Chapter 04. Memory functions and classes

목차

학습목표

- Find out about the concept of C ++ functions..
- Study for the return type, the function name, parameters for defining functions.
- Learn about the declaration of the function (circular definition).
- Learn about storage classes.
- Understand the difference between local and global variables.

01 Understanding the function

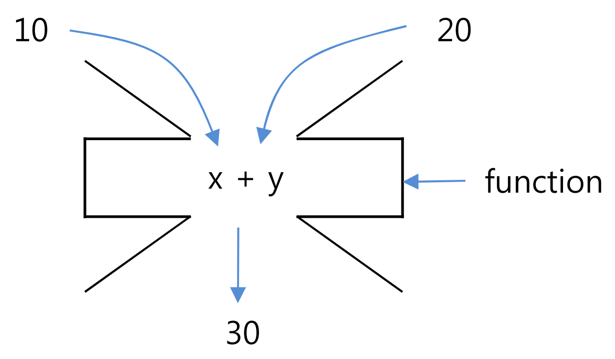
The basic concept of the for loop and the function

- Function is a small program unit of the enclosed instructions for processing a specific function. When necessary, make a call to a code block that is frequently used in the program separately once a function unit that processes a function

If the function finds the sum of a natural number	If you define a function that finds the sum of the natural numbers

```
void main()
{
  int total;
  int a=10;
  int i;

  total=0; // 반드시 초기화해야 한다.
  for(i=1; i<=a; i++) {
    total+=i;
}
  cout<<"total ="<< total<<endl;

  total=0; // 반드시 초기화해야 한다.
  for(i=1; i<=100; i++) {
    total+=i;
}
  cout<<"total ="<< total<<endl;

  total=0; // 반드시 초기화해야 한다.
  for(i=1; i<=5; i++) {
    total+=i;
}
  cout<<"total ="<< total<<endl;
}
```
	```
	int sum(int n)
	{
	  int total=0; // 반드시 초기화해야 한다.
	  int i;
	  for(i=1; i<=n; i++) {
	    total+=i;
	}
	  return total;
	}

	void main()
	{
	  int a=10;
	  cout<<"total ="<<sum(a)<<endl;
	  cout<<"total ="<<sum(100)<<endl;
	  cout<<"total ="<<sum(5)<<endl;
	}
	```

01 Understanding the function



[Picture 4–1] The operation principle of the function

Library functions and user-defined functions

❶ **Library Function:** A function that provides created in the place that created the compiler. Because it is already defined, simply bring in use. Note, that it should be used after the inclusion (include) the specific header file for each function.

❷ **User defined functions:** a function the user has created himself, as needed to create the program.

The advantage of using a function

❶ What you need, you make a run repeatedly as a function you can use to call whenever needed.

❷ Since the program easier to read and modular (blocked), easy to debug and edit.

❸ Easy to learn the functions and structure of the program at a glance.

❹ Can re-use in other programs.

02 Definition and call functions

When you use a function that must exist three kinds

❶ Definition of a function: should the definition of a function that describes the behavior that you want to contain any details on how the function.

❷ function calls: You should call the function defined where the function is not required unless you run a call.

❸ Declaration of the function: You can invoke the function is called before and then. if the function definition must declare the function to appear (circular definition).

The basic format of the function

The function default format	Function Example

Data type function name
(parameter list)
{
Declaration of variables;
    sentence;
    return Results;
}
	int sum(int x, int z)
	{
	    int add;
	    add=x+y;
	    return add;
	}

02 Definition and call functions

❶ Datatype: indicates the data type of the return value specified where the function is called, in accordance with the following rules.

- If you do not specify a return value type void.
- If it declares a return value gives the same result in fewer types and then return.

❷ Function name: Create the same as the identifier written rules, it is a good idea to also report it meaningfully named only possible function name so that you know whether the function that performs what function.

❸ Parameter list: indicates whether a certain parameter (argument) to the input of the incoming function. What usually indicates whether the data type of the parameter and whether any form (number of arguments and order).

Basic types of functions

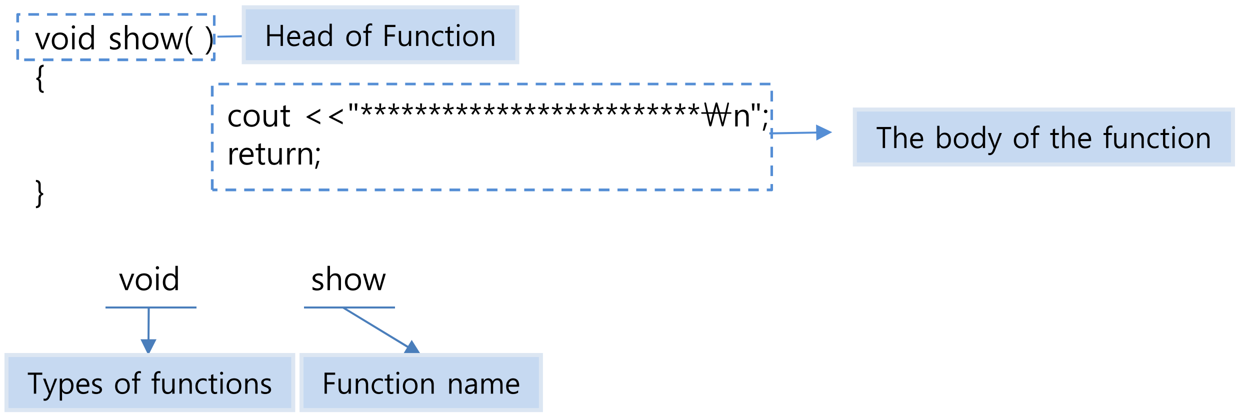
❶ Parameters and return values are not all functions

❷ The only parameters that can function without a return value

❸ Parameters and return value of all functions

02 Definition and call functions

Parameters and return values are not all functions



```
void show( )
{
            cout <<"*************************₩n";
            return;
}
```

- Use void within () may specify a parameter is missing.

  void show(void)
- Must be called in the following forms are described in the logic function, the body is carried out.

  show( );

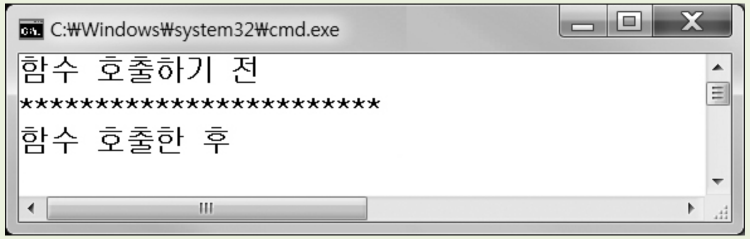
User-defined function is the main function and other points

❶ main function name must be the main function name, but in a user-defined function, makes it adhere to the rules for creating a variable name.

❷ As the program runs, the main function is automatic, user-defined function must call the programmer must explicitly.

Example 4-1. Returns parameters are also not to create a function (04_01.cpp)

```
#include <iostream>
using namespace std;
void show()
{
cout << "************************₩n";
return;
}
void main()
{
cout << "함수 호출하기 전₩n";
show(); // 함수의 호출
cout << "함수 호출한 후₩n";
}
```

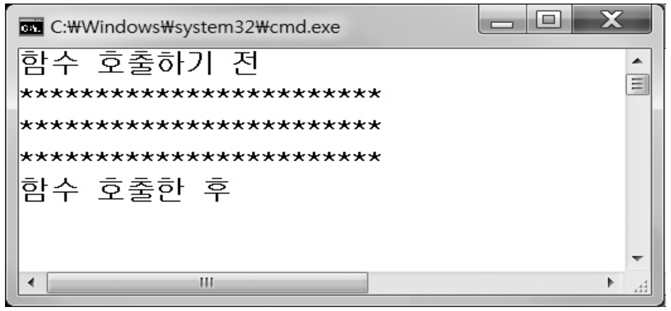


```
함수 호출하기 전
************************
함수 호출한 후
```

02 Definition and call functions

- Function can reduce the redundant code. Execution results are output from the line 3 where a call to the function, as follows.

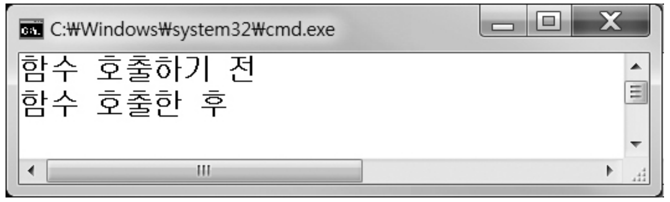
```
10   cout << "함수 호출하기 전₩n";
11   show( );
12   show( );
13   show( );
14   cout << "함수 호출한 후₩n";
```



```
함수 호출하기 전
*************************
*************************
*************************
함수 호출한 후
```

- **<u>Function can be defined once and multiple</u>** calls each time a function is needed can be useful functions. On the other hand, only without calling the function Once you have defined the content described in the user-defined function is never done.

```
08   void main()
09   {
10       cout << "함수 호출하기 전₩n";
11       // show( ); // 함수의 호출
12       cout << "함수 호출한 후₩n";
13   }
```



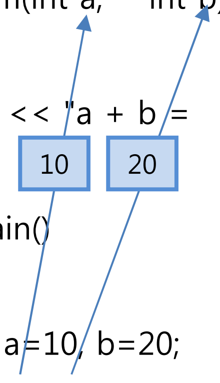
```
함수 호출하기 전
함수 호출한 후
```

02 Definition and call functions

The only parameters that can function without a return value

- Actual parameters: technical parameters that when you call a function variables
  - Type Parameter: Parameter is described to define a variable function

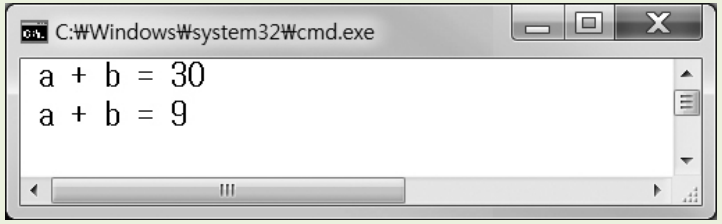
```
void sum(int a,    int b)          // a, b는 Type parameter
{
    cout << "a + b =    "<<a + b<<"₩n" ;
}
void main()
{
    int   a=10, b=20;
    sum(a,   b);                         // 변수 a, b는 실 매개변수
}
```



- Described in the above case, the actual parameter variable is transmitted values stored in the variables memory area not to be a transmission (call by value).

Example 4-2. Creating a function to obtain the sum of two numbers (04_02.cpp)

```
#include <iostream>
using namespace std;
void sum(int a, int b) // a, b는 형식 매개변수
{
cout << " a + b = "<< a + b<<"₩n" ;
}
void main()
{
int a=10, b=20;
sum(a, b); // 변수 a, b는 실 매개변수
sum(4, 5); // 상수(값) 4, 5 역시 실 매개변수
}
```


C:₩Windows₩system32₩cmd.exe

```
a + b = 30
a + b = 9
```

02 Definition and call functions

Parameters and return value of all functions

- If you want to return the results of this function uses the return statement

return문 기본 형식

```
return expression;
```

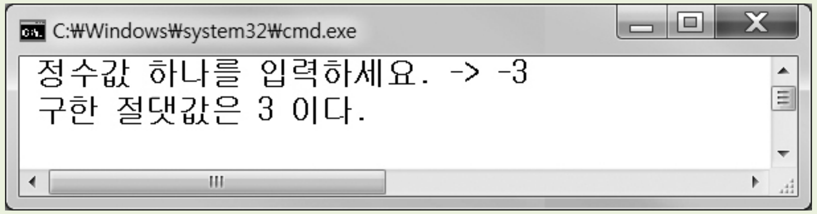
- Part of the equation above format may be used for constants, variables, formulas as follows:

```
return 1;
return a;
return (a);
return a+b;
```

- The return statement is used in two purposes.
  - ❶ Used to give back the results of calling up a function to execute the function.
  - ❷ Without going any further character in the current run time it is used to break out the function.

Example 4-3. Creating a function to obtain the absolute value (04_03.cpp)

```cpp
#include <iostream>
using namespace std;
int myAbs(int x)
{
    int y; // 절댓값을 저장할 변수
    if(x < 0) // 절댓값을 구하는 공식
        y = -x;
    else
        y = x;
    return y; // 구해진 결과값을 return문으로 반환한다.
}
void main()
{
    int a, result;
    cout << "₩n 정수값 하나를 입력하세요. -> ";
    cin >> a;
    result = myAbs(a); // 함수의 결과값을 변수 result에 대입한다.
    cout <<" 구한 절댓값은 " << result << " 이다. ₩n" ;
}
```



```
정수값 하나를 입력하세요. -> -3
구한 절댓값은 3 이다.
```

03 The declaration of a function

The declaration of functions (circular functions)

- Not yet defined (but are below) before calling the function to call in advance to be sure to have the head of the technology function. The declaration of a function uses the compiler tries to resolve the types and number of parameters when calling the function..

Function declaration format	For example, using a function declaration
Data type function name (parameter list);	int myAbs(int x);

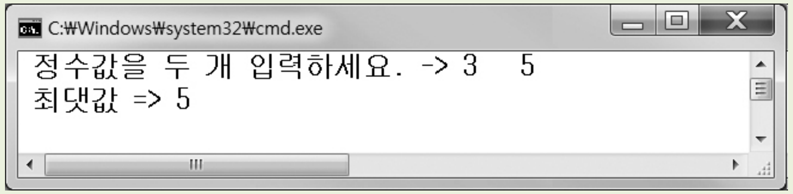
❶
```
#include<iostream>
using namespace std;
int max(int x, int y)
{
    return ( (x > y) ? x : y );
}
void main( )
{
    int a, b;
    cin >> a >> b;
    cout << max(a, b);
}
```

❷
```
#include<iostream>
using namespace std;
int max(int, int);
void main( )
{
    int a, b;
    cin >> a >> b;
    cout << max(a, b);
}
int max(int x, int y)
{
    return ( (x > y) ? x : y );
}
```

- ❶ defines a function on the main Functions And ❷ defined under the main function. If you call a function like ❷, you need to first declare the function before any function calls.

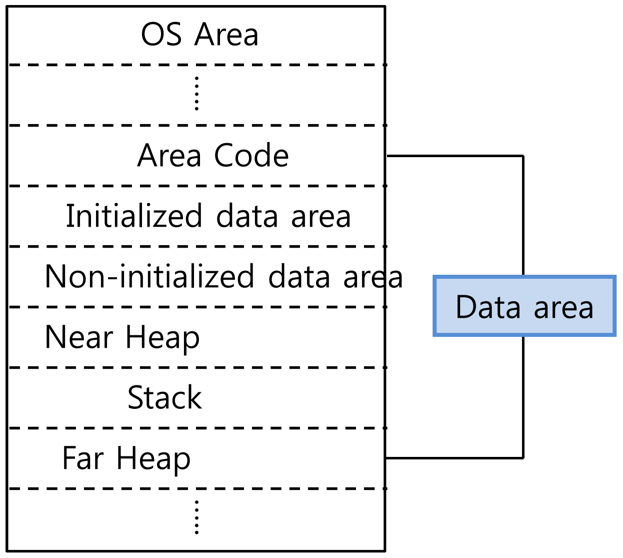
Example 4-4. Creating a function to obtain the maximum value (04_04.cpp)

```
#include <iostream>
using namespace std;
int max(int, int) ; // 함수의 원형
void main()
{
int a, b; // 정수형으로 변수 2개를 선언한다.
cout << "₩n 정수값을 두 개 입력하세요. => ";
cin >> a >> b;
// 함수의 선언
cout << " 최댓값 => " << max(a, b) <<"₩n";
}
int max(int x, int y) // 함수의 정의
{
return ( (x > y) ? x : y ); // return문에 조건 연산자
}
```



```
정수값을 두 개 입력하세요. -> 3    5
최댓값 => 5
```

04 The variable storage classes



[Picture 4–2] Random access memory (RAM) structure

- Where and how variables in the data area is subdivided storage class that determines whether secured.

[Table 4–1] Types of storage classes

division	Effective range	Survival	memory	Initialization Status
auto(자동변수)	My Block	Temporarily	stack	Garbage
extern(외부변수)	My program	Permanently	memory	The number 0
static(정적변수)	Inside: My Block External: module so	Permanently	memory	The number 0
register(레지스터 변수)	My Block	Temporarily	Register in the CPU	Garbage

04 The variable storage classes

- Survival
  - ❶ Temporarily: a variable has been created in the declared time is automatically extinguished when the block ends. Is destroyed and created in the program is repeated several times.
  - ❷ Persistent: are generated when the program starts, there is always the same storage space on the memory during program execution. Until the end of the variable when the program is completely extinguished.

- In general, the storage variable declaration class is omitted.

Variables declared default format	For example, using the variable declaration
Remember the Class data type the variable name;	**auto int a = 0;**

- Variable declarations can be divided into the following two types by the storage class.
  - ❶ Division by types: **it determines the type and size of storage space** that can not be omitted and stored in the variable value (int, char, float, etc.).
  - ❷ Division by the storage class: can be omitted, if recognized as an auto be omitted. **Determine the scope and nature of variables such as survival time** (auto, extern, static, etc.).

04 The variable storage classes

Auto variable

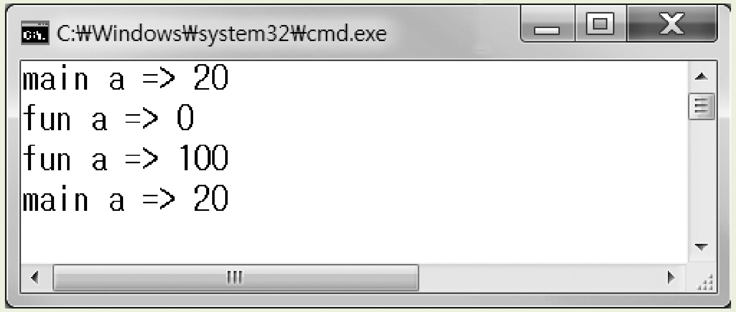
- Variable automatic (auto) is included in an automatic variable parameters as a function of a variable declared in a function definition part, it repeats the creation and destruction while the program is running.
- If the variable is declared memory allocation takes place, block out the outside surface of the variable memory is freed declared variables are unavailable.
- The most commonly used type is automatically variable, the variable will be omitted to automatically specifier the storage class called auto.

Local variable and Global variable

- Variable region: stands for the variables declared in the block (function), a variable range that can be used is limited to the inside of the block variable is declared.
- Global Variables: A variable that is declared outside of the function. Global variables are variables that can be used in all functions within the program. In addition, the external variable and its value is kept while the program is running. A global variable is initialized automatically (as if the data value 0).

Example 4-6. Learn the nature of global variables (04_06.cpp)

```
#include<iostream>
using namespace std;
void fun();
int a; // 전역변수 a
void main()
{
    int a = 20; // 지역변수 a
    cout<<"₩n main a => "<<a;
    fun();
    cout<<"₩n main a => "<<a<<"₩n";
}
void fun()
{
    cout<<"₩n fun a => "<<a;
    a=a+100;
    cout<<"₩n fun a => "<<a;
}
```



```
main a => 20
fun a => 0
fun a => 100
main a => 20
```

04 The variable storage classes

Static variable

- And put in front of a static variable name when you declare a variable is declared as static variables.
  - ❶ The variable is declared inside a block: Static variables as local variables.
  - ❷ Static variables as global variables: The variable is declared outside the block.

Static variables as local variables

[Table 4-2] Compare as a static variable automatic and local variables

Variable name	division	Effective range	Survival	Initialization Status	Whether the value of a variable
auto	Automatic variables	My Block	Temporarily	Garbage	Not keeping
static	Local variables, static variables	My Block	Permanentl y	The number 0	Maintained

Static variables as global variables

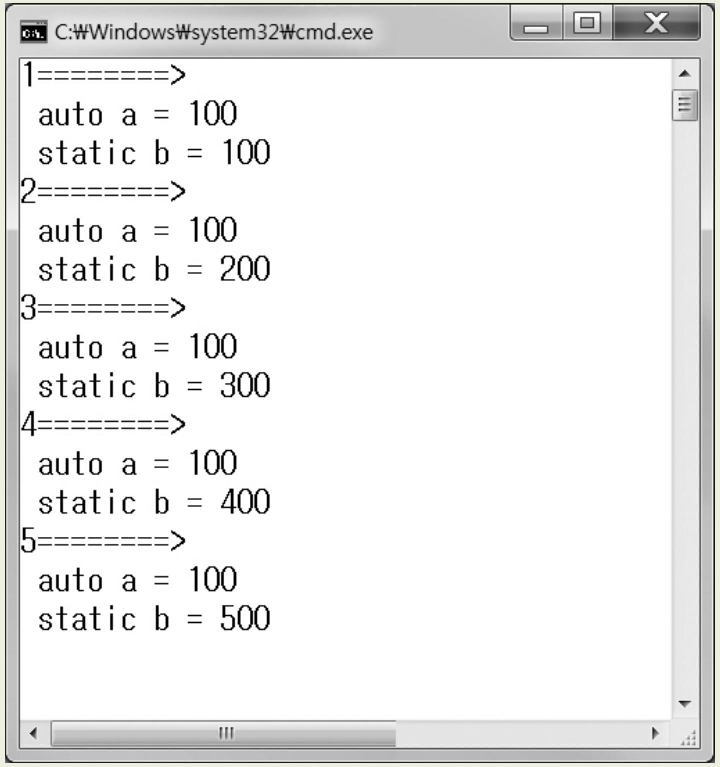
- The program is called Project created one collecting multiple files when creating the program. And it referred to the variables that can be used to share files within all the external factors tied to the project, and then paste used to declare a variable before using the extern to access external variables declared in another file.

External variables declared default format	Declaration of external variables Example
extern data type variable name;	extern int a;

- Static variables as global variables can be used only files that the variable is declared. In other words, if the global variable is available only in single file, attach the required static.

Example 4-7. The difference between static variables and local variables as automatic (04_07.cpp)

```cpp
#include <iostream>
using namespace std;
void sub();
void main()
{
    for(int i=1; i<=5; i++){
        cout<< i << "=========>₩n";
        sub();
    }
}
void sub()
{
    int a = 0;
    static int b = 0;
    a+=100;
    b+=100;
    cout<<" auto a = "<<a<<endl;
    cout<<" static b = "<<b<<endl;
}
```

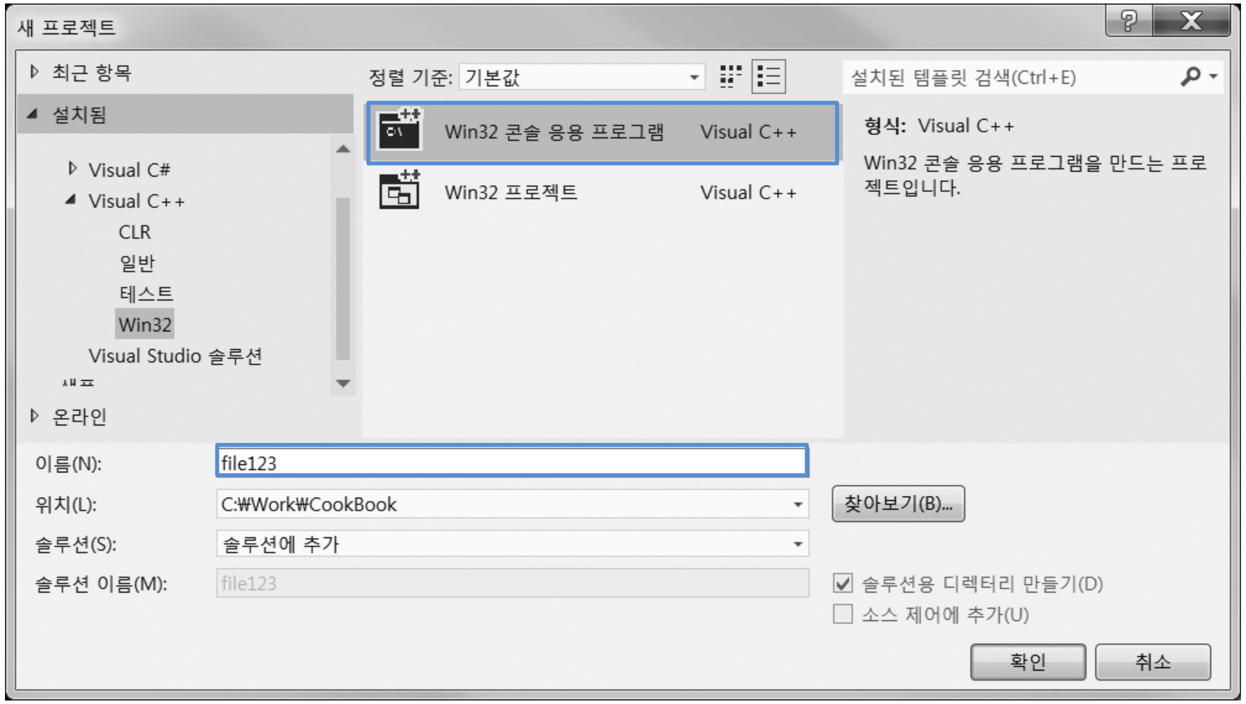

C:₩Windows₩system32₩cmd.exe

```
1========>
 auto a = 100
 static b = 100
2========>
 auto a = 100
 static b = 200
3========>
 auto a = 100
 static b = 300
4========>
 auto a = 100
 static b = 400
5========>
 auto a = 100
 static b = 500
```

04 The variable storage classes

Creating a project consisting of three source file

① Creating a new project



형식: Visual C++

Win32 콘솔 응용 프로그램을 만드는 프로젝트입니다.

이름(N): file123

위치(L): C:₩Work₩CookBook

솔루션(S): 솔루션에 추가

솔루션 이름(M): file123

☑ 솔루션용 디렉터리 만들기(D)

☐ 소스 제어에 추가(U)

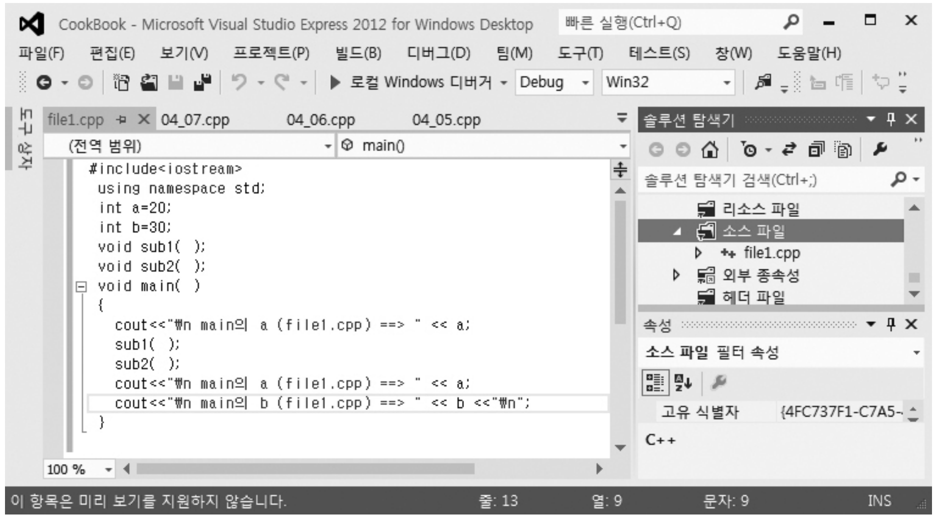
04 The variable storage classes

② Source Input : ‘file1.cpp’

```
#include<iostream>
using namespace std;
int a=20;
int b=30;
void sub1( );
void sub2( );
void main( )
{
    cout<<"₩n main의 a (file1.cpp) ==> " << a;
    sub1( );
    sub2( );
    cout<<"₩n main의 a (file1.cpp) ==> " << a;
    cout<<"₩n main의 b (file1.cpp) ==> " << b <<"₩n";
}
```

04 The variable storage classes

③ Create a new file



```
#include<iostream>
 using namespace std;
 int a=20;
 int b=30;
 void sub1( );
 void sub2( );
 void main( )
 {
   cout<<"₩n main의 a (file1.cpp) ==> " << a;
   sub1( );
   sub2( );
   cout<<"₩n main의 a (file1.cpp) ==> " << a;
   cout<<"₩n main의 b (file1.cpp) ==> " << b <<"₩n";
 }
```

CookBook - Microsoft Visual Studio Express 2012 for Windows Desktop
빠른 실행(Ctrl+Q)
파일(F) 편집(E) 보기(V) 프로젝트(P) 빌드(B) 디버그(D) 팀(M) 도구(T) 테스트(S) 창(W) 도움말(H)
로컬 Windows 디버거
Debug
Win32
도구 상자
file1.cpp
04_07.cpp
04_06.cpp
04_05.cpp
(전역 범위)
main()
100 %
솔루션 탐색기
솔루션 탐색기 검색(Ctrl+;)
리소스 파일
소스 파일
file1.cpp
외부 종속성
헤더 파일
속성
소스 파일 필터 속성
고유 식별자 {4FC737F1-C7A5-
C++
이 항목은 미리 보기를 지원하지 않습니다.
줄: 13
열: 9
문자: 9
INS

④ Used to bring external variables declared in another file: 'file2.cpp'

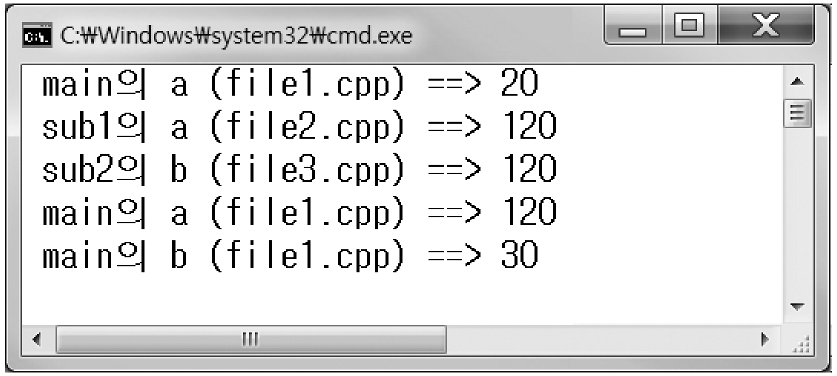
```
#include<iostream>
using namespace std;
extern int a;
void sub1( )
{
    a+=100;
    cout<<"₩n sub1의 a (file2.cpp) ==> " << a;
}
```

04 The variable storage classes

⑤ Declare global variables using only one file: 'file.cpp'

```
#include<iostream>
using namespace std;
static int b=20;
void sub2( )
{
    b+=100;
    cout<<"₩n sub2의 b (file3.cpp) ==> " << b;
}
```

⑥ The results



```
main의 a (file1.cpp) ==> 20
sub1의 a (file2.cpp) ==> 120
sub2의 b (file3.cpp) ==> 120
main의 a (file1.cpp) ==> 120
main의 b (file1.cpp) ==> 30
```

Homework

- **Chapter 4 Exercise: 7, 9, 10, 11, 13, 15**