) | s의 pos 위치에 len만큼을 삭제 |
| s.find(s2) | s에서 문자열 s2가 발견되는 첫번째 인덱스를 반환 |
| s.find(pos, s2) | s의 pos 위치부터 문자열 s2가 발견되는 첫번째 인덱스를 반환 |

# find() 함수 이용

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

int main()
{
	string s="When in Rome, do as the Romans.";

	int index = s.find("Rome"); // R의 인덱스 반환
	cout << index << endl;

	return 0;
}
```

# string 객체에서 문자 추출하기

- String 객체에서 문자를 추출하려면 배열처럼 [] 연산자 적용

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

int main()
{
	string s="When in Rome, do as the Romans.";

	int index = s.find("Rome"); // R의 인덱스 반환
	cout << s[index] << endl;

	return 0;
}
```

# string 객체에서 문자 추출하기(2)

```
int main()
{
	string s;
	cout << "주민등록번호를 입력하시오: ";
	cin >> s;

	cout << "-가 제거된 주민등록번호: ";
	for (auto& c : s) {
		if (c == '-') continue;
		cout << c;
	}
	cout << endl;

	return 0;
}
```

# auto 키워드

- 자동 타입 추론(automatic type deduction)
  - 문맥 기반하여 변수 타입 추론

```
auto d = 1.0; // d의 타입은 double로 추론
```

```
auto add(int x, int y)
{
        return x + y;
}

int main()
{
        auto sum = add(5, 6); // sum의 타입은?
        return 0;
}
```

# 문자열의 배열

- 문자열 배열 표현

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

int main()
{
        string list[] = { "철수", "영희", "길동" };

        for (auto& s : list)        // '&' 삭제시 그 차이점은?
                cout << (s + "야 안녕!") << endl;
        return 0;
}
```

# Lab: 해밍거리 구하기

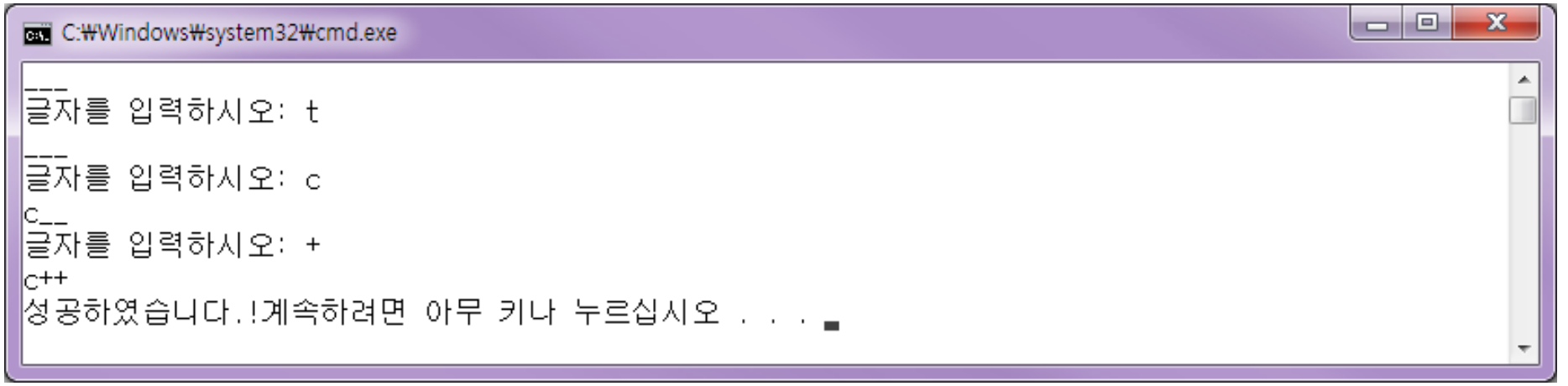
- 유전자를 나타내는 2개의 문자열을 받아서 동일한 위치에 틀린 글자가 몇 개나 있는지를 계산하는 프로그램을 작성해보자. 이것을 해밍 거리(Hamming distance)라고 한다.
  - 입력: 동일한 길이의 DNA 문자열
  - 출력: 문자열간의 해밍 거리

# Lab: 해밍거리 구하기(2)

```
int main() {
        string s1, s2;
        int count = 0;

        cout << "DNA1: "; // 2개 유전자 문자열 입력
        cin >> s1;
        cout << "DNA2: ";
        cin >> s2;

        if (s1.length() != s2.length())  // 문자열 길이 검사
                cout << "오류: 길이가 다름" << endl;
        else {
                for (int i = 0; i < s1.length(); i++) // 헤밍거리 조사
                    if (s1[i] != s2[i])
                        count += 1;

                cout << "해밍 거리는 " << count << endl;
        }
        return 0;
}
```

# Lab: 행맨


H I S M A S
Click here to display hint
A B C D E F
G H I J K L
M N O P Q R
S T U V W X
Y Z


C:₩Windows₩system32₩cmd.exe
___
글자를 입력하시오: t
___
글자를 입력하시오: c
c__
글자를 입력하시오: +
c++
성공하였습니다.!계속하려면 아무 키나 누르십시오 . . .

```
int main()
{   char ch;        string solution;
   string list[] = {"the", "c++", "programming", "language"}; // 사전 구축
   int n = rand() % 4;

   solution = list[n]; // 사전에서 임의 단어 선택
   string guess(solution.length(), '_'); // guess를  '_'으로 초기화

   while (true) {
            cout << guess << endl; // guess 현재 상태 보여주기
            cout << "글자를 입력하시오: ";  // 사용자가 글자 추측
            cin >> ch;

            for (int i=0; i < solution.length(); i++) // 추측 글자를 guess에 반영
                if (ch == solution[i])
                    guess[i] = ch;
            if (solution == guess) {
                    cout << solution << endl;
                    cout << "성공하였습니다.!";
                    break;
            }
   }
   return 0;
}
```