

윈도우 프로그래밍

1. C++ 개요

2018. 3.9.
심미나 교수



목 차

- I. 객체지향프로그래밍
- II. C++ 시작하기
- III. C와 C++ 비교
- IV. 실습

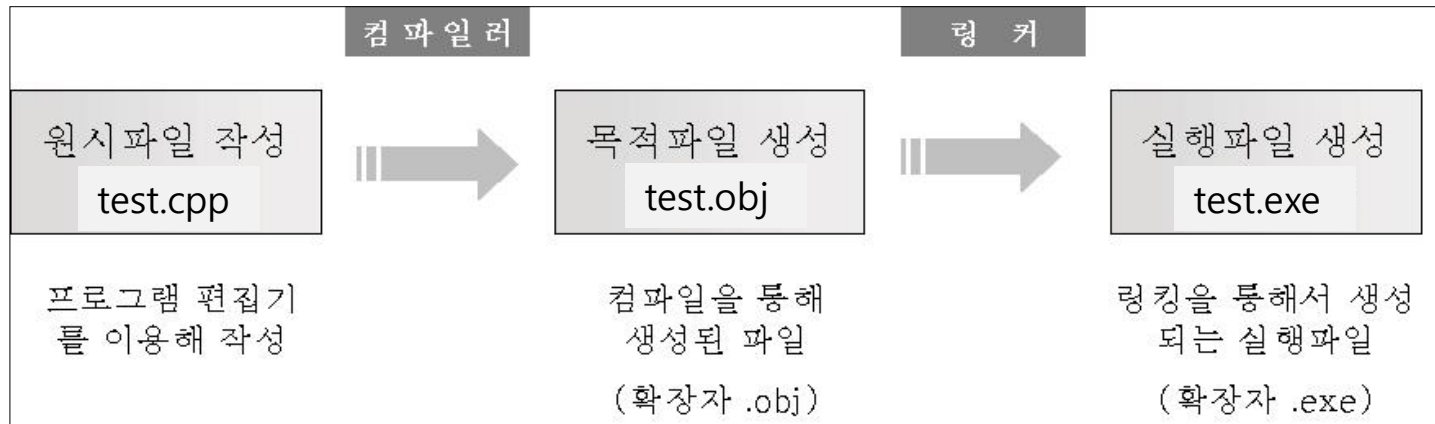
I. 객체지향프로그래밍

프로그래밍 개요



C++ 프로그래밍 과정 - 편집/컴파일/링크/실행

- C++ 도구(편집기)를 통해서 C++ 프로그램 소스를 작성
- 작성된 C++소스(.cpp)를 기계어로 바꾸는 과정인 컴파일을 수행하며, 컴파일을 통해서 목적파일(.obj) 생성(문법검증 및 기계어 변경)
- 링커통해 생성한 목적파일과 시스템상에 생성된 목적파일들을 연결, 실행파일(.exe) 생성(라이브러리 형식검사 및 실행파일 생성)

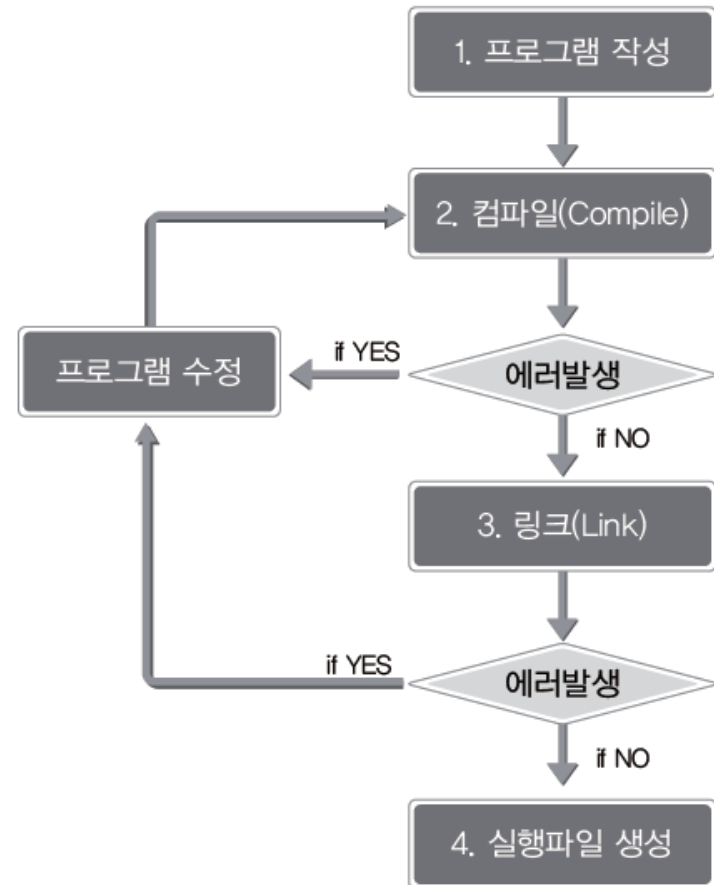


프로그래밍 개요



C++ 프로그래밍 과정 - 편집/컴파일/링크/실행

- 1단계) 프로그램 작성
- 2단계) 작성한 프로그램을 컴파일 수행
- 3단계) 컴파일된 결과물을 링크
- 4단계) 실행



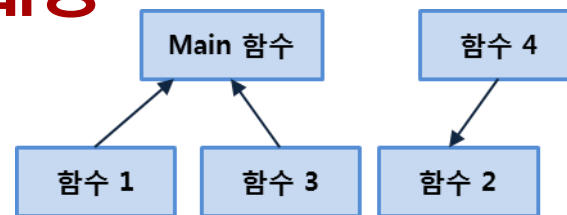
객체지향프로그래밍 개요



절차적 프로그래밍과 객체지향 프로그래밍

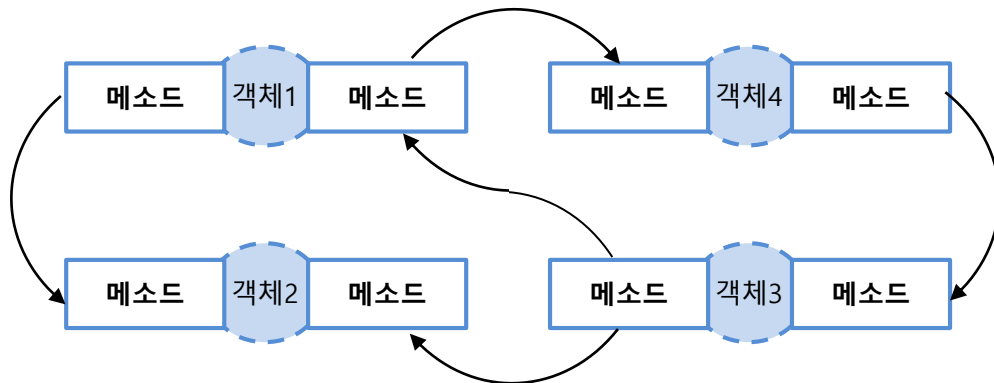
절차적 프로그래밍

- 명령수행의 과정을 중시하는 프로그램 방식
- 프로그램이 함수의 집합으로 구성되므로 함수를 정의하면서 함수에 필요한 데이터를 선언하여 사용
- 대표 언어: C 등



객체지향 프로그래밍

- 객체를 지향하는 프로그램 방식
- 객체를 생산하기 위한 클래스를 설계한 후에 이를 다룰 함수(사용자 인터페이스)를 정의하여 함수로 객체를 다룸
- 컴퓨터 프로그램을 명령어의 목록이 아닌 여러 개의 독립된 단위, 즉 "객체"들의 모임으로 파악하고자 하는 것으로 각각의 객체는 메시지를 주고받고, 데이터를 처리
- 대표 언어: C++, 자바, C# 등



객체지향프로그래밍 개요



객체지향프로그래밍의 특징

• 추상화

- 불필요 정보는 숨기고 중요 정보만을 표현함으로써 프로그램을 간단히 만드는 것
- 데이터를 직접 다루면 데이터가 손상될 수 있으므로 이를 방지하고자 제공되는 것
- 캡슐화(Encapsulation)와 데이터 은닉(Data hiding)

• 다형성(Polymorphism)과 함수의 오버로딩, 연산자 오버로딩

- 다형성이란, 어떤 한 요소에 여러 개념을 넣어 놓는 것
- 즉, 동일한 함수나 연산자를 자료에 따라 다르게 동작하도록 적용하는 것
 - 함수의 오버로딩: 함수명은 동일하지만 매개변수의 자료형이나 개수를 서로 다르게 주어 함수명을 여러 번 정의
 - 연산자 오버로딩: 이미 사용중인 연산자를 다른 용도로 다시 정의해서 사용하는 것

• 상속성(Inheritance)

- 특정 객체의 성격을 다른 객체가 상속받아 사용할 수 있도록 하는 것
- 즉, 새로운 클래스가 기존의 클래스의 자료와 연산을 이용할 수 있게 하는 기능
- 객체지향의 가장 대표적인 특징

II. C++ 시작하기

C++ 프로그램의 구조



기본구조 알아보기

```
/* 주석 내용 */                                ①  
  
//전처리기                                    ②  
#include <iostream>  
  
int main(void)                                ③  
{                                              ④  
    // 일반명령문(식/문)                      ⑤  
    .....  
    return 0;                                ⑥  
}
```

- ① 주석: 사용자의 이해를 돕는 설명
- ② 전처리기: 시스템 함수사용 선언
- ③ Main함수: 프로그램 시작
- ④ 괄호: 함수의 시작과 끝
- ⑤ 문장들: 실행명령문들
 - 식- 하나의 값으로 도출(상수, 변수, 대입식, 반환함수)
 - 문- 컴퓨터에게 동작을 수행케 함(할당문, 호출문, 제어문 등)
- ① 반환: return문으로 반환

C++ 프로그램의 구조



예제 코드로 기본구조 알아보기

```
#include <iostream>
```

```
int main(void)
```

```
{
```

```
    std::cout << "Hello world!" << std::endl;
```

```
    return 0;
```

```
}
```

① 이미 만들어져서 기본으로 제공되는 함수를 표준함수라고 하며, 표준 함수들의 모임을 표준 라이브러리라고 함

전처리기(헤더파일의 선언)

- `iostream` 은 입출력을 위한 헤더파일
- C의 `stdio.h`와 같은 역할
- `iostream` 파일에 정의된 몇 가지 입출력 기능 중 `cout` 사용시 필요
- `#include`지시자 의해 컴파일러에 전달

문자열 출력

- `std::cout` 호출문장
- 인자는 문자열 `"Hello world! "` 임
- `std::endl` 출력하여 개행 수행

반환문

- 함수를 호출한 영역으로 값을 전달(반환)
- 현재 실행중인 함수의 종료를 의미

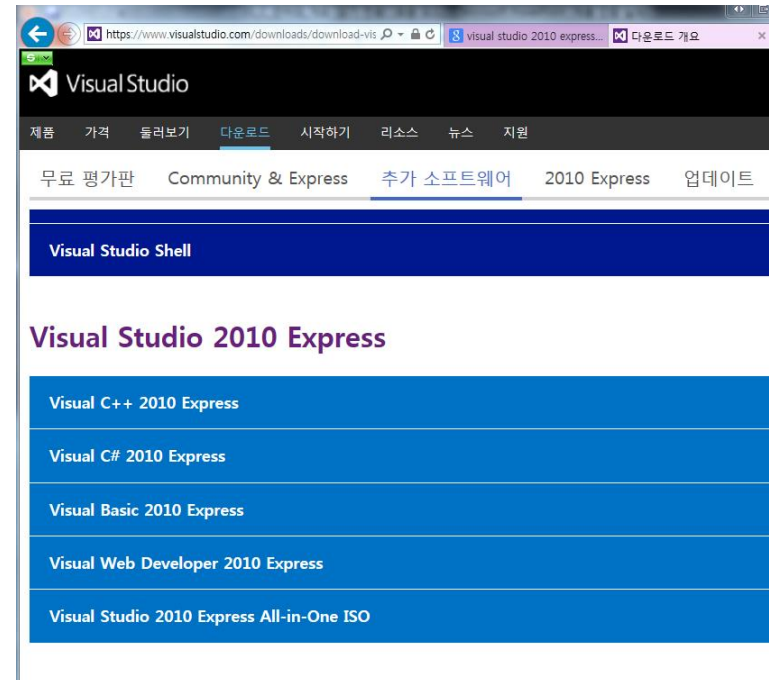
①
②
③
④
⑤
⑥
⑦

C++ 시작하기



(실습) Visual C++ 2010 Express 설치 및 실행

- 1단계) MS 무료다운로드 사이트 접속, 무료배포파일 다운로드 후 설치
(<https://www.visualstudio.com/downloads/download-visual-studio-vs>)
- 2단계) Visual C++ Express 실행
- 3단계) 솔루션, 프로젝트 생성
- 4단계) 소스파일(.cpp)파일 작성
- 5단계) 컴파일 및 실행결과 확인



C++ 시작하기



(실습) Visual Studio Community 2015 설치 및 실행

- Visual Studio Community 2015 무료배포파일 다운로드 후 설치
<https://www.microsoft.com/ko-kr/download/details.aspx?id=48146>
- VS Studio에서 C++ 시작하기
<https://msdn.microsoft.com/ko-kr/library/jj620919.aspx>
- VS Studio 2015에서 디버그 시작하기
<https://msdn.microsoft.com/ko-kr/library/dn986851.aspx>

Microsoft Visual Studio Community 2015

언어 선택:

한국어

다운로드

Microsoft Visual Studio Community 2015는 플랫폼 간 솔루션 개발에 Visual Studio의 강력한 기능을 이용할 수 있도록 하는 모든 기능을 갖춘 새로운 버전입니다.

⊕ 자세한 내용

⊕ 시스템 요구 사항

⊕ 설치 지침

⊕ 관련 리소스

원하는 다운로드 선택

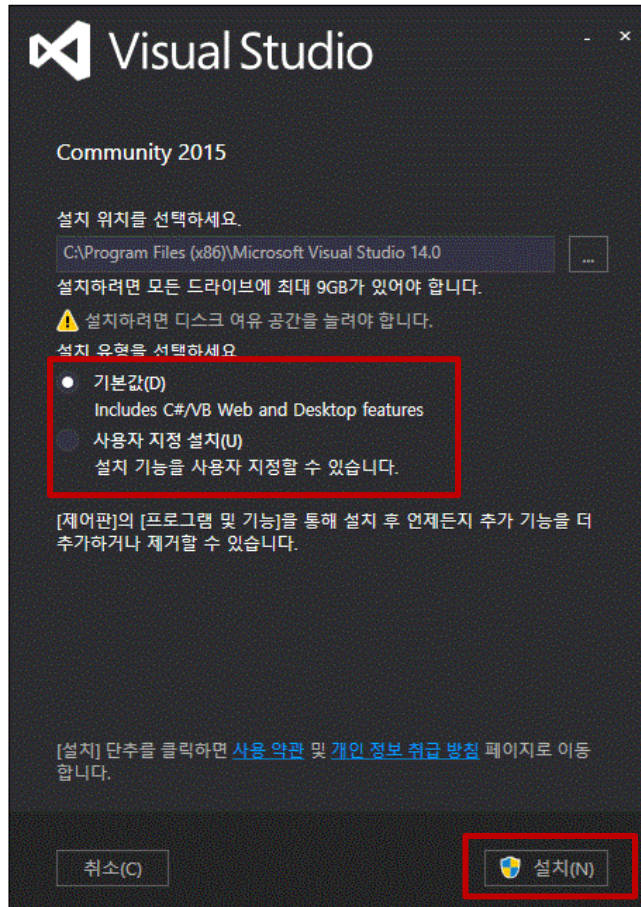
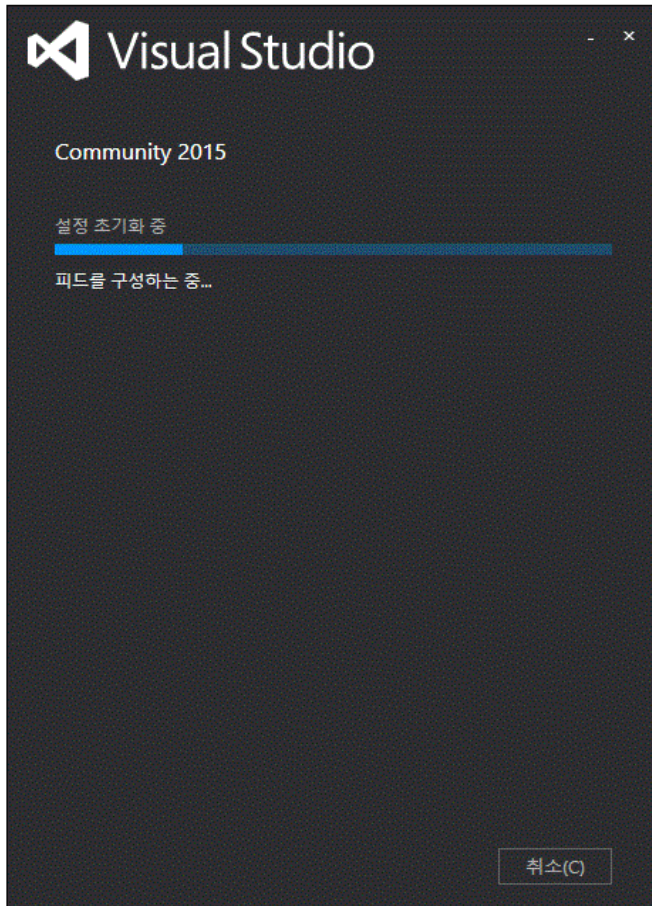
<input type="checkbox"/> 파일 이름	크기
<input checked="" type="checkbox"/> vs_community.exe	3.0 MB
<input type="checkbox"/> vs2015.com_kor.iso	3.7 GB

 vs_community

C++ 시작하기



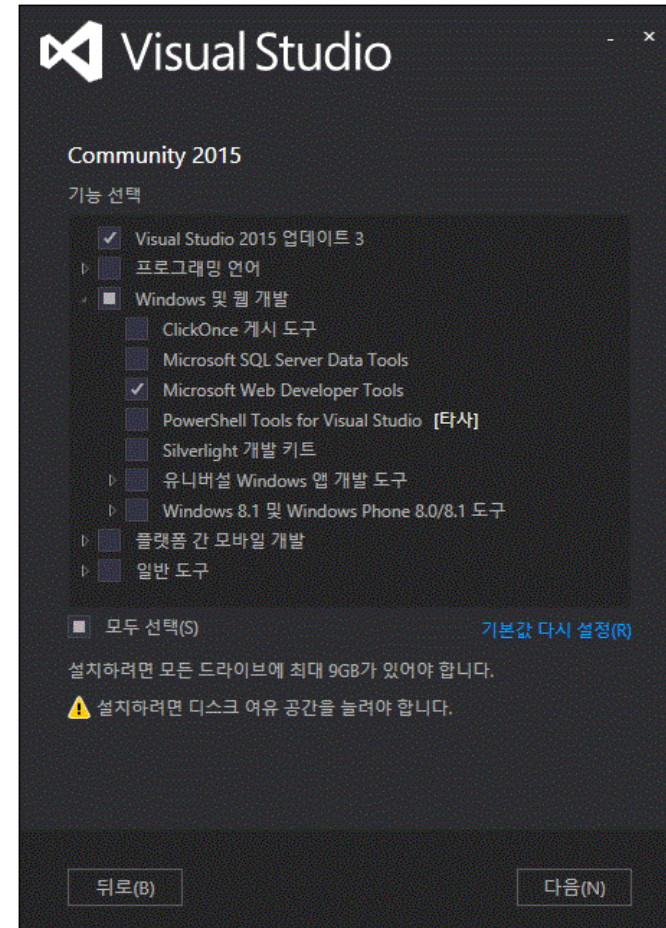
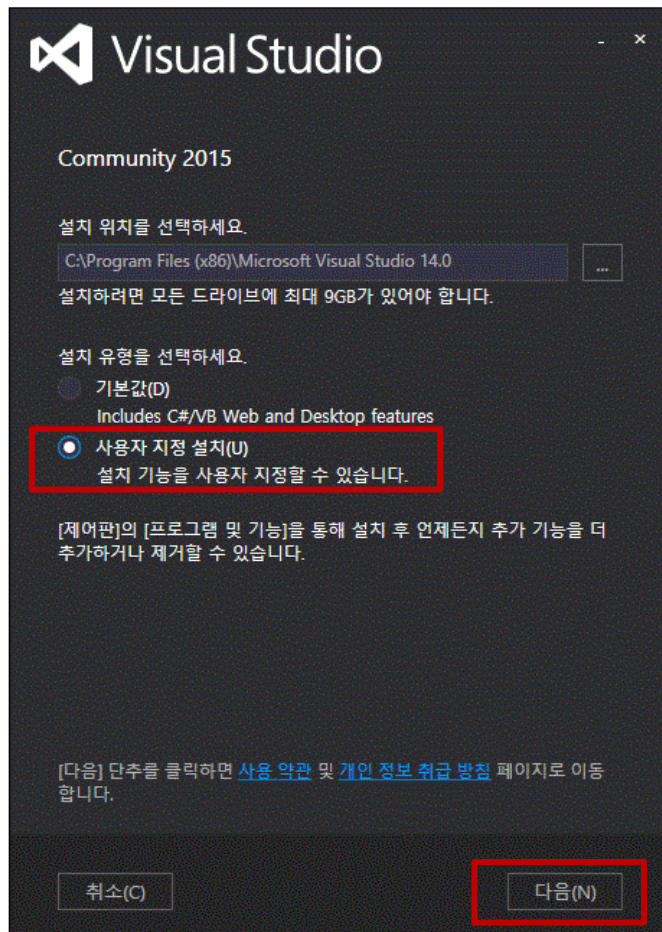
1단계) Visual C++ 설치(2015)



C++ 시작하기



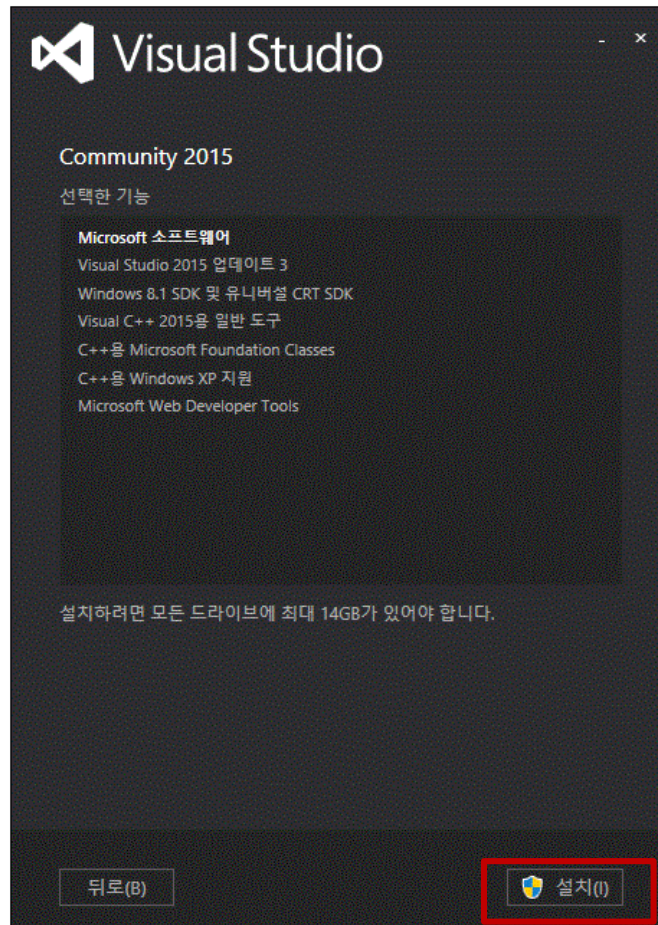
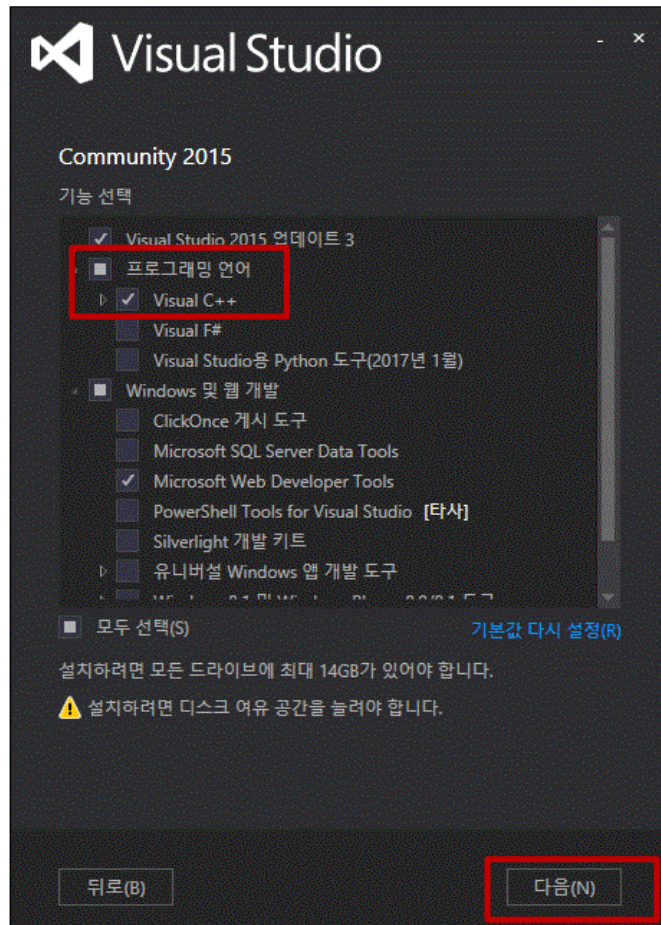
1단계) Visual C++ 설치(2015)



C++ 시작하기



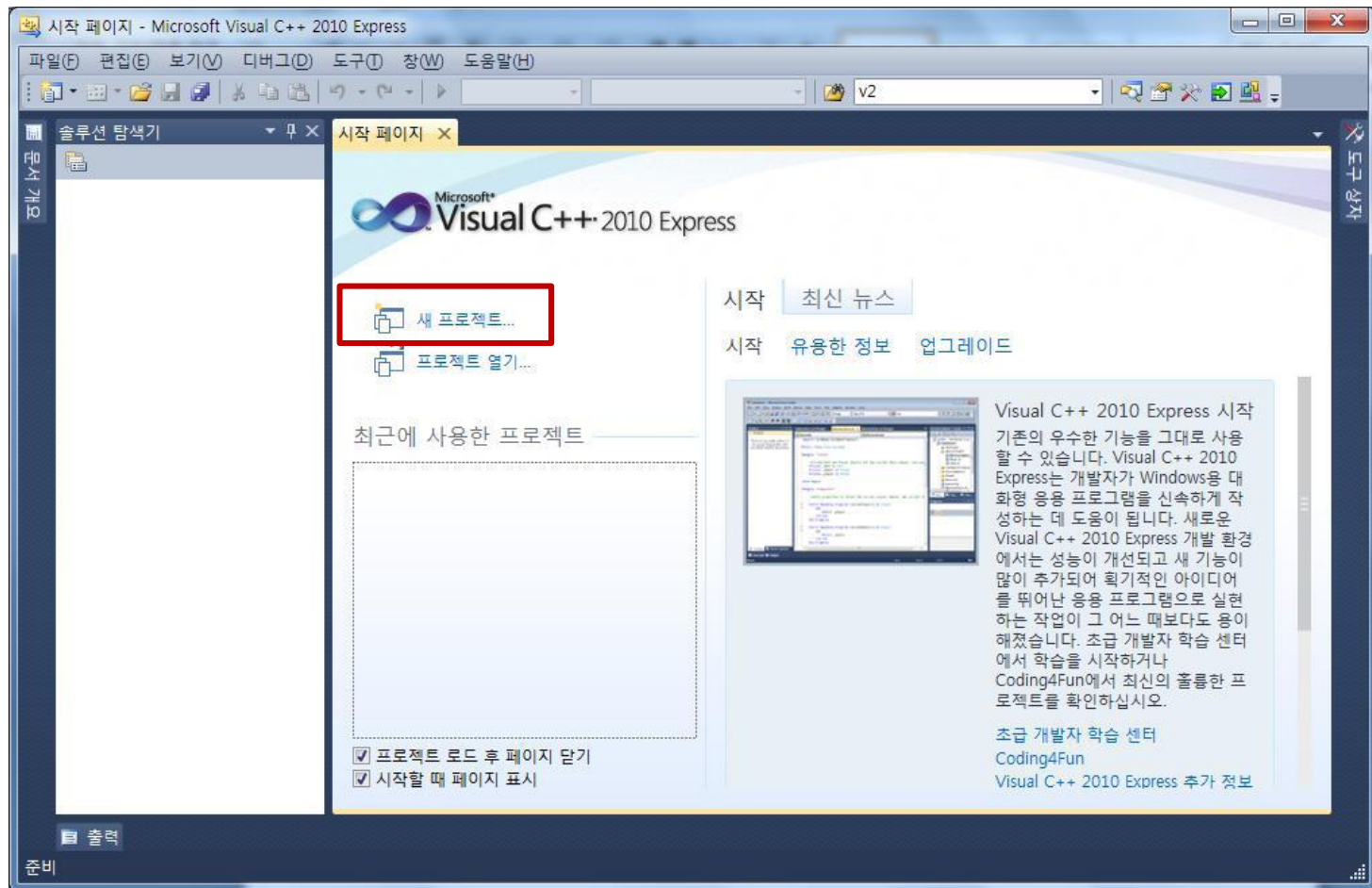
1단계) Visual C++ 설치(2015)



C++ 시작하기



2단계) Visual C++ 실행

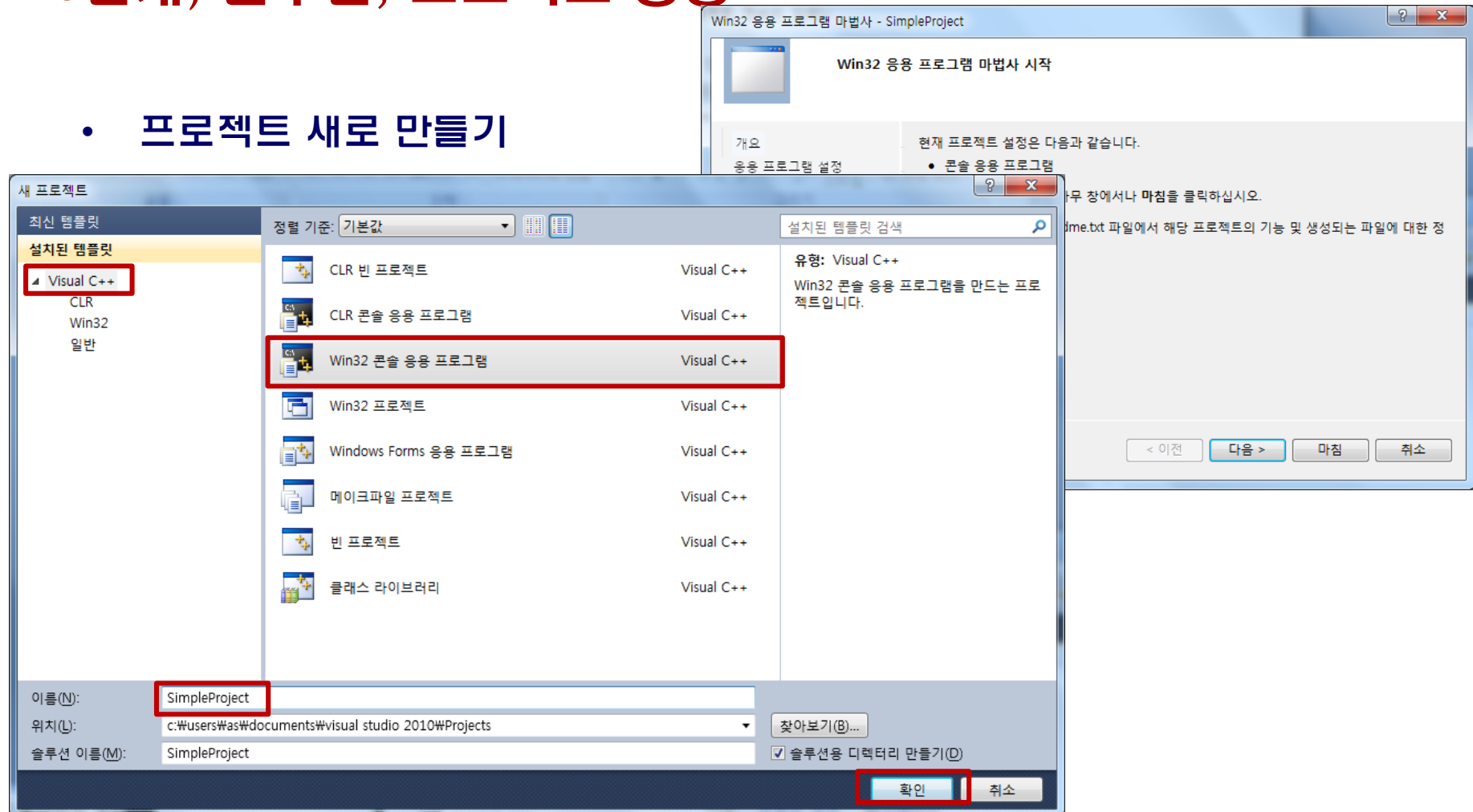


C++ 시작하기



3단계) 솔루션, 프로젝트 생성

• 프로젝트 새로 만들기



C++ 시작하기



3단계) 솔루션, 프로젝트 생성

- 콘솔 응용 프로그램, 빈 프로젝트 선택으로 작업공간 마련을 위한 최종작업

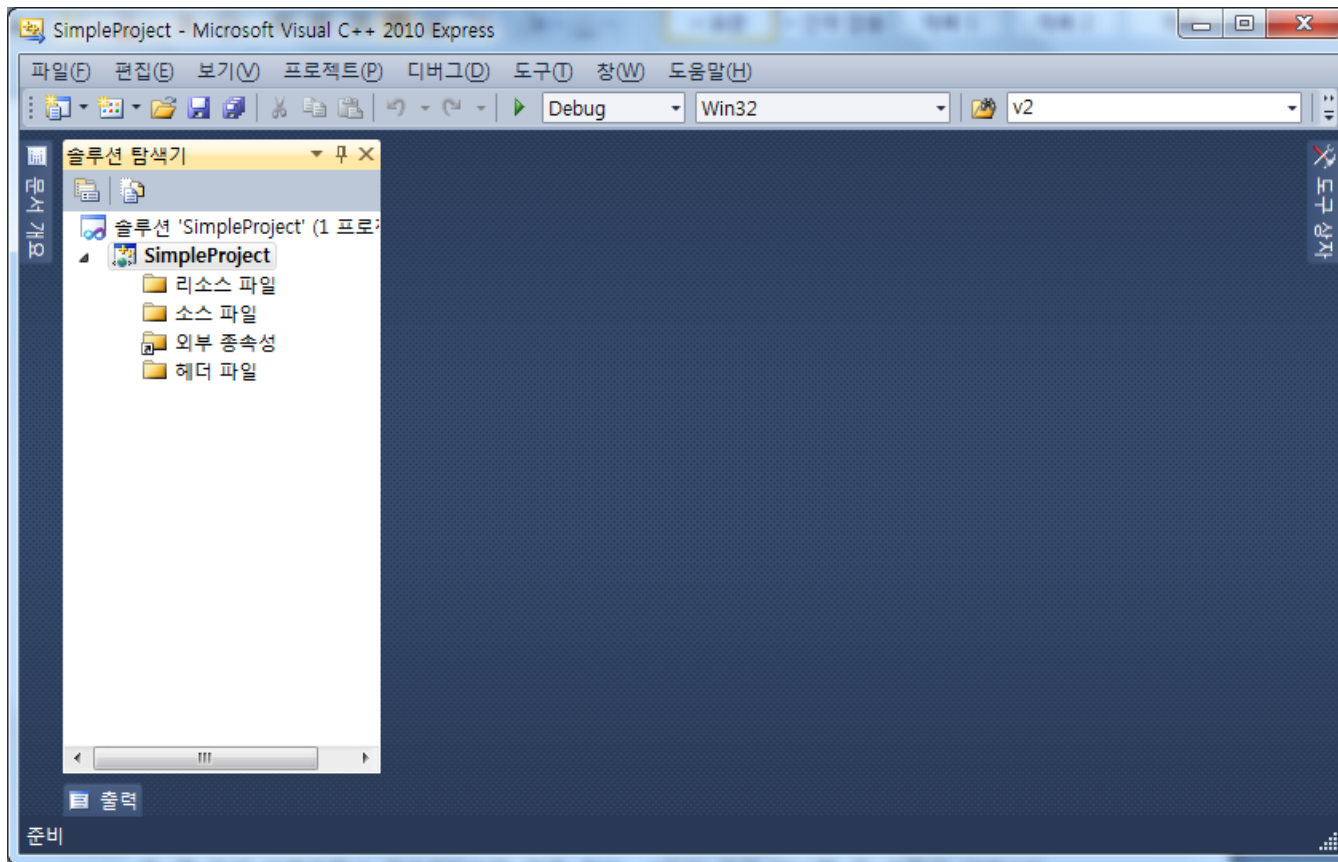


C++ 시작하기



3단계) 솔루션, 프로젝트 생성

- 솔루션 탐색기에서 솔루션, 프로젝트 생성여부 확인

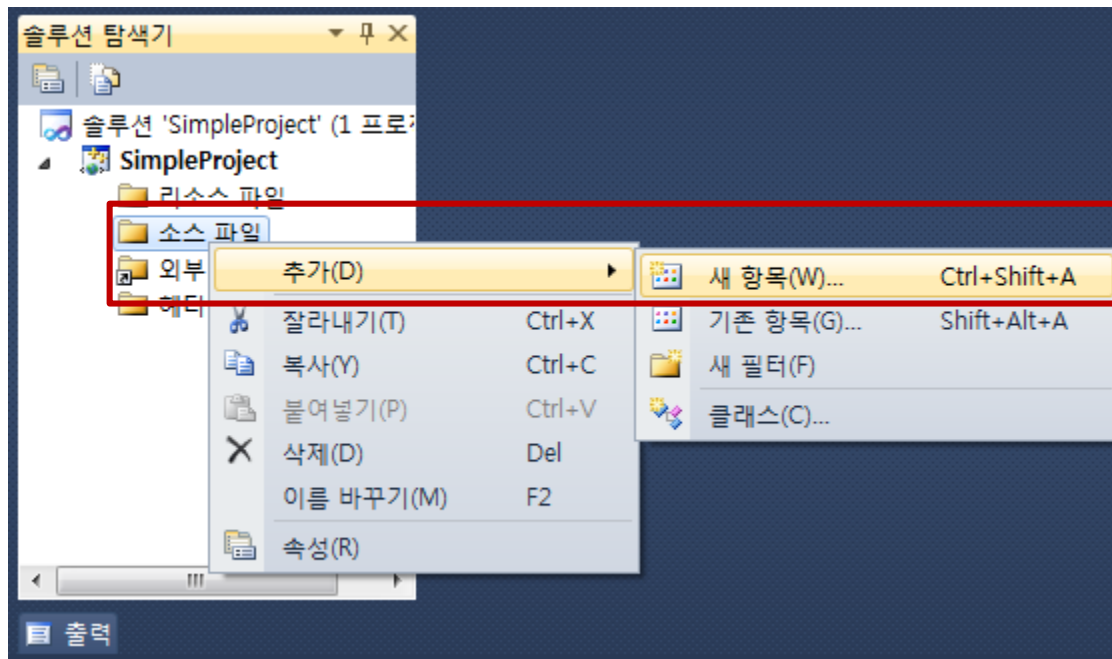


C++ 시작하기



4단계) 소스파일(.cpp) 생성 및 작성

- 솔루션 탐색기 - 소스파일 위치, 오른쪽 마우스 클릭하여 새 소스파일 추가

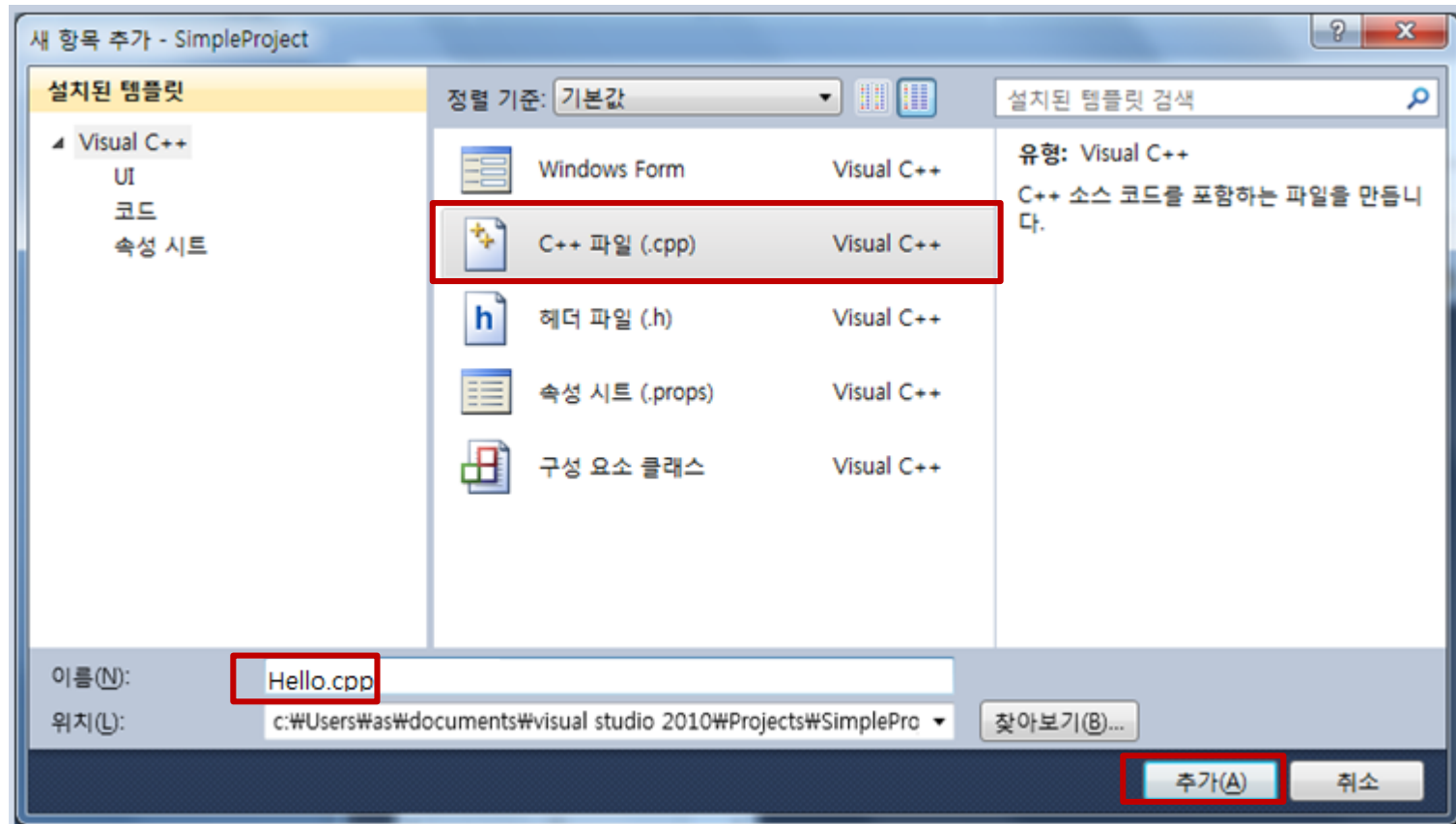


C++ 시작하기



4단계) 소스파일(.cpp) 생성 및 작성

- 확장자를 .cpp로 하는 소스파일 생성

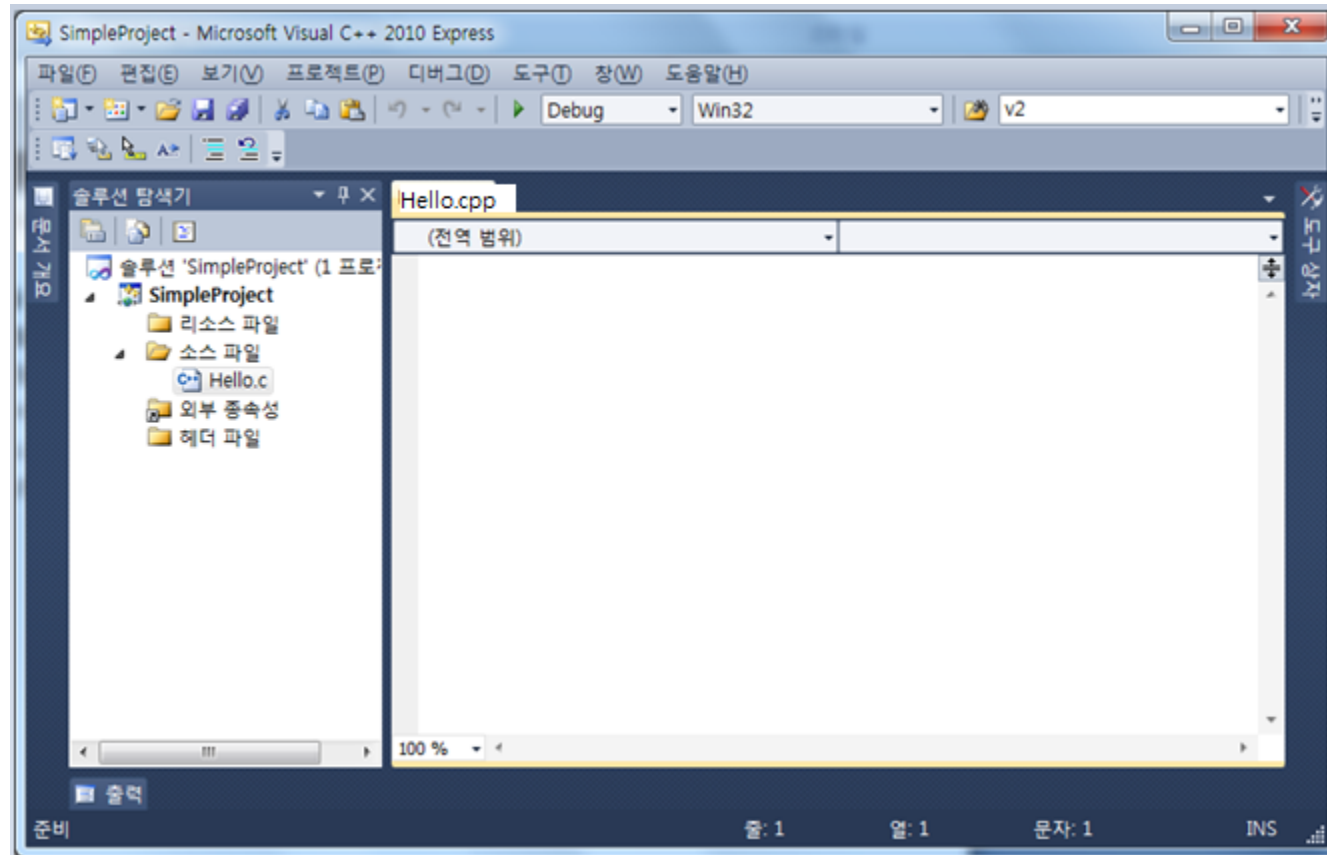


C++ 시작하기



4단계) 소스파일(.cpp) 생성 및 작성

- 빈 소스파일 생성 완료

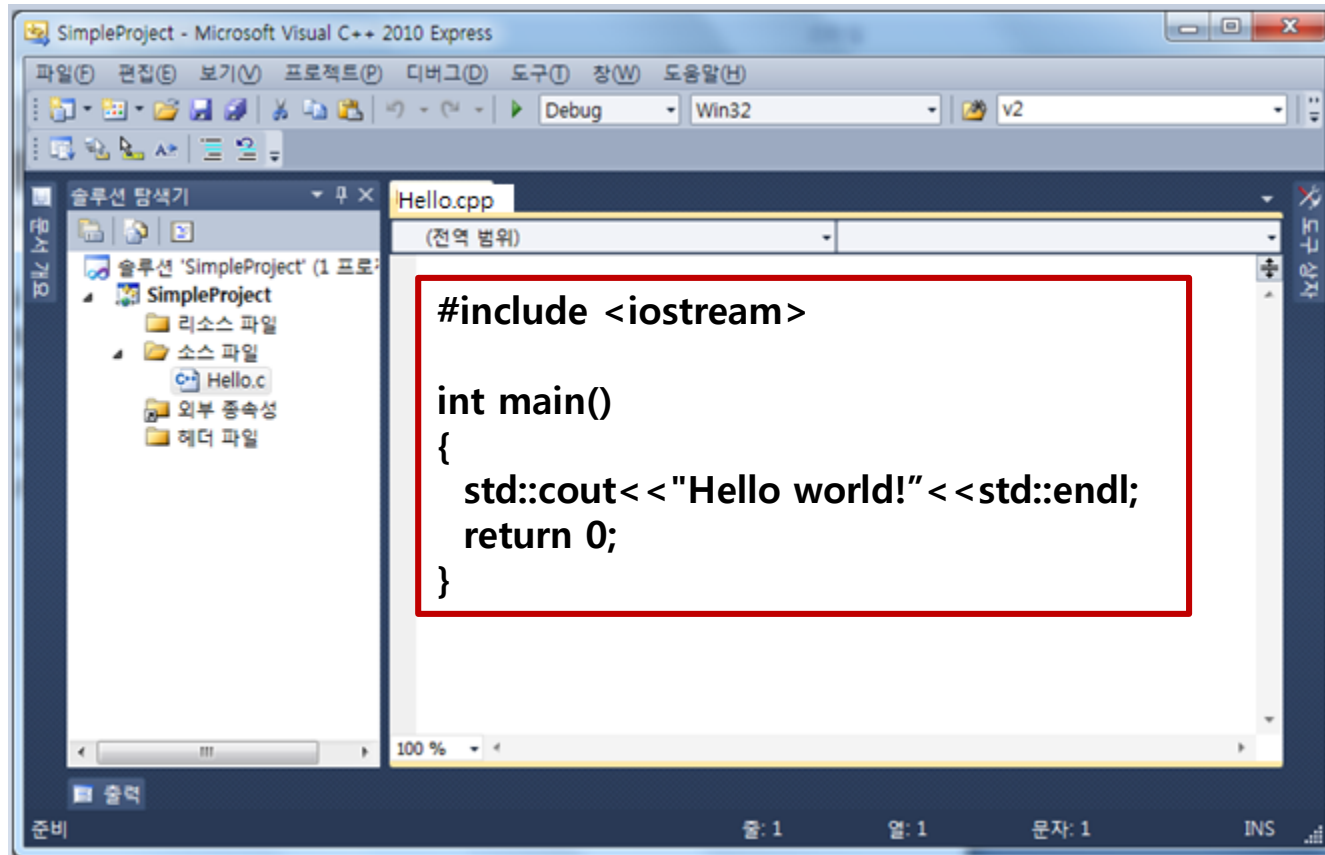


C++ 시작하기



4단계) 소스파일(.cpp) 생성 및 작성

- 빈 소스파일에 소스코드(내용)를 작성

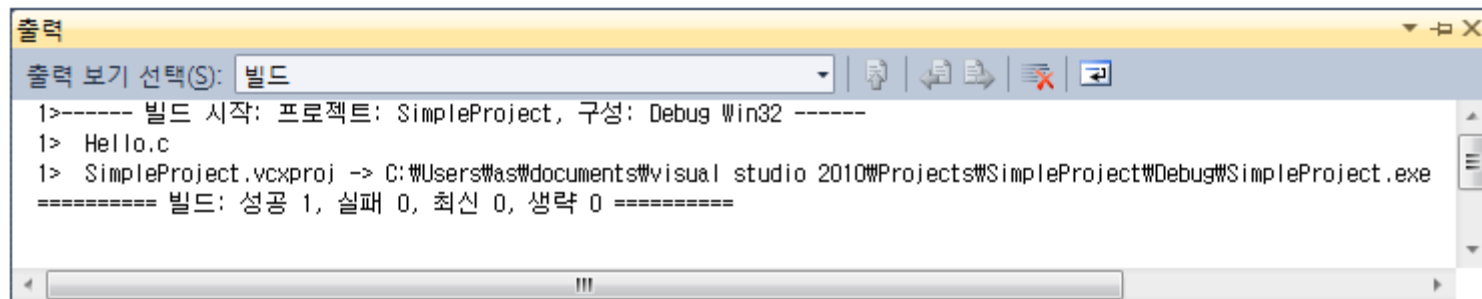
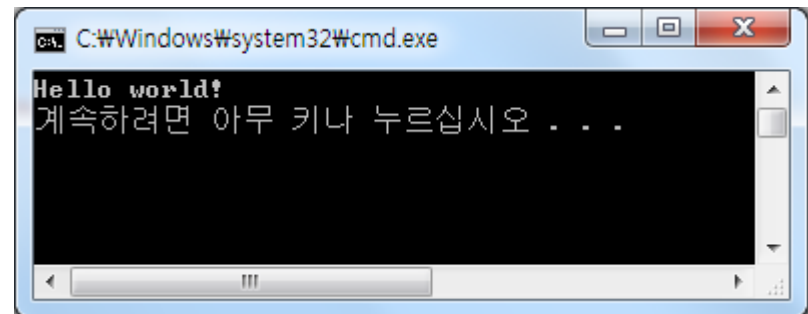
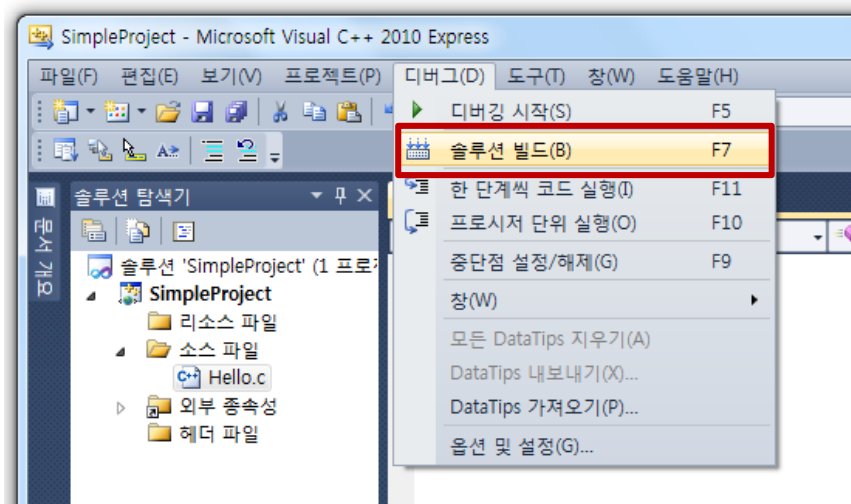


C++ 시작하기



5단계) 컴파일 및 실행결과 확인

- 완성한 소스파일을 컴파일(F7)하고 실행결과를 확인(Ctrl+F5)



C++ 시작하기



(실습) ex1.cpp

- 도스 창에 학과, 성명을 출력하는 소스코드 작성 후 제출
- (예시) “저는 컴퓨터공학과 000입니다.”

※ 반드시 정상실행 되는지 확인 후 제출

III. C와 C++ 비교

C와 C++ 비교



C++의 헤더파일 선언

- 공통적으로 .h를 생략
- 표준입출력을 위한 헤더파일 `iostream` 지정
 - `iostream` 파일에 몇 가지 입출력 기능 정의
 - `#include`지시자 의해 필요한 입출력 정의 내용을 컴파일러에 전달하여 대체

```
#include <iostream>
```

① 헤더파일 선언

EX1-01

```
int main(void)
{
    int num=20;
    std::cout<<"Hello World!"<<std::endl;
    std::cout<<"Hello " <<"World!"<<std::endl;
    std::cout<<num<<' ' <<'A';
    std::cout<<' ' <<3.14<<std::endl;
    return 0;
}
```

실행 결과

```
Hello world!
Hello world!
20 A 3.14
```

C와 C++ 비교



C++의 입출력 방식

- printf/scanf 함수대신 cout/cin 사용하여 입출력 수행

(출력형식) std::cout<<'출력대상1'<<'출력대상2'<<'출력대상3';

- 범위지정 연산자(::), 출력연산자(cout), 스트림 삽입연산자(<<) 사용
- 출력대상에 따른 서식지정 불필요

```
#include <iostream>
```

EX1-01

```
int main(void)
```

```
{
```

```
    int num=20; ① 문자열 출력
```

```
    std::cout<<"Hello World!"<<std::endl; ① 개행(\n) 수행
```

```
    std::cout<<"Hello "<<"World!"<<std::endl; ① 복수 문자열
```

```
    std::cout<<num<<' '<<'A';
```

```
    std::cout<<' '<<3.14<<std::endl; ① 데이터 값, 숫자, 문자 등
```

```
    return 0;
```

```
}
```

실행 결과

```
Hello world!
Hello world!
20 A 3.14
```

C와 C++ 비교



C++의 입출력 방식

- printf/scanf 함수대신 cout/cin 사용하여 입출력 수행

(입력형식) std::cin >> '변수';

- 콘솔로 입력받은 값을 변수에 입력, 변수선언은 함수의 중간 부분에서도 가능

```
#include <iostream>
```

EX1-02

```
int main(void)
```

```
{
```

```
    int val1;
```

```
    std::cout<<"첫 번째 숫자입력: ";
```

```
    std::cin>>val1;
```

① 콘솔로 입력 받은 값을 >>방향으로 입력

```
    int val2;
```

```
    std::cout<<"두 번째 숫자입력: ";
```

```
    std::cin>>val2;
```

① 선언위치 제한 없음

```
    int result=val1+val2;
```

```
    std::cout<<"덧셈결과: "<<result<<std::endl;
```

```
    return 0;
```

```
}
```

실행 결과

첫 번째 숫자입력: 3
두 번째 숫자입력: 5
덧셈결과: 8

C와 C++ 비교



C++의 입출력 방식

- (EX1-03) 데이터입력하기
 - std::in로 복수데이터 입력
 - 입력데이터는 화이트스페이스 문자(스페이스 바, 엔터, 탭 등 공백 문자)로 구분

실행 결과

두 개의 숫자입력: 3 7
두 수 사이의 정수 합: 15

EX1-03

```
#include <iostream>

int main(void)
{
    int val1, val2;
    int result=0;
    std::cout<<"두 개의 숫자입력: ";
    std::cin>>val1>>val2; ① 연이어 데이터 입력
    if(val1<val2)
    {
        for(int i=val1+1; i<val2; i++)
            result+=i;
    }
    else ① for문 내의 지역변수 선언
    {
        for(int i=val2+1; i<val1; i++)
            result+=i;
    }
    std::cout<<"두 수 사이의 정수 합: "<<result<<std::endl;
    return 0;
}
```

C와 C++ 비교



C++의 입출력 방식

- (EX1-04) 배열기반 문자열 입출력하기

실행 결과

이름은 무엇입니까? Yoon
좋아하는 프로그래밍 언어는 무엇인가요? C++
내 이름은 Yoon입니다.
제일 좋아하는 언어는 C++입니다.

```
#include <iostream>

int main(void)
{
    char name[100];
    char lang[200];
    std::cout<<"이름은 무엇입니까? ";
    std::cin>>name;
    std::cout<<"좋아하는 프로그래밍 언어는 무엇인가요? ";
    std::cin>>lang;
    std::cout<<"내 이름은 "<<name<<"입니다.\n";
    std::cout<<"제일 좋아하는 언어는 "<<lang<<"입니다."<<std::endl;
    return 0;
}
```

C와 C++ 비교



C++의 함수 오버로딩(Function Overloading)

- 매개변수의 선언이 다른 경우, 동일 명칭의 함수 정의가 가능
- 즉, 함수 오버로딩이란 동일한 이름의 함수를 중복 정의하는 것
 - C++은 함수명과 전달인자의 정보(매개변수형태/개수) 모두 참조해 호출함수 결정
 - C는 매개변수와 관계 없이 함수 이름에만 의존하여 함수 호출

```
int MyFunc(int num)
{
    num++;
    return num;
}
```

```
int MyFunc(int a, int b)
{
    return a+b;
}
```

```
int main(void)
{
    MyFunc(20); // MyFunc(int num) 함수의 호출
    MyFunc(30, 40); // MyFunc(int a, int b) 함수의 호출
    return 0;
}
```


C와 C++ 비교



C++의 함수 오버로딩(Function Overloading)

- (예시) 함수 오버로딩

- 매개변수 자료형이 상이한 경우 → 함수 오버로딩 성립(O)

```
int MyFunc(char c) { . . . }  
int MyFunc(int n) { . . . }
```

- 매개변수 개수가 상이한 경우 → 함수 오버로딩 성립(O)

```
int MyFunc(int n) { . . . }  
int MyFunc(int n1, int n2) { . . . }
```

- 반환형이 상이한 경우 → 함수 오버로딩 미성립(X)

```
void MyFunc(int n) { . . . }  
int MyFunc(int n) { . . . }
```



C++의 디폴트 매개변수(Default Parameter)

- 전달되지 않는 인자를 대신하기 위한 기본값이 설정되어 있는 변수
 - 디폴트값은 함수 선언에만 위치하며, 부분적으로 전달 가능

- (예시) 인자가 1개인 경우

- MyFuncOne(); 인자 미전달 시, 7(디폴트값) 전달
- 즉, MyFuncOne(7); 실행된 것과 동일한 효과

```
int MyFuncOne(int num=7)
{
    return num+1;
}
```

- (예시) 인자가 2개인 경우

- MyFuncTwo(); 인자 미전달 시, 각각 5와 7(디폴트값) 전달
- 즉, MyFuncTwo(5,7); 실행된 것과 동일한 효과

```
int MyFuncTwo(int num1=5, int num2=7)
{
    return num1+num2;
}
```

C와 C++ 비교



C++의 디폴트 매개변수(Default Parameter)

- 디폴트 값의 설정 - 함수 선언에만 설정 가능 (함수 선언문 존재 시)
 - 컴파일러는 함수의 디폴트 값 지정여부를 알아야 함수 호출문 컴파일 가능

EX1-05

```
#include <iostream>
```

```
int Adder(int num1=1, int num2=2);
```

① 함수 선언부에 설정

```
int main(void)
```

```
{
```

```
std::cout<<Adder()<<std::endl;
```

① 각각 1과 2가 전달

```
std::cout<<Adder(5)<<std::endl;
```

① 각각 5와 2가 전달

```
std::cout<<Adder(3, 5)<<std::endl;
```

① 각각 3과 5가 전달

```
return 0;
```

```
}
```

```
int Adder(int num1, int num2)
```

① 함수 정의부에 설정 불가

```
{
```

```
return num1+num2;
```

```
}
```

C와 C++ 비교



C++의 디폴트 매개변수(Default Parameter)

• 디폴트 값의 부분 설정

- 매개변수 일부에만 디폴트 값 설정하고, 미지정 매개변수에만 인자 전달하기 가능

```
int YourFunc(int num1, int num2=5, int num3=7) { ... }
```

```
YourFunc(10);
```

① YourFunc(10, 5, 7) 수행

```
YourFunc(10, 20);
```

① YourFunc(10, 20, 7) 수행

- 전달 인자는 왼쪽부터 채워짐. 따라서, 디폴트 값은 오른쪽부터 채워야 함

```
int YourFunc(int num1, int num2, int num3=30) { . . . } (○)
```

```
int YourFunc(int num1, int num2=20, int num3=30) { . . . } (○)
```

```
int YourFunc(int num1=10, int num2=20, int num3=30) { . . . } (○)
```

- 오른쪽이 빈 상태로 왼쪽 매개변수만 일부 채워진 디폴트 값은 무의미(컴파일에러)

```
int WrongFunc(int num1=10, int num2, int num3) { . . . } (×)
```

```
int WrongFunc(int num1=10, int num2=20, int num3) { . . . } (×)
```

IV. 실습

간단한 문자열을 출력하는 프로그램

```
01 #include <iostream> // 헤더파일을 포함시키는 문장
02 void main()
03 {
04 /* cout은 출력을 담당하는 객체로써,
05 스트림 삽입 연산자(stream insertion operator)인
06 <<를 이용해서 ""안에 있는 문자열을 출력한다. */
07
08 std::cout<<"C++ 세계에 오신 것을 환영합니다. \n";
09 }
```

여러 줄에 걸쳐서 여러 문장 출력하기

```
01 #include <iostream> // 헤더파일을 포함시키는 문장
02 void main()
03 {
04     std::cout<<" 이 름 : O O O "<<std::endl;
05     std::cout<<" 소 속 : 컴퓨터공학부 "<<std::endl;
06     std::cout<<" 이메일 : OOO@sungkyul.ac.kr "<<std::endl;
07 }
```

정수형 상수 출력하기

```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     cout<<"사과가 "<< 10 <<"개 있다.\n";
06     cout<<"사과가 한 개에 "<< 500 <<"원이다.\n";
07 }
```


정수형 변수 사용하기

```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     int unit; // 변수 선언
06     int count;
07     int total;
08     unit=500; // 한 개에 500원짜리
09     count=5; // 5개를 구입한
10     total=unit*count; // 총 금액 구하기
11     cout<<" 총 금액 : " << total <<"\n";
12 }
```

cin을 이용해서 변수 입력받기

```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     int unit, count, total; // 변수 선언
06     cout<<"상품의 단가를 입력하시오-> ";
07     cin>>unit;
08     cout<<"구입할 개수를 입력하시오-> ";
09     cin>>count;
10     total=unit*count; // 키보드에서 입력받은 데이터로 총금액 구하기
11     cout<<" 총 금액 : " << total <<"\n";
12 }
```

함수의 오버로딩 없이 절대값 구하기

→ 매개변수의 자료형이 다른 함수의 오버로딩 이용하여 절대값 구하기

```
#include <iostream>
using namespace std;
int myabs(int num)
{ if(num<0)
    num=-num;
  return num; }

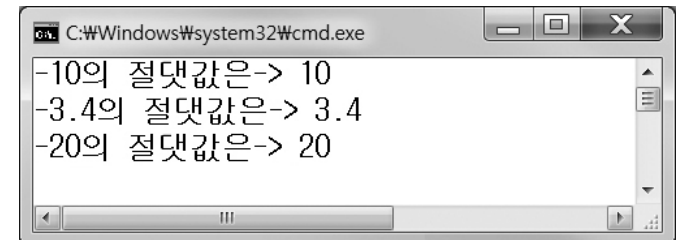
double fmyabs(double num)
{ if(num<0)
    num=-num;
  return num; }

long int lmyabs(long int num)
{ if(num<0)
    num=-num;
  return num; }
```

```
Void main( )
{ int a=-10;
  cout<<a<<"의 절대값은 "<<myabs(a)<<endl;

  double b=-3.4;
  cout<<b<<"의 절대값은 "<<fmyabs(b)<<endl;

  long int c=-20L;
  cout<<c<<"의 절대값은 "<<lmyabs(b)<<endl;
}
```



```
C:\Windows\system32\cmd.exe
-10의 절대값은-> 10
-3.4의 절대값은-> 3.4
-20의 절대값은-> 20
```

매개변수의 개수가 다른 함수의 오버로딩 살펴보기

```
01 #include <iostream>
02 using namespace std;
03 void print(int x, int y, int z)
04 {
05     cout<<x<<" "<<y<<" "<<z<<endl;
06 }
07 void print(int x, int y)
08 {
09     cout<<x<<" "<<y<<endl;
10 }
11 void print(int x)
12 {
13     cout<<x<<endl;
14 }
15 void main()
16 {
17     print(10, 20, 30);
18     print(10, 20);
19     print(10);
20 }
```

함수의 매개변수에 기본값 지정하기

```
01 #include <iostream>
02 using namespace std;
03 void print(int x=0, int y=0, int z=0);
04 void main()
05 {
06     print(10, 20, 30);
07     print(10, 20);
08     print(10);
09     print();
10 }
11 void print(int x, int y, int z)
12 {
13     cout<<x<<" "<<y<<" "<<z<<endl;
14 }
```

과제1



파일명 “과제1_분반_학번_이름”으로 제출

- 과제1

- 1-1. 절차적 프로그래밍과 객체지향 프로그래밍의 비교 정리하기
- 1-2. 객체지향언어의 특징 정리하기

- 제출 시 주의사항

- 보고서(HWP, MS Word) 파일 작성
- 겉표지에 “학과, 학년, 분반, 학번, 이름” 반드시 기재할 것
- 분량: 겉표지 포함 4~5페이지 (A4 11포인트)



감사합니다

mnshim@sungkyul.ac.kr

