C 기반의 C++

박 종 혁 교수
UCS Lab
(http://www.parkjonghyuk.net)
Tel: 970-6702
Email: jhpark1@seoultech.ac.kr

목 차

- **스트림 입출력**
- **함수**
  - **오버로딩 (overloading)**
  - **디폴트 매개변수 (default parameter)**
  - **인-라인 함수 (in-line function)**
- **이름공간 (namespace)**

스트림 입출력

printf와 scanf

- 출력의 기본 형태 : 과거 스타일!
  - iostream.h 헤더 파일의 포함
  - HelloWorld1.cpp 참조

❖ cout << 출력 대상;

❖ cout<<출력 대상1<<출력 대상2<<출력 대상3;

❖ cout<<1<<'a'<<"String"<<endl;

printf와 scanf

- 출력의 기본 형태 : 현재 스타일!
  - iostream 헤더 파일의 포함
  - HelloWorld2.cpp

❖ std::cout << 출력 대상;

❖S td::cout<<출력 대상1<<출력 대상2<<출력 대상3;

❖ std::cout<<1<<'a'<<"String"<<std::endl;

printf와 scanf

- 입력의 기본 형태 : 과거 스타일!
  - iostream.h 헤더 파일의 포함
  - SimpleAdder1.cpp, BetweenAdder.cpp

❖ cin>>입력 변수;

❖ cin>>입력 변수1>>입력 변수2>>입력 변수3;

❖ cin>>val1;

printf와 scanf

- 입력의 기본 형태 : 현재 스타일!
  - iostream 헤더 파일의 포함
  - SimpleAdder2.cpp

❖ std::cin>>입력 변수;

❖ std::cin>>입력 변수1>>입력 변수 2>>입력 변수3;

❖ std::cin>>val1;

입출력 연산자

- C++ 언어에서는 스트림 입출력 연산자를 사용하여 표준 입출력
  - 스트림(stream)이란 물리적인 입출력 장치를 입출력이 순서대로 이루어지는 가상의 통로로 정의
  - cin, cout, cerr로 정의된 예약어와 <<, >> 스트림 연산자와 함께 사용

```
표준입력
    #include <iostream.h>
    cin >> 변수 [ >> 변수 ..... ];
표준출력
    #include <iostream.h>
    cout << 식 [ << 식 .....];
표준에러출력
    #include <iostrem.h>
    cerr << 식 [ << 식 .....];
```

```
표준입력
    #include <iostream>
    std::cin >> 변수 [ >> 변수 ..... ];
표준출력
    #include <iostream>
    std::cout << 식 [ << 식 .....];
표준에러출력
    #include <iostrem>
    std::cerr << 식 [ << 식 .....];
```

입출력 예

```
#include <stdio.h>
int main(void)
{
    int sum, i, n;
    scanf("%d", &n);
    sum = 0;
    for (i=1; i <= n; i++)
            sum += i;
     printf("sum = %d₩n", sum);
     return 0;
}
```

```
#include <iostream.h>
int main(void)
{
    int sum, i, n;
    cin >> n;
    sum = 0;
     for (i=1; i <= n; i++)
            sum += i;
      cout << "sum = " << sum << endl;
      return 0;
}
```

```
#include <iostream>
int main(void)
{
    int sum, i, n;
    std::cin >> n;
    sum = 0;
     for (i=1; i <= n; i++)
            sum += i;
      std::cout << "sum = " << sum << endl;
      return 0;
}
```

함 수

함수 오버로딩(overloading)

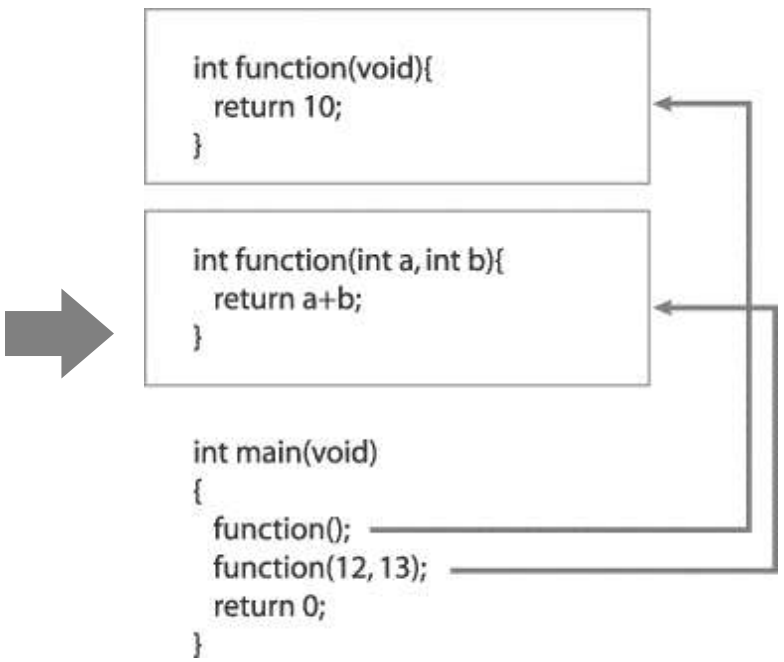
- 함수 오버로딩 (function overloading)
  - C++ 언어에서는 같은 이름을 가진 여러 개의 함수를 정의 가능
  - 같은 이름을 갖는 각 함수는 인수의 형과 갯수, 함수의 반환형에 의해 구분

- 함수 뿐만 아니라 연산자 오버로딩도 가능한데 이에 관한 자세한 소개는 10장에서 소개

함수 오버로딩(overloading)

- 함수 이름의 중복!
  - 파일의 확장자는 .C이다! 무엇이 문제?

```
int function(void){
  return 10;
}

int function(int a, int b){
  return a+b;
}

int main(void)
{
  function();
  function(12, 13);
  return 0;
}
```



함수 오버로딩(overloading)

- 함수 오버로딩이란?
  - 동일한 이름의 함수를 중복해서 정의하는 것!
- 함수 오버로딩의 조건
  - 매개 변수의 개수 혹은 타입이 일치하지 않는다.
- 함수 오버로딩이 가능한 이유
  - 호출할 함수를 매개 변수의 정보까지 참조해서 호출
  - 함수의 이름 + 매개 변수의 정보

함수 오버로딩(overloading)

- 함수 오버로딩의 예
  - FunctionOverloading.cpp

```
int function1(int n){…}
int function1(char c){…}
```

```
int function2(int v){…}
int function2(int v1, int v2){…}
```

오버로딩 예

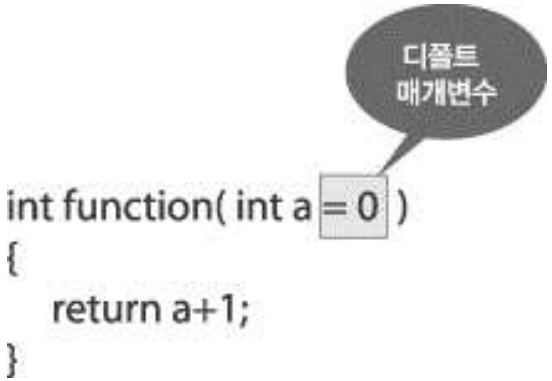
```
// 절대값 함수의 중복 정의
#include <iostream>

int abs(int n);              // 정수의 절대값 함수
double abs(double n);
                    // double 형 실수의 절대값 함수

int main(void)
{
   std::cout << "-123     절대값 : "
       << abs(-123) << '₩n';
   std::cout << "-12.345 절대값 : "
       << abs(-12.345) << '₩n';

  return 0;
}

// 정수형 인수의 abs() 함수
int abs(int n)
{
   return n < 0 ? -n : n;
}
// double 정수형 인수의 abs() 함수
double abs(double n)
{
   return n < 0 ? -n : n;
}
```

-123 절대값 : 123
-12.345 절대값 : 12.345

절대값을 구하는 2개의 함수가 모두 같은 이름의 abs()로 사용되었으나 각 함수는 서로 다른 인수의 형에 따라 구분된다.

디폴트 매개변수(인수)

- 디폴트 매개 변수란?
  - 전달되지 않은 인자를 대신하기 위한 기본 값이 설정되어 있는 변수



디폴트 매개변수 예

```
#include <iostream>

int BoxVolume(int length, int width=1, int height=1);

int main(void)
{
    std::cout<<"[3, 3, 3]      : "<<BoxVolume(3, 3, 3)<<std::endl;
    std::cout<<"[5, 5, def]    : "<<BoxVolume(5, 5)<<std::endl;
    std::cout<<"[7, def, def]  : "<<BoxVolume(7)<<std::endl;
    return 0;
}

int BoxVolume(int length, int width, int height)
{
    return length*width*height;
}
```

```
[3, 3, 3]       :  27
[5, 5, def]     :  25
[7, def, def]   :  7
```

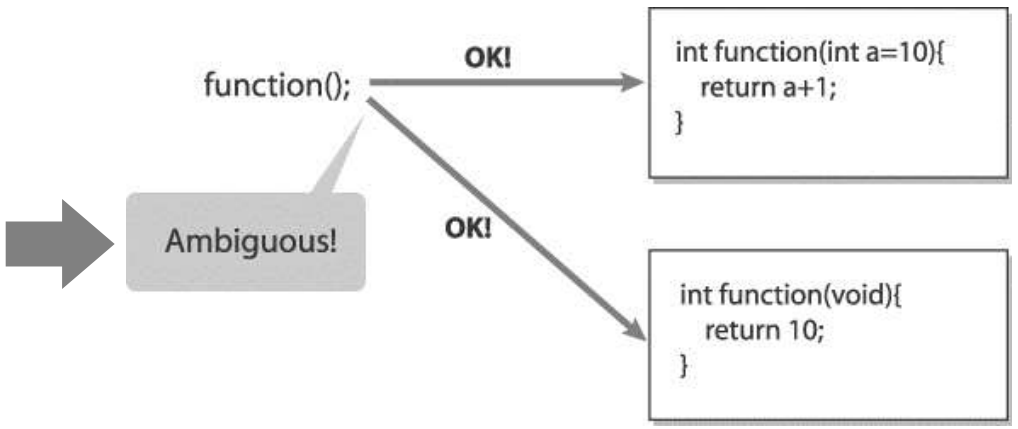
디폴트 매개변수(인수)

- C++ 언어에서는 함수 선언 시 인수의 디폴트(default) 값을 지정
  - 디폴트 값이 지정된 인수는 함수 호출 시 인수를 생략
  - 인수가 생략된 경우에는 자동적으로 디폴트 값으로 선언된 값을 인수로 전달
  - 인수의 디폴트 값의 선언 시 디폴트 값이 선언된 인수 뒤에 오는 모든 인수에 대해서는 디폴트 값을 지정해야 함.

디폴트 매개변수와 함수 오버로딩

```
#include<iostream>

int function(int a=10)
{
  return a+1;
}
int function(void)
{
  return 10;
}

int main(void)
{
  std::cout<<function(10)<<std::endl;
  return 0;
}
```

std::cout<<function()<<std::endl;



인-라인 함수(inline function)

- 인-라인 함수를 이용하여 정의된 함수를 호출하면 호출한 함수로 분기하여 함수를 실행한 후 복귀하는 것이 아니라 호출한 함수의 내용을 복사하여 실행
  - C 언어의 매크로 정의함수를 사용하는 것과 같은 기능
  - 매크로 정의함수인 경우 인수의 전달 시 부작용(side effect)을 갖는데 반하여 확장함수는 일반함수와 같이 부작용 없이 안전하게 사용
  - 인-라인 함수는 일반함수와 같이 함수로 분기하여 실행한 후 복귀하는 과정이 없이 직접 실행하므로 일반함수 보다 프로그램의 실행속도가 빠름
  - 함수를 호출할 때 마다 함수의 내용이 복사되므로 프로그램의 코드 길이가 길어진다는 단점
  - 일반적으로 확장함수는 코드의 길이가 짧은 경우에 주로 사용
  - 확장함수의 정의는 함수 정의 앞에 inline이란 키워드를 붙임

인라인-함수 사용 예

```
#include <iostream>
#define SQUARE(x) ((x)*(x))

int main(void)
{
    std::cout<<SQUARE(5)<<std::endl;
    return 0;
}
```

```
#include <iostream>
inline int SQUARE(int x)
{
    return x*x;
}

int main(void)
{
    std::cout<<SQUARE(5)<<std::endl;
    return 0;
}
```

이름공간(namespace)

이름공간(namespace)

- 이름공간
  - 최근 C++에 추가된 기능으로 변수,함수 이름 등의 영역 지정기능
- 사용 예
  - Lee와 Kim이 두 사람이 독자적으로 프로그램 개발 후 통합하는 예
  - 두 사람 모두 다른 기능의 foo() 함수 정의
  - namespace으로 이름공간 정의, :: 으로 이름공간 표시

```
        ……
namespace Lee {
    ……
    void foo() { … }
    ……
}
        ……
```

```
        ……
namespace Kim {
    ……
    void foo() { … }
    ……
}
        ……
```

```
        ……
main()
{
    ……
    Kim::foo();
    ……
    Lee::foo();
}
```

이름공간(namespace)

- 이름 공간이란?
  - 공간에 이름을 주는 행위!
  - "202호에 사는 철수야"

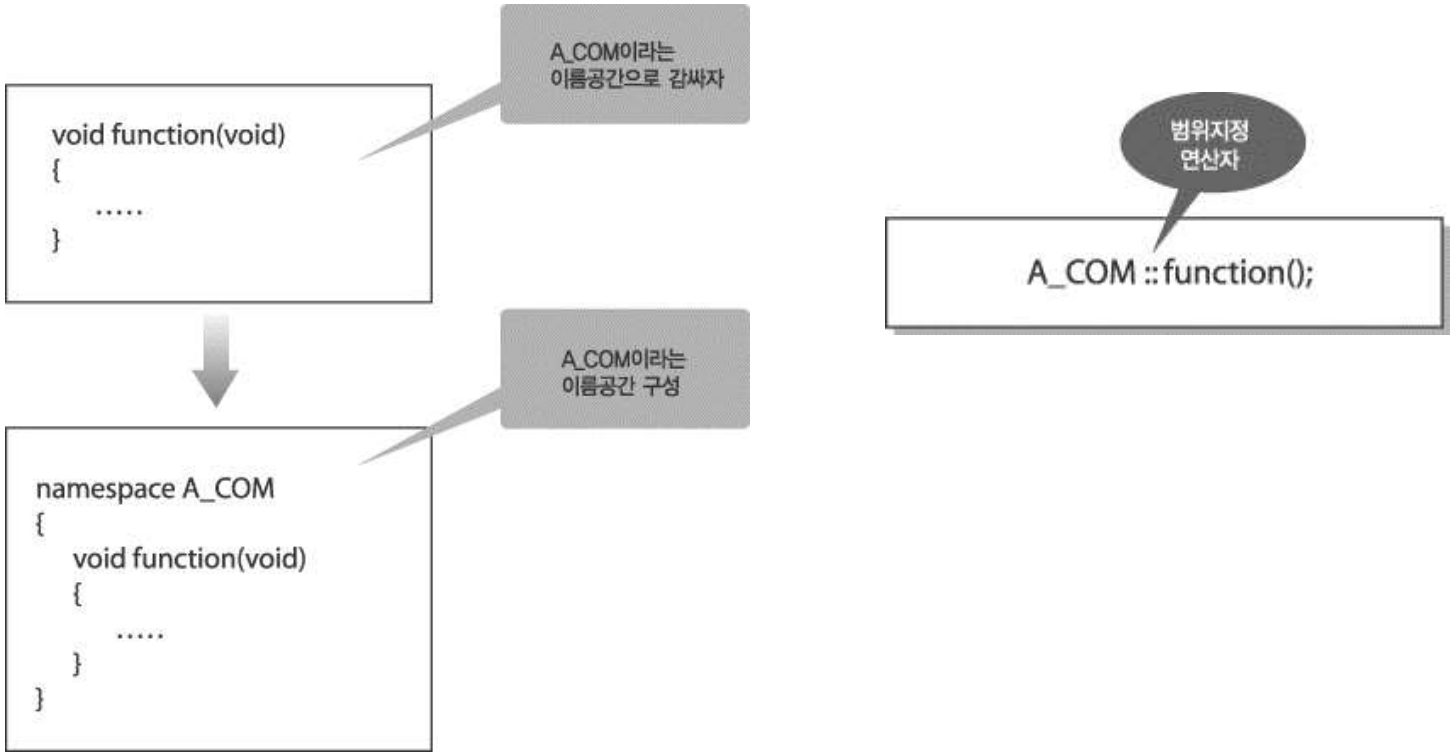
```
#include <iostream>

void function(void)
{
    std::cout<<"A.com에서 정의한 함수"<<std::endl;
}

void function(void)
{
    std::cout<<"B.com에서 정의한 함수"<<std::endl;
}

int main(void)
{
    function();
    return 0;
}
```

이름공간(namespace)

```
#include <iostream>

Namespace A_COM
{
      void function(void)
      {
         std::cout<<"A.com에서 정의한 함수"<<std::endl;
      }
}

 Namespace B_COM
{

      void function(void)
      {
         std::cout<<"B.com에서 정의한 함수"<<std::endl;
      }
 }

int main(void)
{
   A_COM::function();
   B_COM::function()
   return 0;
}
```

이름공간(namespace)

- 이름 공간의 적용



이름공간(namespace)

- 아하! std란 namespace!

그림 1-6

이름공간(namespace)

- 편의를 위한 using 선언!
  - using1.cpp, HelloWorld3.cpp
  - using2.cpp, HelloWorld4.cpp

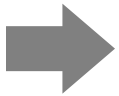
❖ using A_COM::function;

❖ using namespace A_COM;

이름공간(namespace)

- 범위 지정 연산자 기반 전역 변수 접근

```
int val=100;

int main(void)
{
  int val=100;
  val+=1;
  return 0;
}
```



```
int val=100;

int main(void)
{
  int val=100;
  ::val+=1;
  return 0;
}
```

표준 라이브러리 이름공간

- C++ 컴파일러가 제공하는 표준 라이브러리의 함수, 변수,클래스 등은 std 이름공간 영역으로 지정
  - 새 C++에서는 .h 확장자 생략하고 이름공간 명시

```
#include
     <iostream.h>

main()
{

     …
     cin >> data;
     cout << sum;
     …
}
```

```
#include <iostream>

main()
{
    using std::cin;
      …
     cin >> data;
     std::cout << sum;
     …
}
```

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

main()
{

     …
     cin >> data;
     cout << sum;
     …
}
```

변수의 범위 지정 연산자

- C 언어에서는 자동변수(지역변수)와 외부변수(전역변수)가 같은 이름으로 선언되어 사용되었을 때에는 자동변수의 값이 참조

- C++ 언어에서는 이와 같은 경우 변수 앞에 :: 연산자를 해당변수 앞에 기술하면 외부변수를 참조하는 것이 가능

범위 지정 연산자 사용 예

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

int a = 88;

int main(void)
{
   int a = 30;
   {
      int a = 10;
      cout << "1-1 a = " << a << "₩n";          // a = 10 참조
      cout << "1-2 ::a = " << ::a << "₩n";      // a = 88 참조
      {
         int a = 20;
         cout << "2-1 a = " << a << "₩n";       // a = 20 참조
         cout << "2-2 ::a = " << ::a << "₩n";   // a = 88 참조
      }
   }
   cout << "3-1 a = " << a << "₩n";             // a = 30 참조
   cout << "3-2 ::a = " << ::a << "₩n";         // a = 88 참조

   return 0;
}
```

1-1 a = 10
1-2 ::a = 88
2-1 a = 20
2-2 ::a = 88
3-1 a = 30
3-2 ::a = 88

:: 연산자는 블럭의 위치에 관계없이 외부변수를 참조한다.

은행계좌 관리 프로그램

- 요구기능
  - 계좌개설: MakeAccount()
  - 입금: Deposit()
  - 출금: Withdraw()
  - 전체 고객 잔액 조회: Inquire()
- 자료구조

```
typedef struct _Account
{
  int id;       // 계좌번호
  int balance; // 잔액
  char name[20]; // 고객 이름
} Account;
Account pArray[100];
index=0;    // 저장된 Account 수
```

은행계좌 관리 프로그램 1

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

const int NAME_LEN = 20;
typedef struct _Account
{
	int id;
	int balance;
	char name[NAME_LEN];
} Account;
Account pArray[100];
int index = 0;

void PrintMenu();
void MakeAccount();
void Deposit();
void Withdraw();
void Inquire();

enum{MAKE = 1, DEPOSIT, WITHDRAW, INQUIRE, EXIT};

int main(void)
{
    int choice;

	while(1)
	{
	  PrintMenu();
	  cout<<" 선택 : ";
	  cin>>choice;

	switch(choice)
	  {
	    case MAKE :
		MakeAccount();
		break;
	    case DEPOSIT :
		Deposit();
		break;
	    case WITHDRAW :
		Withdraw();
		break;
	    case INQUIRE :
		Inquire();
		break;
	    case EXIT :
		return 0;
	    default :
		cout<<" Illegal selection.." <<endl;
		break;
	    }
	  }
	  return 0;
}

void PrintMenu()    // 메뉴 출력
{
	cout<<" ---Menu------ "<<endl;
	cout<<" 1. 계좌 개설 "<<endl;
	cout<<" 2. 입금 "<<endl;
	cout<<" 3. 출금 "<<endl;
	cout<<" 4. 잔액 조회 "<<endl;
	cout<<" 5. 프로그램 종료 "<<endl;
}
```

은행계좌 관리 프로그램 2

```
void MakeAccount()                  // 계좌 개설
{
        int id;
        char name[NAME_LEN];
        int balance;

        cout<<" 계좌 개설------- "<<endl;
        cout<<" 계좌 ID : ";  cin>>id;
        cout<<" 이 름 : ";    cin>>name;
        cout<<" 입금액 : ";   cin>>balance;

        pArray[index].id = id;
        pArray[index].balance = balance;
        strcpy(pArray[index].name, name);

        index++;
}
void Deposit()       //입금
{
        int money;
        int id;          //찾고자 하는 ID
        cout<<" 계좌 ID : ";          cin>>id;
        cout<<" 입금액 : ";                                  cin>>money;
        for(int i = 0; i<index; i++)
        {
           if(pArray[i].id == id)
           {
                   pArray[i].balance += money;
                   cout<<" 입금 완료 "<<endl;
                   return;
           }
        }
        cout<<" 유효하지 않은 ID입니다. "<<endl;
}
```

```
void Withdraw()     // 출   금
{
        int money;
        int id;          // 찾고자 하는 ID

        cout<<" 계좌ID : ";          cin>>id;
        cout<<" 출금액 : ";          cin>>money;

        for(int i = 0; i<index; i++)
        {
           if(pArray[i].id == id)
           {
                   if(pArray[i].balance < money)
                   {
                          cout<<"잔액 부족 "<<endl;
                          return;
                   }
                          pArray[i].balance -= money;
                          cout<<" 출금 완료 "<<endl;
                          return;
           }
        }
        cout<<" 유효하지 않은 ID입니다."<<endl;
}

void Inquire()     // 전체 고객의 정보(잔액) 조회
{
        for(int i=0; i<index; i++)
        {
                   cout<<" 계좌 ID : "<<pArray[i].id<<endl;
                   cout<<" 이   름 : "<<pArray[i].name<<endl;
                   cout<<" 자   액 : "<<pArray[i].balance<<endl;
        }
}
```

Q&A