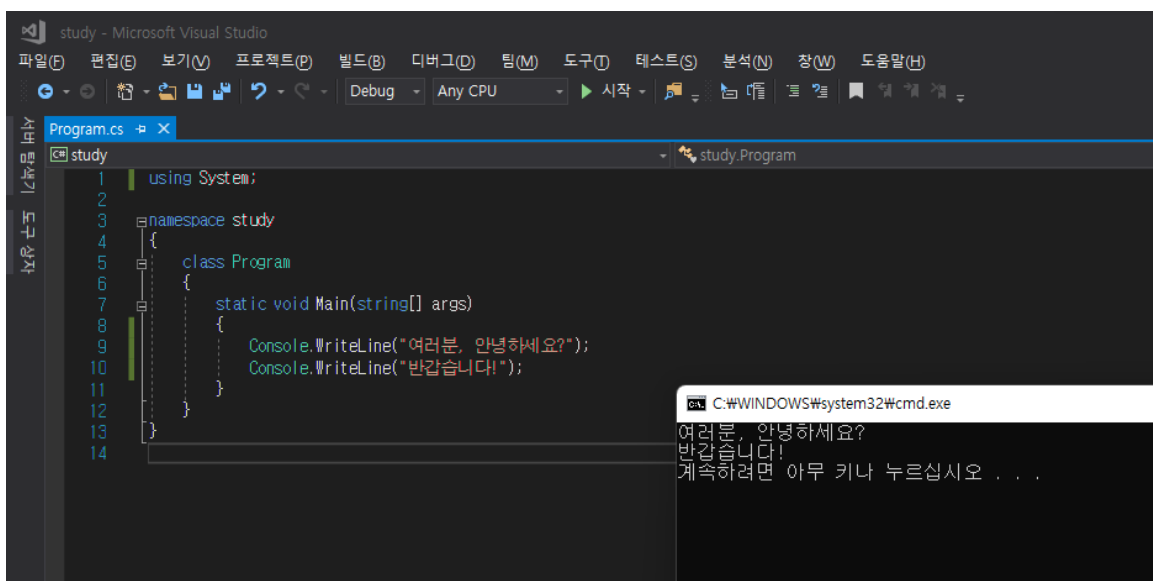
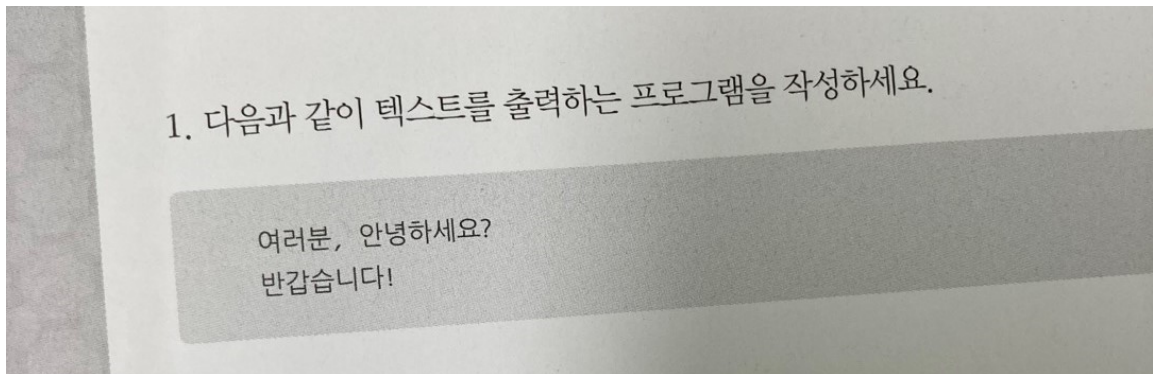




1주차 (2장~3장)

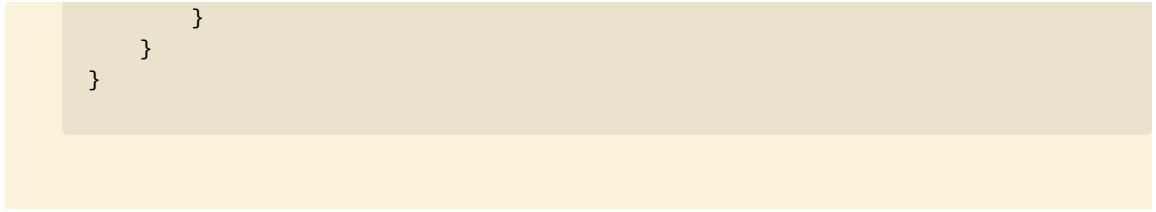
- 2장



▼ 코드

```
using System;

namespace study
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("여러분, 안녕하세요?");
            Console.WriteLine("반갑습니다!");
        }
    }
}
```



- 3장

1. 다음 코드에서 잘못된 부분을 찾고, 그 이유를 설명하세요.

```
int a = 7.3;
float b = 3.14;
double c = a * b;
char d = "abc";
string e = '한';
```

2. 값 형식과 참조 형식의 차이는 무엇인가요?

3. 박싱과 언박싱을 설명하세요.

4. 다음과 같이 사용자로부터 사각형의 너비와 높이를 입력받아 넓이를 계산하는 프로그램을 완성하세요. 다음 코드 중 주석 부분을 바꾸면 됩니다.

실행 결과

```
사각형의 너비를 입력하세요.
30
사각형의 높이를 입력하세요.
40
사각형의 넓이는 : 1200
```

```
using System;

namespace RectArea
{
    class MainApp
    {
        public static void Main()
        {
            Console.WriteLine("사각형의 너비를 입력하세요.");
            string width = Console.ReadLine();

            Console.WriteLine("사각형의 높이를 입력하세요.");
            string height = Console.ReadLine();
```

1. int는 정수형인데, double을 받을 수 없음

float는 F를 입력해야 올바른 값이 나옴

char는 str.ToCharArray() 함수를 사용하여 문자열 변수 str내의 모든 문자를 charArray 문자 배열로 변환(참고 :

<https://www.delftstack.com/ko/howto/csharp/csharp-convert-string-to-char/>)

▼ 코드

```
using System;

namespace study
{
    class Program
    {
        static void Main(string[] args)
        {
            String str = "abc";
            char[] d = str.ToCharArray();

            Console.WriteLine(d);
        }
    }
}
```

string 사용 시 “ ” 로 나타내어야함

2. 값형식의 데이터는 항상 값으로 복사되지만, 참조형식의 데이터는 항상 참조로 복사된다.

- 값형식 : 해당 메소드 실행이 종료되면 사라짐, 어느 하나의 값을 수정해도 다른쪽에는 영향을 끼치지 않음

▼ 코드

```
using System;

namespace study
{
    class Program
    {
        static void Main(string[] args)
        {
            int a = 2;
            int b = a;

            Console.WriteLine("a : " + a);
            Console.WriteLine("b : " + b);

            a++;
            Console.WriteLine("a++ : " + a);
        }
    }
}
```

- 참조형식 : 서로 다른 두 변수가 같은 데이터를 참조한다면 한개의 변수값을 변경하면 다른 변수도 영향을 받음

▼ 코드

```
using System;

namespace ValueVSReference
{
    class MyInt // 사용자 정의 클래스 - 참조 형식
    {
        public int Value { get; set; } //속성
        public MyInt(int value) //생성자
        {
            Value = value;
        }
        public override string ToString() //재정의
        {
            return Value.ToString();
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            //참조 형식에 대한 테스트 코드
            MyInt a = new MyInt(1); // MyInt 개체를 생성하여 변수 a에 대입
            MyInt b = a; //MyInt 형식 변수 b 선언 및 초기화(a를 초기화에 사용)

            a.Value++; //a의 속성 Value를 1 증가
            Console.WriteLine("a:{0} b:{1}", a, b); //참조 형식인 a와 b를 출력
        }
    }
}
```

3. 박싱 : 값형식 → 참조형식

▼ 코드

```
using System;

namespace Study
{
    class Program
    {
        static void Main(string[] args)
        {
            int i = 123;
```

```

        object o = i; // 박싱

        Console.WriteLine(o.ToString());
    }
}

```

언박싱 : 참조형식 → 값형식

▼ 코드

```

using System;

namespace Study
{
    class Program
    {
        static void Main(string[] args)
        {
            int i = 123; // a value type*

            object o = i; // boxing

            int j = (int)o; // unboxing
        }
    }
}

```

연습 문제 ☆

```
// 이 곳에 사각형의 넓이를 계산하고  
// 출력하는 루틴을 추가하세요.
```

```
    }  
  }  
}
```

5. 다음 코드를 컴파일한 후의 a와 b는 각각 어떤 데이터 형식이겠습니까?

```
var a = 2020;  
var b = "double";
```

The screenshot shows a C# program in a file named Program.cs. The code defines a namespace RectArea containing a class MainApp. The MainApp class has a static Main method that prompts the user for the width and height of a rectangle, calculates the area, and displays the result. The console output shows the program running, with the user entering 30 for width and 40 for height, resulting in an area of 1200.

```
1 using System;
2
3 namespace RectArea
4 {
5     class MainApp
6     {
7         public static void Main()
8         {
9             Console.WriteLine("사각형의 너비를 입력하세요.");
10            string width = Console.ReadLine();
11
12            Console.WriteLine("사각형의 높이를 입력하세요.");
13            String height = Console.ReadLine();
14
15
16            double area = double.Parse(width) * double.Parse(height);
17            Console.WriteLine("사각형의 넓이는 : " + area);
18        }
19    }
20 }
21
22
23
```

C:\WINDOWS\system32\cmd.exe

사각형의 너비를 입력하세요.
30
사각형의 높이를 입력하세요.
40
사각형의 넓이는 : 1200
계속하려면 아무 키나 누르십시오 . . .

4.

▼ 코드

```
using System;

namespace RectArea
{
    class MainApp
    {
        public static void Main()
        {
            Console.WriteLine("사각형의 너비를 입력하세요.");
            string width = Console.ReadLine();

            Console.WriteLine("사각형의 높이를 입력하세요.");
            String height = Console.ReadLine();


            double area = double.Parse(width) * double.Parse(height);
            Console.WriteLine("사각형의 넓이는 : " + area);
        }
    }
}
```



```

    }
}
}

```

5. a : Int32 , b : String

The screenshot shows a Visual Studio IDE with a C# file named 'Main0' in the 'RectArea.MainApp' namespace. The code defines a 'MainApp' class with a 'Main' method that declares two variables, 'a' and 'b', and prints their types using 'GetType()'. A console window titled 'C:\WINDOWS\system32\cmd.exe' displays the output: 'System.Int32', 'System.String', and a Korean message.

```

1  using System;
2
3  namespace RectArea
4  {
5      class MainApp
6      {
7          public static void Main()
8          {
9              var a = 2020;
10             var b = "double";
11
12             Console.WriteLine(a.GetType());
13             Console.WriteLine(b.GetType());
14         }
15     }
16 }
17
18
19

```

Output in console window:

```

System.Int32
System.String
계속하려면 아무 키나 누르십시오 . . .

```

▼ 코드

```

using System;

namespace Study
{
    class MainApp
    {
        public static void Main()
        {
            var a = 2020;
            var b = "double";

```

```
        Console.WriteLine(a.GetType());  
        Console.WriteLine(b.GetType());  
    }  
}  
}
```