

# Data Structures

## Lab # 02

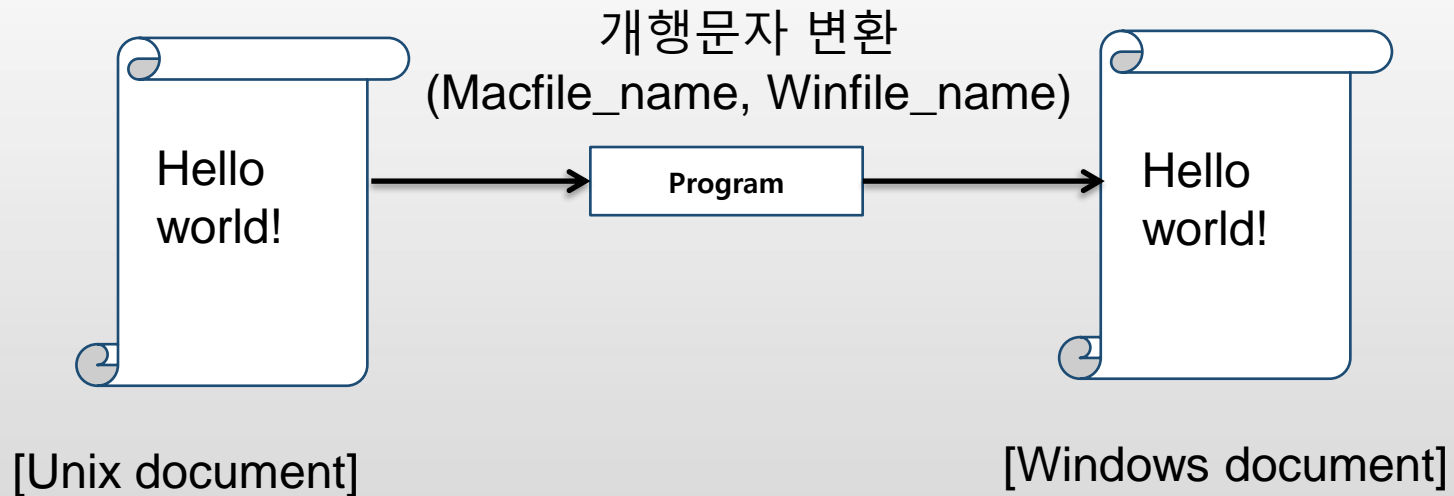


1. **Unix to Dos**
2. **Exercise 9 (한글 교재 9)**

# 1. Unix to Dos

## ■ 문제

- ❖ Unix에서 생성한 파일 형태를 MSDOS(Windows)에서 생성한 파일 형태로 변환하는 프로그램을 작성하여라.
- ❖ 프로그램의 첫번째 argument로 MAC 파일 이름을 입력받고 두번째 argument로 출력할 WINDOWS 파일 이름을 입력받는다



## ■ 개행 문자 ?

- ❖ 텍스트의 한줄이 끝남을 표시하는 문자 또는 문자열
- ❖ 새줄문자(newline), EOL(end-of-line)과 같은 뜻

## ■ 줄바꿈에 사용되는 특수 문자들

- ❖ LF(line feed) : ASCII -> 0x0A
- ❖ CR(Carriage Return) : ASCII -> 0x0D


## ■ 운영 체제 별 개행 문자 표현 방법

- ❖ Windows : CR LF (2bytes)
- ❖ Unix/Linux : LF (1bytes)
- ❖ Apple, Mac : CR (1bytes)

# 1-help slides (1/5)

## ■ c/c++언어 에서의 줄바꿈 문자 : 'Wn'

❖ Ex) `cout << "HelloWnworld";`    `cout << "Hello" << endl << "world";`

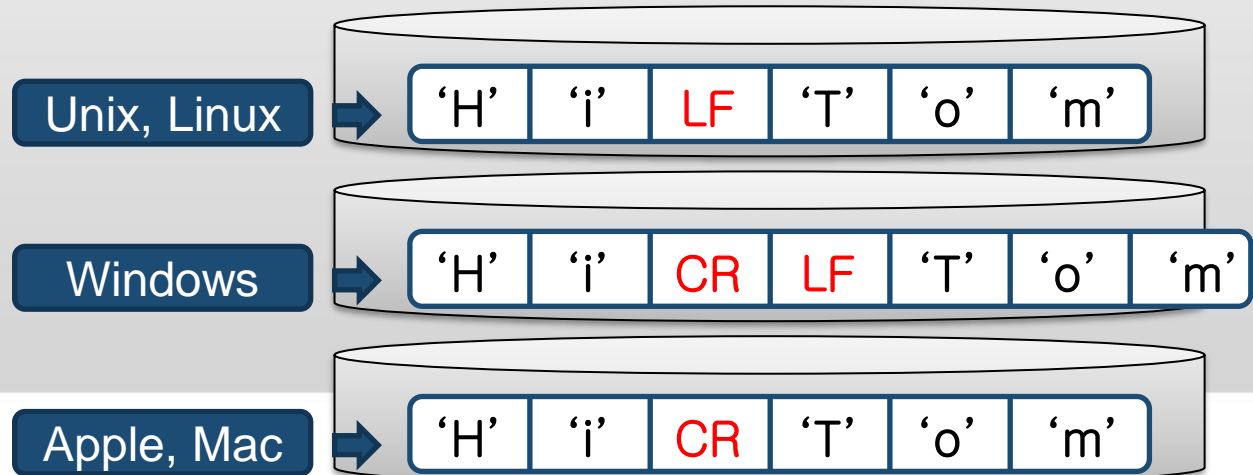


Hello  
world

[화면 출력]

## ■ 운영체제 별 개행문자 표현의 예

❖ Ex) `char str[1024] = {"HiWnTom"};`



## ■ Command-line argument

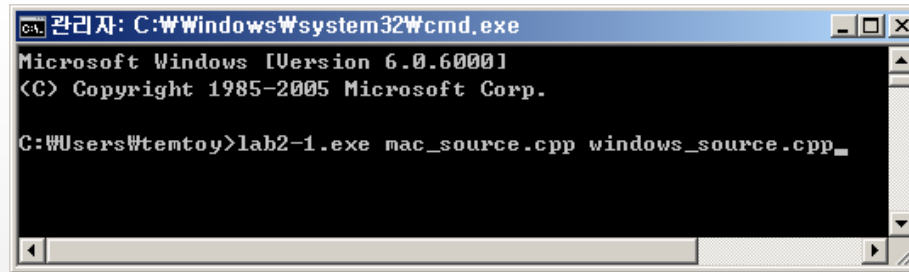
- ❖ main() 함수가 파라미터를 가질 때, 이 파라미터를 command-line argument라고 함
- ❖ 이점 : 프로그램 외부에서 값을 설정해줄 수 있음

## ■ Arguments를 프로그램에 지정해주는 경우 프로그램 내부에서 사용할 변수의 값이 변경되더라도 다시 컴파일 할 필요가 없음

- ❖ 사용자에게 콘솔로 입력받아도 가능하지만 프로그램을 반복적으로 다르게 설정하여 구동할때나 자동으로 처리하고 싶은 경우에 Arguments로 하는 방법이 더 효율적임
  - Example ) 프로그램에서 참조하는 파일의 경로가 변경된 경우,  
설정된 옵션에 따라서 프로그램을 다르게 구동하고 싶은 경우

## ■ Arguments를 적용한 프로그램의 예

- ❖ 프로그램의 이름이 lab2-1.exe 이고 첫번째 argument를 변환할 mac파일, 두번째 argument를 windows파일로 설정한 예시



```
관리자: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.0.6000]
(C) Copyright 1985-2005 Microsoft Corp.

C:\Users\temtoy>lab2-1.exe mac_source.cpp windows_source.cpp
```

## ■ Argument를 프로그램에서 사용하는 방법

- ❖ 실습 자료 뒷 부분의 부록에 있는 Command line argument 사용법 참조

## ■ 작성할 소스코드의 예시)

- ❖ 파일 입출력 객체들을 이용하여 교재의 샘플소스를 읽어 윈도우로 변환함
- ❖ ifstream : 파일로부터 데이터를 읽어오기 위한 클래스
- ❖ ofstream : 데이터를 파일에 저장하기 위한 클래스

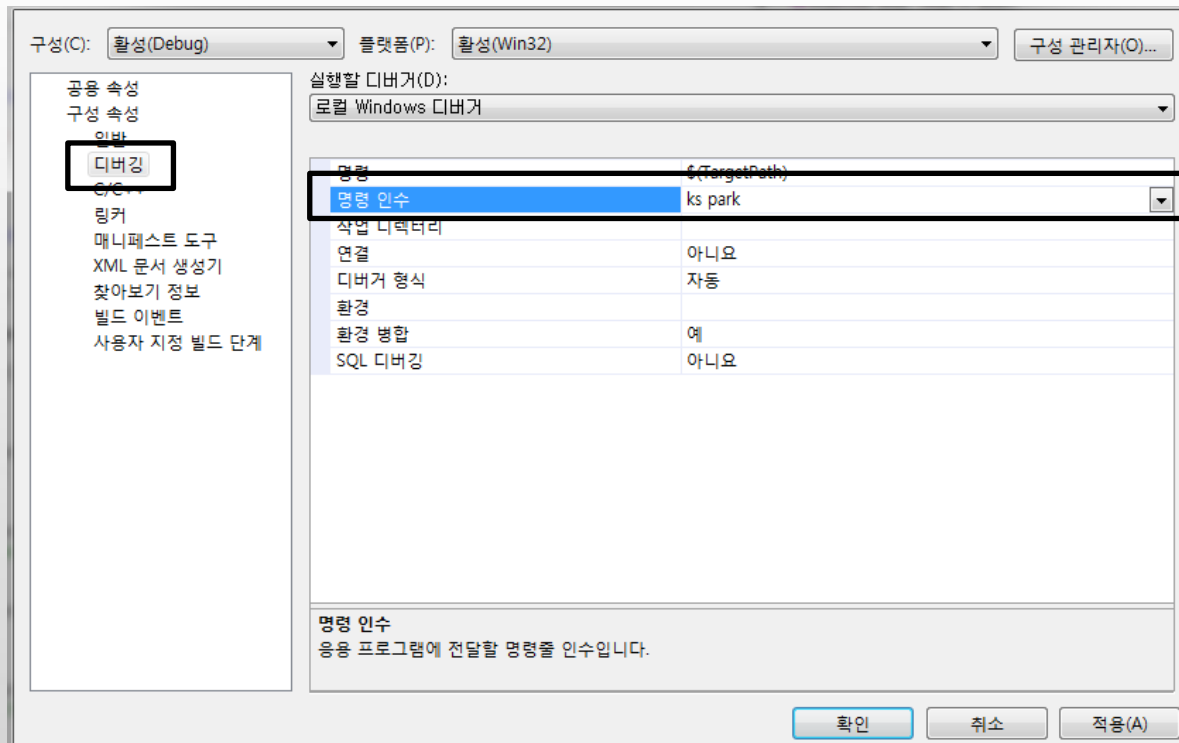
```
#include <fstream>
// myIn.txt 파일에서 첫 문자 읽어서 myOut.txt에 기록
int main()
{
    // 입력파일이름과 출력파일 이름은 argument를 이용하여 받을수 있도록 변경할 것
    ifstream input_file("myIn.txt");
    ofstream output_file("myOut.txt");

    char ch;
    while( ! _____ ) //파일의 끝이 아닐때까지 루프를 반복함
    {
        // 공백문자 여부 상관 없이 한 문자(1byte) 만큼 읽음
        input_file.get(ch);
        if( _____ ) break; // 입력 파일 스트림에 에러비트가 존재한다면 루프를 빠져나감
        if( _____ ) // ch가 LF ('\n') 이라면
            output_file << '\n';
        else
            output_file<< ch;
    }
    inputfile.close();
    output_file.close();
}
```



## ■ 테스트

- ❖ Project (프로젝트) -> Setting (속성)
- ❖ 이곳에서 미리 인수를 입력해 두면 Debug나 실행시 자동으로 프로그램에 인수(argument)가 입력되어 테스트가 편하다.



argv[1] → “ks”  
argv[2] → “park”

# Redefinition problem

- 같은 헤더파일을 여러 개의 소스파일에서 include 시키는 경우 재정의 문제가 발생함
  - ❖ 컴파일러는 같은 문장이 여러 번 써져 있는 것으로 인식되므로 재정의한다고 판단함
- 예시 )
  - ❖ 과제에서 주어진 ItemType.h, ItemType.cpp, sorted.h, sorted.cpp 4개의 파일을 프로젝트에 추가하고 main.cpp에서 #include "sorted.h"와 #include "ItemType.h"를 추가하는 경우 ItemType.h가 sorted.h와 main.cpp에서 두 번 선언되었기 때문에 에러를 발생함
- 구문
  - ❖ 중복으로 선언되는 코드에 다음의 구문을 위 아래에 추가함
  - ❖ **#ifndef macro\_name** // 주로 macro\_name은 파일의 이름을 대문자로 설정함  
#define macro\_name  
... (source code)  
# endif

## ■ 컴파일 중복 방지를 위한 다른 구문

### ❖ **#pragma once** (source code)

- 처음 한번만 컴파일하고 그 후 동일한 파일의 경우 컴파일 하지 않음

## ■ **#pragma once vs #ifndef**

### ❖ **#ifndef**

- 전처리기 지시자(Preprocessor directive)
- 모든 컴파일러에서 동작
- define 여부를 여러 번 체크하게 될 가능성이 존재해서 #pragma once보다 느림

### ❖ **#pragma once**

- 처음 한번만 컴파일
- 컴파일러 지시자(Compiler directive)
- 특정 컴파일러에서만 동작

# Redefinition problem example

```
#ifndef ITEMYPE_H
#define ITEMYPE_H

// The following declarations and definitions go into file
// ItemType.h.

#include <fstream>
const int MAX_ITEMS = 5;
enum RelationType {LESS, GREATER, EQUAL};

class ItemType
{
public:
    ItemType();
    RelationType ComparedTo(ItemType) const;
    void Print(std::ostream&) const;
    void Initialize(int number);
private:
    int value;
};

#endif
```

## 2. Exercise 9 (한글 교재 9)

### ■ 문 제

- ❖ 비정렬 리스트 ADT에 대한 명세는 삭제될 요소가 리스트에 있다는 것을 말한다.
  - a. Deleteltem에 대한 명세를 재작성해서 삭제될 요소가 리스트 내에 없으면 리스트는 변하지 않게 하여라.
  - b. (a)에 기술한 것 같이 Deleteltem을 구현하여라.
  - c. Deleteltem에 대한 명세를 재작성해서 삭제될 요소가 리스트에 있으면 삭제될 모든 요소를 삭제하여라.
  - d. (c)에 기술한 것 같이 Deleteltem을 구현하여라.
- ❖ 이번 문제는 "unsorted.h" 및 "unsorted.cpp"에 Deleteltem\_a(ItemType item), Deleteltem\_c(ItemType item) 와 같이 함수를 추가하여 작성하세요.

### ■ 경 로

- ❖ 문제에 필요한 샘플 소스 코드의 경로 입니다.
- ❖ "WlabplusWLab, C++ 3rdWChapter3WUnsorted"

ItemType.h  
ItemType.cpp  
unsorted.h  
unsorted.cpp



4개의 파일을 사용합니다.

## ■ 책에서 구현된 deleteItem의 명세

```
void DeleteItem(ItemType item);  
// Function: Deletes the element whose key matches item's key.  
// Pre: List has been initialized.  
//      Key member of item is initialized.  
//      One and only one element in list has a key matching item's key.  
// Post: No element in list has a key matching item's key.
```

```
void UnsortedType::DeleteItem(ItemType item)  
// Pre: item's key has been initialized.  
//      An element in the list has a key that matches item's.  
// Post: No element in the list has a key that matches item's.  
{  
    int location = 0;  
  
    while (item.ComparedTo(info[location]) != EQUAL)  
        location++;  
  
    info[location] = info[length - 1];  
    length--;  
}
```

- ❖ 리스트에 삭제할 아이템이 하나만 있다고 가정하고 있음
- ❖ 만약 아이템이 리스트에 존재하지 않을 경우, location이 리스트내의 원소 개수보다 커지므로 사용하지 않는 메모리 영역에 접근하게 됨 (문제 a,b에 해당)
- ❖ 만약 아이템이 중복된다면 1개만 삭제되는 경우가 발생함 (문제 c,d에 해당)

- Deleteltem에 대한 명세를 재작성해서 삭제될 요소가 리스트 내에 없으면 리스트는 변하지 않게 하여라.
- (a)에 기술한 것 같이 Deleteltem을 구현하여라.

### ■ 교과서의 현재 알고리즘의 경우, 삭제할 아이템이 없으면 배열의 끝을 지나 무한대로 반복되는 에러가 나게 됩니다.

❖ 검색횟수를 제한해야 함.

### ■ 아이템을 찾을 때까지 하나씩 체크한 후, 아이템을 찾으면 삭제를 함. 삭제한 후에는 바로 리턴 가능.

```
//function      : 작성할 것
//precondition  : 작성할 것
//postcondition : 작성할 것
void List::Deleteltem(ItemType item)
{
    bool deleted = false;
    for (int i = 0; i < length && !deleted; i++)
    {
        if(item.CompareTo(_____) == EQUAL){
            // 해당 아이템을 삭제
            // deleted 플래그 값 설정
        }
    }
}
```

- c. DeleteItem에 대한 명세를 재작성해서 삭제될 요소가 리스트에 있으면 삭제될 모든 요소를 삭제하여라.
- d. (c)에 기술한 것 같이 DeleteItem을 구현하여라.

## ■ 검색을 가지고 있는 아이템 개수만큼(끝까지) 합니다.

- ❖ 삭제할 아이템을 찾아도 계속 진행(여러 개가 있을 수 있기 때문)

## ■ 구현의 예시)

```
//function      : 작성할것
//precondition  : 작성할것
//postcondition : 작성할것
void List::DeleteItem(ItemType item)
{
    int l = 0;
    while (l < length)
    {
        if(item.CompareTo(_____) == EQUAL){
            // 해당 아이템을 삭제
            // 삭제 후 현재 i 위치의 값이 변경되었으므로 나중에 다시 확인 하도록
            // i 값 변경 하지 않음 , 추가로 length값은 감소
        }
        else {
            // 다음 아이템을 보기 위해서 i값 변경
        }
        l++;
    }
}
```





**부록**

# **Command-line Arguments: 인수를 갖는 프로그램 작성법**



# Command-line Arguments

- `main()` 함수가 파라미터를 가질 때, 이 파라미터를 *command-line argument*라고 함
- *command-line argument*는 프로그램이 실행될 때 운영체제가 받아서 `main()` 함수로 전달함



# Argument to main

```
int main ()  
{
```

```
} //main
```

**main : without command-  
line arguments**

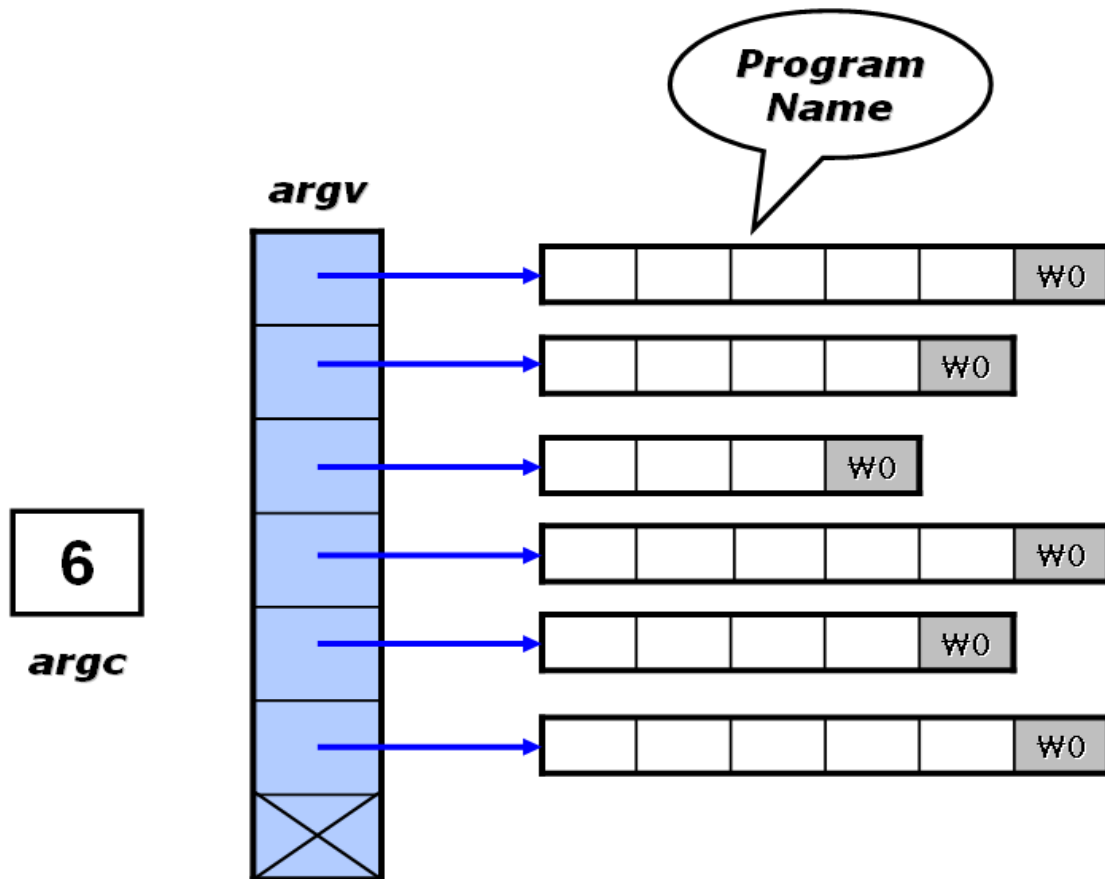
```
int main (int argc, char* argv[])  
{
```

```
} //main
```

**main : with command-  
line arguments**



# argc and argv





# 수행 방법: MSDOS 창 이용

```
C:\>C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\User>cd "바탕 화면"

C:\Documents and Settings\User\바탕 화면>cd Project

C:\Documents and Settings\User\바탕 화면\Project>cd test

C:\Documents and Settings\User\바탕 화면\Project\test>cd Debug

C:\Documents and Settings\User\바탕 화면\Project\test\Debug>test a b c d
파라메터 개수 = 5
0-th argement = test
1-th argement = a
2-th argement = b
3-th argement = c
4-th argement = d

C:\Documents and Settings\User\바탕 화면\Project\test\Debug>_
```

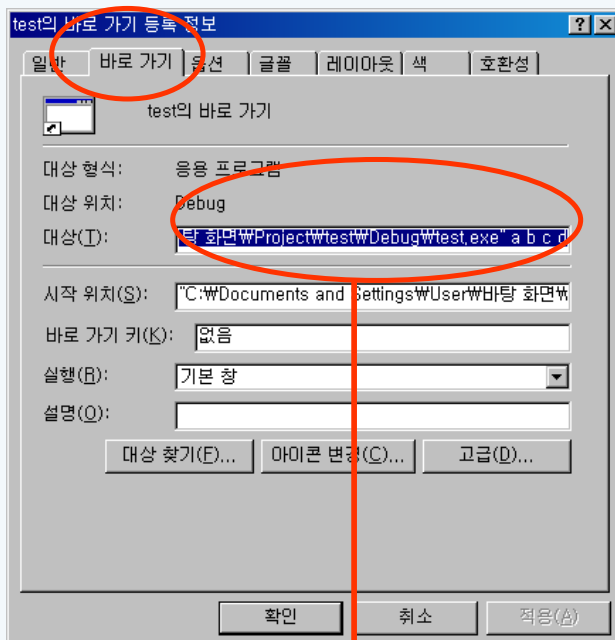
**argument**

입력



# 수행 방법: 실행 파일의 속성 이용

- 실행파일의 “바로 가기”를 만들어서 그 “바로 가기 파일”의 속성에서 실행파일명 뒤쪽으로 **argument** 입력



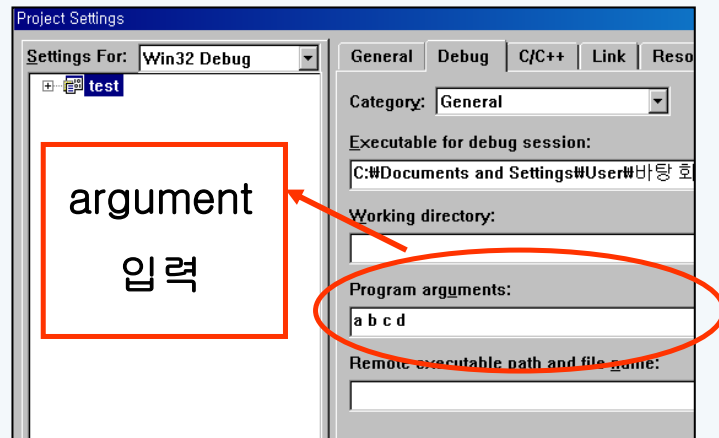
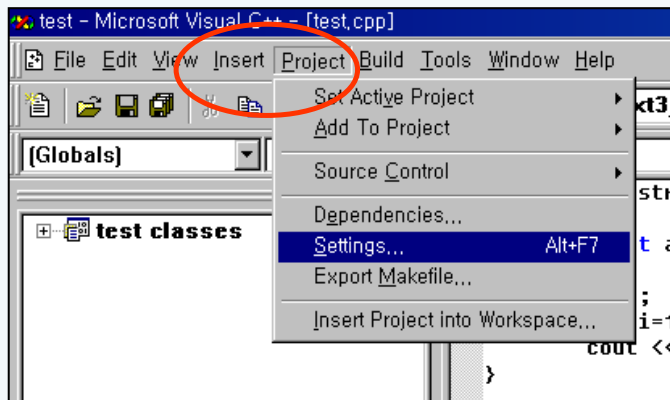
"C:\Documents and Settings\User\바탕 화면\Project\test\Debug\test.exe" a b c d

```
C:\> test의 바로 가기
파라미터 개수 = 5
0-th argement = C:\Documents and Settings\User\바탕 화면\Project\test\Debug\test.exe
1-th argement = a
2-th argement = b
3-th argement = c
4-th argement = d
```

파일명의 끝에  
**argument** 입력



# 수행 방법 : Visual studio에서 실행



```
C:\WINDOWS\SYSTEM32\cmd.exe /c "C:\DOCUMENTS AND SETTINGS\USER\바탕 화면\PROJECT\test\Debug\test.exe" a b c d
파라미터 개수 = 5
0-th argument = C:\DOCUMENTS AND SETTINGS\USER\바탕 화면\PROJECT\test\Debug\test.exe
1-th argument = a
2-th argument = b
3-th argument = c
4-th argument = d
Press any key to continue.
```





# 예제소스

```
#include <iostream.h>

int main(int argc, char *argv[])
{
    cout << "파라미터 개수 = " << argc << endl;

    // argv[0]은 실행파일 이름이 들어감을 눈 여겨 볼 것
    for(int i=0; i<argc; i++)
        cout << i << "-th argement = " << argv[i] << endl;

    return 0;
}
```