

닷넷프로그래밍

과제1

2019305059

이현수

4-12 복소수클래스

(소스코드 캡처)


```
1    using System;
    참조 29개
2    class Complex
3    {
4        public double real;
5        public double image;
        참조 0개
6        public Complex(double num)
7        {
8            real = num;
9            image = num;
10       }
        참조 6개
11       public Complex(double real, double image)
12       {
13           this.real = real;
14           this.image = image;
15       }
        참조 14개
16       override public string ToString()
17       {
18           return "(" + real + ", " + image + "i)";
19       }
        참조 1개
20       public static Complex AddComplex(Complex x1, Complex x2)
21       {
22           Complex x = new Complex(0.0, 0.0);
23           x.real = x1.real + x2.real;
24           x.image = x1.image + x2.image;
25           return x;
26       }
        참조 1개
27       public static Complex SubComplex(Complex x1, Complex x2)
28       {
29           Complex x = new Complex(0.0, 0.0);
30           x.real = x1.real - x2.real;
31           x.image = x1.image - x2.image;
32           return x;
33       }
        참조 1개
34       public static Complex MulComplex(Complex x1, Complex x2)
35       {
36           Complex x = new Complex(0.0, 0.0);
37           x.real = x1.real * x2.real - x1.image * x2.image;
38           x.image = x1.real * x2.image + x1.image * x2.real;
39           return x;
40       }
        참조 1개
41       public static Complex DivComplex(Complex x1, Complex x2)
42       {
43           Complex x = new Complex(0.0, 0.0);
44           x.real = (x1.real * x2.real + x1.image * x2.image) / (Math.Pow(x2.real, 2) + Math.Pow(x2.image, 2));
45           x.image = (x1.image * x2.real - x1.real * x2.image) / (Math.Pow(x2.real, 2) + Math.Pow(x2.image, 2));
46           return x;
47       }
48    }
```

```

49     참조 0개
50     class EX1
51     {
52         참조 0개
53         public static void Main()
54         {
55             Complex c, c1, c2;
56             c1 = new Complex(1, 2);
57             c2 = new Complex(3, 4);
58             Console.WriteLine("c1 = {0}, c2 = {1}\n", c1.ToString(), c2.ToString());
59
60             Console.WriteLine("[덧셈 테스트]");
61             c = Complex.AddComplex(c1, c2);
62             Console.WriteLine("{0} + {1} = {2}\n", c1.ToString(), c2.ToString(), c.ToString());
63
64             Console.WriteLine("[뺄셈 테스트]");
65             c = Complex.SubComplex(c1, c2);
66             Console.WriteLine("{0} - {1} = {2}\n", c1.ToString(), c2.ToString(), c.ToString());
67
68             Console.WriteLine("[곱셈 테스트]");
69             c = Complex.MulComplex(c1, c2);
70             Console.WriteLine("{0} * {1} = {2}\n", c1.ToString(), c2.ToString(), c.ToString());
71
72             Console.WriteLine("[나눗셈 테스트]");
73             c = Complex.DivComplex(c1, c2);
74             Console.WriteLine("{0} / {1} = {2}\n", c1.ToString(), c2.ToString(), c.ToString());
75         }
76     }

```

(실행결과)

 Microsoft Visual Studio 디버그 콘솔

```

c1 = (1, 2i), c2 = (3, 4i)

[덧셈 테스트]
(1, 2i) + (3, 4i) = (4, 6i)

[뺄셈 테스트]
(1, 2i) - (3, 4i) = (-2, -2i)

[곱셈 테스트]
(1, 2i) * (3, 4i) = (-5, 10i)

[나눗셈 테스트]
(1, 2i) / (3, 4i) = (0.44, 0.08i)

C:\Users\nec\Desktop\닷넷프로그래밍_중간이후\닷넷프로그래밍_중간이후\bin\De
로그래밍_중간이후.exe(프로세스 11796개)이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...

```

C1객체, C2객체를 생성했다.

C1객체는 $1+2i$, C2객체는 $3+4i$ 이다. 이것을 가지고 덧셈, 뺄셈, 곱셈, 나눗셈을 해보니 교재에 있는 수식으로 정상적으로 계산되었다.

(설명)

우선 Complex 클래스를 선언한다. 교재에 나와있는데로 public 접근수정자로 실수형 변수 real와 image필드를 선언한다. 그리고 (코드6-10),(코드11-15)생성자 두개를 만든다. 첫번째 생성자는 한 개의 실수형 인수를 받아 real와 image 필드를 그 인수로 초기화하는 생성자이고, 두번째 생성자는 각 각 인수를 받아 real와 image 필드를 초기화한다. 여기서 this.real = real; 코드로 인수와 클래스 필드의 이름이 같더라도 this.를 통해 구별한다.

(코드16-19) ToString() 메소드를 만든다. 이 메소드는 하나의 복소수를 (실수부, 허수부) 형태로 반환하는 기능을 한다. 기본적으로 override public string ToString() 형태로 선언해준다. 그리고 return "(" + real + ", " + image + "i)"; 을 통해 (a, bi)형태의 string 형으로 반환한다.

(코드20-26) public static으로 두개의 Complex 객체를 더하는, 즉 두개의 복소수를 더하는 메소드를 만든다. 덧셈은 $(a + bi) + (c + di) = (a + c) + (b + d)i$ 이다. 반환형은 Complex가 온다. 그 후 매개변수 Complex x1, Complex x2를 두개 받는다. 즉 객체 x1, x2를 더하는 것이다. 우선 Complex 클래스 객체 x를 만들어 실수부, 허수부를 모두 0으로 생성자를 통해 초기화한다. 그리고 실수부는 실수부끼리, 허수부는 허수부끼리 모두 더한다. 그리고 더한 결과가 담긴 객체 x를 반환한다.

(코드27-33) public static으로 두개의 Complex 객체를 빼는, 즉 두개의 복소수를 빼는 메소드를 만든다. 뺄셈은 $(a + bi) - (c + di) = (a - c) + (b - d)i$ 이다.

(코드34-40) public static으로 두개의 Complex 객체를 곱하는, 즉 두개의 복소수를 곱하는 메소드를 만든다. 곱셈은 $(a + bi) * (c + di) = (ac - bd) + (ad + bc)i$ 이다.

(코드41-48) public static으로 두개의 Complex 객체를 나누는, 즉 두개의 복소수를 나누는 메소드를 만든다. 나눗셈은 $(a + bi)/(c + di) = \frac{ac+bd}{c^2+d^2} + \frac{bc-ad}{c^2+d^2}i$ 이다. 나눗셈의 경우 c^2 과 같은 연산이 등장한다. 그래서 Math.Pow(x2.real, 2) 코드를 통해 계산할 수 있다. 수학함수를 사용하기위해서 Math.을 사용한다. Pow함수의 경우 어떤수의 몇승을 계산할 수 있는 함수이다. 그래서 x2.real^2를 계산하는 것이다. 이와 같은 Math.Pow()함수를 사용해 복소수의 나눗셈을 계산한다.

(코드49~) 클래스 EX1 내 Main()에서 Complex c, c1, c2를 선언한다. c1의 경우 객체를 생성할 때 생성자를 통해 real필드를 1로, image필드를 2로 초기화한다. c2객체는 real필드를 3, image필드를 4로 초기화한다. 그리고 사용자가 만든 ToString()메소드를 통해 c1과 c2 객체의 실수부와 허수부를 출력한다.

덧셈, 뺄셈, 곱셈, 나눗셈을 테스트 한다. 계산된 결과 객체는 c에 반환되 c를 사용자가 만든 ToString()메소드를 통해 출력한다. 덧셈의 경우 두개의 객체를 더하기 위한 메소드가 static이므로 메소드를 사용하기 위해서 클래스 이름인 Complex.AddComplex로 호출한다. 뺄셈, 곱셈, 나눗셈도 마찬가지로 static이므로 같은 방식으로 메소드를 호출해 계산한다. 그리고 결과를 ToString()메소드로 출력한다.

4-13 스택

(소스코드캡처)

```
1      using System;
      참조 3개
2      class Stack
3      {
4          private int[] stack;
5          int sp = -1;
6          int size = 0;
      참조 1개
7          public int Size
8          {
9              get { return size; }
10         }
      참조 1개
11         public Stack(int size=100)
12         {
13             stack = new int[size];
14         }
      참조 1개
15         public void Push(int data)
16         {
17             stack[++sp] = data;
18             size++;
19         }
      참조 1개
20         public int Pop()
21         {
22             size--;
23             return stack[sp--];
24         }
25     }
      참조 0개
26     class EX2
27     {
      참조 0개
28         public static void Main()
29         {
30             Stack st = new Stack(10);
31
32             Console.WriteLine("[STACK PUSH]");
33             while(true)
34             {
35                 string buffer = Console.ReadLine();
36                 int inputNum = int.Parse(buffer);
37                 if (inputNum == 0) break;
38                 st.Push(inputNum);
39             }
40
41             Console.WriteLine("\n[STACK POP]");
42             while (st.Size != 0)
43             {
44                 Console.Write("{0} ", st.Pop());
45             }
46             Console.WriteLine();
47         }
48     }
```

(실행결과)

```
Microsoft Visual Studio 디버그 콘솔
[STACK PUSH]
1
2
3
10
20
30
100
200
300
0

[STACK POP]
300 200 100 30 20 10 3 2 1

C:\Users\wneec\Desktop\닷넷프로그래밍_중간이후\닷넷프로그래밍_중간이후.exe(프로세스 14816개)이(가) 종료되었습니다. 이 창을 닫으려면 아무 키나 누르세요...
```

입력을 1,2,3,10,20,30,100,200,300 순서대로 push했다. 스택이므로 마지막에 넣은 원소가 먼저 빠져나온다. 그래서 pop을 해주면 300,200,100,30,20,10,3,2,1 순서대로 출력된다.

(설명)

클래스 Stack을 선언한다. 클래스 Stack에는 private 접근수정자로 int형 배열선언을 하고 스택의 인덱스 위치를 나타내기위한 변수 sp와 스택의 크기를 나타내는 size변수를 private로 만든다. 그리고 sp = -1, size = 0으로 초기화한다.

(코드7-10) 프로퍼티를 만든다. Private 접근수정자로 생성된 size를 외부에서 접근하기 위함이다. 여기서는 겹-접근자만 구성했다. 그리고 외부에서는 객체이름.Size로 접근가능하게 만들었다. size를 늘이고 줄이는 것은 클래스 내 메소드에서 이루어지기 때문에 겹-접근자만 만들어도 된다.

(코드11-15)생성자 함수이다. 여기서 size를 디폴트 100으로 설정해 디폴트 생성자를 만들었다. 생성자 안에서는 stack = new int[size];를 통해 size크기의 int형 배열을 만들어준다.

(코드16-20) 아무것도 반환하지 않고 매개변수 data를 가지는 Push()함수이다. Push()함수는 매개변수 data를 객체 내 배열에 저장시킨다. 이때 sp 초기값이 -1이기 때문에 전위연산자 stack[++sp] = data;를 통해 먼저 sp를 1증가시키고 그 인덱스에 data를 저장시킨다. 그리고 size를 1 증가시킨다.

(코드21-25) int형을 반환하는 Pop()함수이다. 여기에서는 우선 size를 1 줄이고, return stack[sp--];를 통해 현재 sp 인덱스의 값을 반환시킨 후 sp 값을 1 감소시킨다.

(코드29~) 클래스 EX2 내 Main()메소드에서 Stack 클래스 객체 st를 선언해주고 크기는 10으로 설정한다. Stack st = new Stack(10); 을 통해 생성자함수가 스택크기를 10으로 만들어 줄 것이다. 그리고 while 무한반복문을 만들어 준다. 여기서 string 형 buffer 변수에 사용자로부터 숫자를 입력받고 int.Parse(buffer) 함수를 통해 입력받은 string형을 int형으로 형변환 해 inputNum변수에 넣는다. 그리고 만약 그 숫자가 0이라면 반복문을 바로 빠져나온다. 만약 그것이 아니라면 st.Push(inputNum);을 통해 st객체 내 정수형 배열에 inputNum을 저장한다. 그 후 출력을 위한 while 반복문을 만든다. 이때 조건은 st객체 내의 스택배열 크기가 0이 아닐때까지 계속반복한다. 즉 st 객체 내 스택배열 크기가 0이면 반복문을 중단한다. 반복문 안 코드를 살펴보면 st.Pop()을 통해 정수를 출력한다.

4-14 벡터

(소스코드캡처)

```
1    using System;
    참조 4개
2    class Vector
3    {
4        public int[] v;
        참조 0개
5        public Vector()
6        {
7            v = new int[100];
8        }
        참조 1개
9        public Vector(int size)
10       {
11           v = new int[size];
12       }
        참조 3개
13       static void Swap(ref int x, ref int y)
14       {
15           int temp = x;
16           x = y;
17           y = temp;
18       }
        참조 3개
19       public void Qsort(int left, int right)
20       {
21           int pe;
22           int i, last;
23
24           if (left >= right) return;
25           pe = (left + right) / 2;
26           Swap(ref v[left], ref v[pe]);
27           last = left;
28           for (i = left+1; i <= right; i++)
29               if (v[i] < v[left]) Swap(ref v[++last], ref v[i]);
30           Swap(ref v[left], ref v[last]);
31           Qsort(left, last - 1);
32           Qsort(left + 1, right);
33       }
34   }
    참조 0개
35   class EX3
36   {
        참조 0개
37       public static void Main()
38       {
39           Vector vector = new Vector(10);
40
41           Console.WriteLine("[vector 입력]");
42           int index = 0;
43           while (true)
44           {
45               string buffer = Console.ReadLine();
46               int inputNum = int.Parse(buffer);
47               if (inputNum == 0) break;
48               vector.v[index++] = inputNum;
49           }
50
51           vector.Qsort(0, index - 1);
52
53           Console.WriteLine("\n[정렬결과]");
54           for(int i = 0; i < index; i++)
55           {
56               Console.Write("{0} ", vector.v[i]);
57           }
58           Console.WriteLine();
59       }
60   }
```

(실행결과)

```
Microsoft Visual Studio 디버그 콘솔
[vector 입력]
1
5
3
-1
10
7
0

[정렬결과]
-1 1 3 5 7 10

C:\Users\wnech\Desktop\닷넷프로그래밍_중간이후\닷넷프로
(프로세스 15384개)이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...
```

입력한 숫자가 오름차순으로 정렬되어있다.

(설명)

클래스 Vector를 선언한다. 여기에 private로 필드 int형 배열 v를 선언한다.

(코드5-8) 매개변수 없는 생성자 함수를 만든다. v배열을 크기 100으로 생성한다.

(코드9-12) size매개변수가 있는 생성자 함수를 만들어 size크기의 배열 v를 생성한다.

(코드13-18) Swap함수는 참조에 의한 호출로 ref int x와 ref int y를 통해 x, y를 변경해준다. 그러기 위해서 임시변수 temp가 있다.

(코드19-33) 조건문에서 $left \geq right$ 라면 바로 해당 Qsort메소드를 빠져나간다. 그리고 피벗 변수 pe에 중앙자리를 설정한다. 그래서 $pe = (left + right) / 2$ 를 한다. 그리고 맨 왼쪽 값과 피벗값을 Swap함수를 통해서 바꿔준다. 그리고 $last = left$ 로 설정한다. 그리고 for반복문을 통해서 $i = left + 1$, 즉 피벗이 위치한 left 다음의 인덱스부터 맨 끝 인덱스 right까지 반복문이 돌면서 만약에 $v[i]$ 가 $v[left]$ (피벗값)보다 작으면 last를 1증가시키고 그곳에 있는 값과 i인덱스에 있는 값을 교환한다. 그리고 left인덱스에 있는 값 즉 피벗값과 last인덱스에 있는 값을 교환한다. 퀵정렬은 피벗값을 중심으로 작은값과 같거나 큰값을 분리해 위치시켜 $Qsort(left, last - 1); Qsort(left + 1, right);$ 를 호출한다. 즉 퀵정렬 알고리즘은 재귀호출을 통해서 이루어진다.

(코드37~) 클래스 EX3 내 Main()메소드에서는 Vector 클래스 객체 vector를 만든다. 이때 vector 객체 내 배열크기는 10으로 만들기 위해서 10을 매개변수로 설정하면 매개변수 있는 생성자함수가 호출된다. 객체 내 배열 v의 인덱스를 나타내는 변수 index 변수를 0으로 초기화시킨다.

그리고 while 무한 반복문을 만들어 string형변수 buffer에 사용자로부터 숫자를 입력받고 $int.Parse(buffer)$ 를 통해 buffer를 int형으로 형변환 시킨다. 그 결과를 int형 변수 inputNum에 저장한다. 만약 이 값이 0이라면 반복문을 빠져나가고 그것이 아니라면 vector객체 내 v배열에 인덱스 index에 값을 저장하고 후위연산자를 통해 인덱스를 한 개 증가시킨다.

그리고 vector 객체 내 메소드 Qsort함수를 호출해 vector내 v배열의 인덱스 left~right까지 퀵정렬을 진행해 오름차순으로 정렬한다. 이때 $Qsort(0, index-1);$ 을 한 이유는 처음 index를 0으로 초기화시키고 삽입 후 후위연산자로 1씩 증가시켰기 때문에 두번째 매개변수로 index-1을 설정한다.

그리고 for반복문을 $i=0$ 부터 1씩증가해 $i < index$ 일때까지 반복한다. 여기서는 $vector.v[i]$ 를 출력한다.