



| Basis and Practice in Programming

week 7



Function

- Why function?
 - Modularization
 - “Divide and conquer!”
- Four types of functions
 - With arguments, with return value
 - With arguments, without return value
 - Without arguments, with return value
 - Without arguments, without return value

Function

- A function with arguments and return value

```
int Add(int i, int j)
{
    int result = i+j;
    return result;
}
```

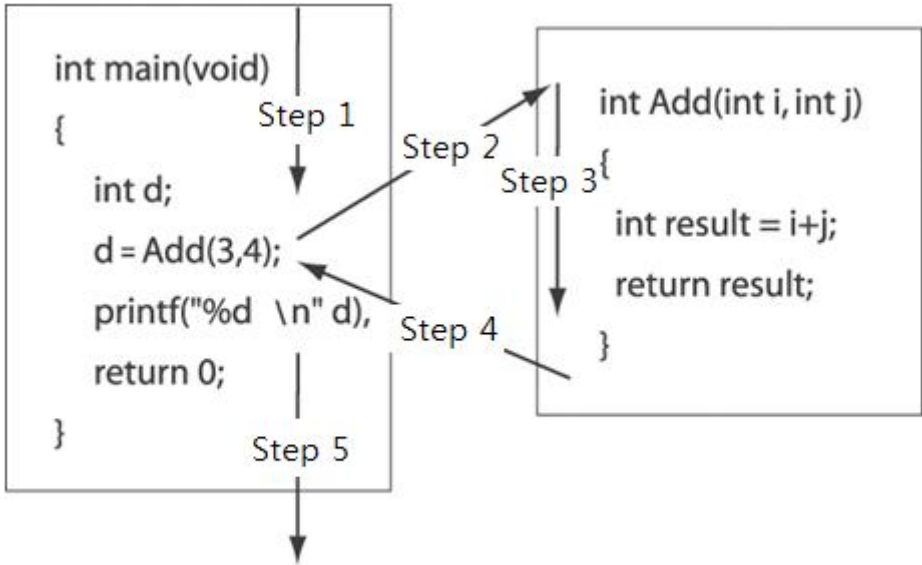
```
a int b Add c (int i, int j)
{
    int result = i+j;
d return result;
}
```

- a** return type
- b** function name
- c** arguments
- d** return result

Function

- A process of calling function

```
#include <stdio.h>
int Add(int i, int j)
{
    int result = i + j;
    return result;
}
int main(void)
{
    int d;
    d = Add(3, 4);
    printf("%d \n", d);
    return 0;
}
```



Function

- Diverse types of functions

```
void Intro(void)
{
    printf("***** START ***** \n");
    printf("Input two integer numbers : ");
}
```

```
int Input(void)
{
    int input;
    scanf("%d", &input);
    return input;
}
```

```
void Result_Print(int val)
{
    printf("Result of Addition: %d \n", val);
    printf("***** END ***** \n");
}
```

Function

- Declaration of functions
 - Functions should be pre-defined before being called

```
Int add(int a, int b)
{
    int result = a + b;
    return result;
}
```

```
Int main(void)
{
    add(3, 4);
    return 0;
}
```

```
int add(int a, int b);
```

```
Int main(void)
{
    add(3, 4);
    return 0;
}
```

```
Int add(int a, int b)
{
    int result = a + b;
    return result;
}
```

Scope of variables

- Local variable
 - Declared & used in { }
- Global variable
 - Not declared in functions
 - Accessible in a entire program
- Static variable
 - Can be declared both in internal and external of functions

Scope of variables

- Local variable

```
/* week 7 example 1 */
#include <stdio.h>

int main(void)
{
    int val=0;

    if(1)
    {
        int val=0;
        val+=10;
        printf("The value of local variable in if statement : %d \n", val);
    }

    printf("The value of local variable in main function : %d \n", val);

    return 0;
}
```


Scope of variables

- Global variable

```
/* week 7 example 2 */  
  
#include <stdio.h>  
  
int val;  
  
void add(int num);  
  
int main(void)  
{  
    printf("val : %d \n", val);  
  
    add(3);  
    printf("val : %d \n", val);  
  
    val++;  
    printf("val : %d \n", val);  
  
    return 0;  
}
```

```
void add(int n)  
{  
    val+=n;  
}
```

Scope of variables

- Global variable

```
/* week 7 example 3 */
#include <stdio.h>

int val=0;

void fct(void);

int main(void)
{
    val=10;
    printf("val : %d \n", val);

    fct();
    printf("val : %d \n", val);

    return 0;
}
```

```
void fct(void)
{
    int val=20;
    val++;
}
```

Scope of variables

- Static variable

```
/* week 7 example 4 */
#include <stdio.h>

void fct(void);

int main(void)
{
    int i;

    for(i=0; i<5; i++)
        fct();

    return 0;
}

void fct(void)
{
    int val=0;
    val++;
    printf("%d ",val);
}
```

```
/* week 7 example 5 */
#include <stdio.h>

void fct(void);

int main(void)
{
    int i;

    for(i=0; i<5; i++)
        fct();

    return 0;
}

void fct(void)
{
    static int val=0;
    val++;
    printf("%d ",val);
}
```

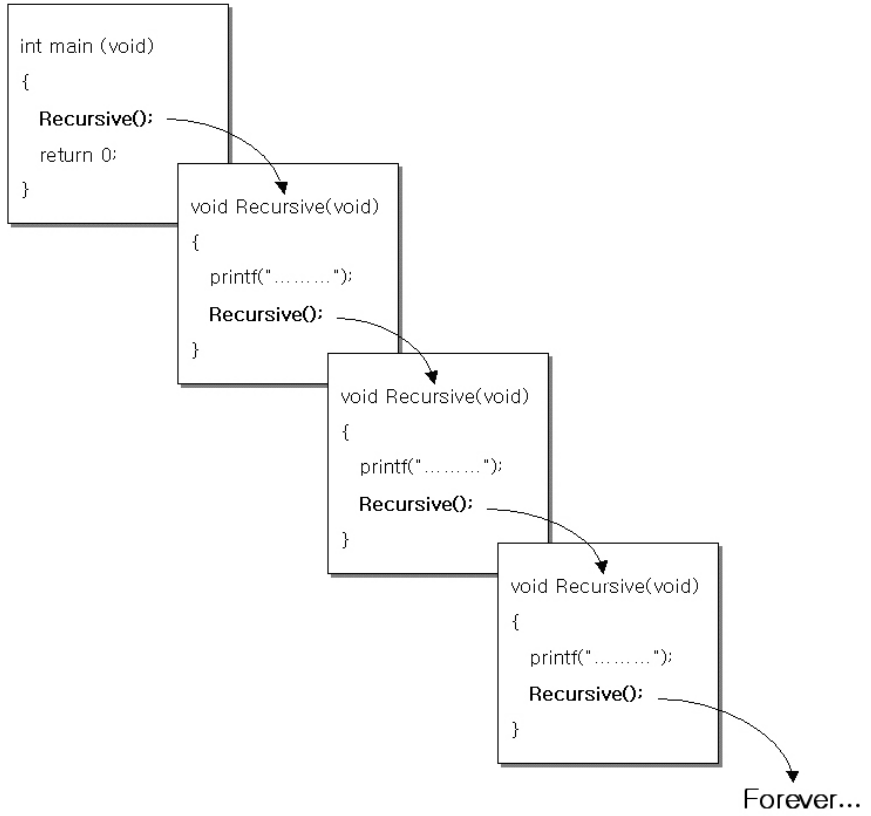
Scope of variables

- Recursive function call
 - A function which calls itself

```
#include <stdio.h>

void Recursive(void)
{
    printf("Recursive Call! \n");
    Recursive();
}

int main(void)
{
    Recursive();
    return 0;
}
```



Scope of variables

- Recursive function call

```
/* week 7 example 6 */
#include <stdio.h>

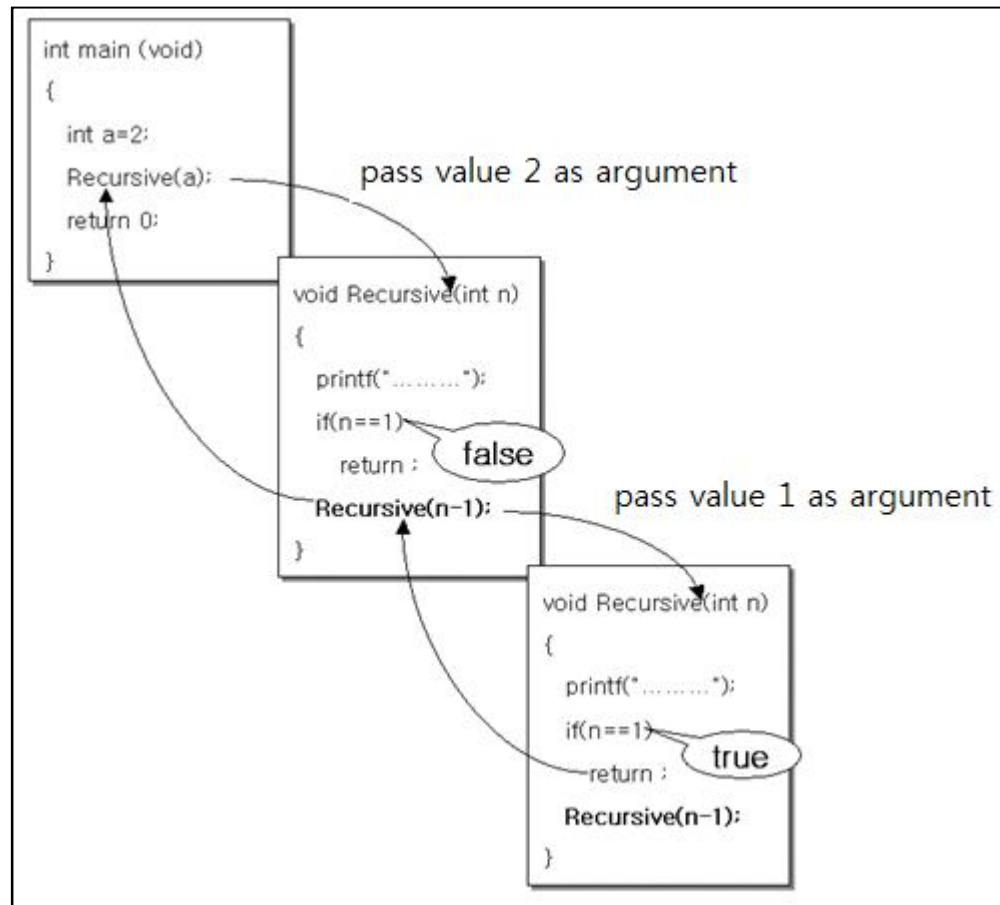
void Recursive(int n)
{
    printf("Recursive Call! \Wn");

    if(n==1)
        return;

    Recursive(n-1);
}

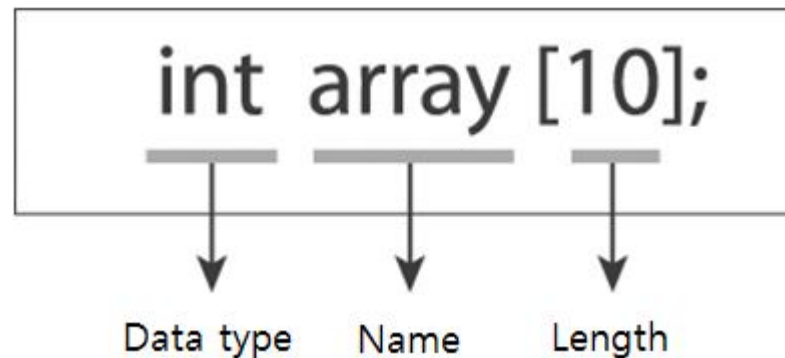
int main(void)
{
    int a=2;
    Recursive(a);

    return 0;
}
```



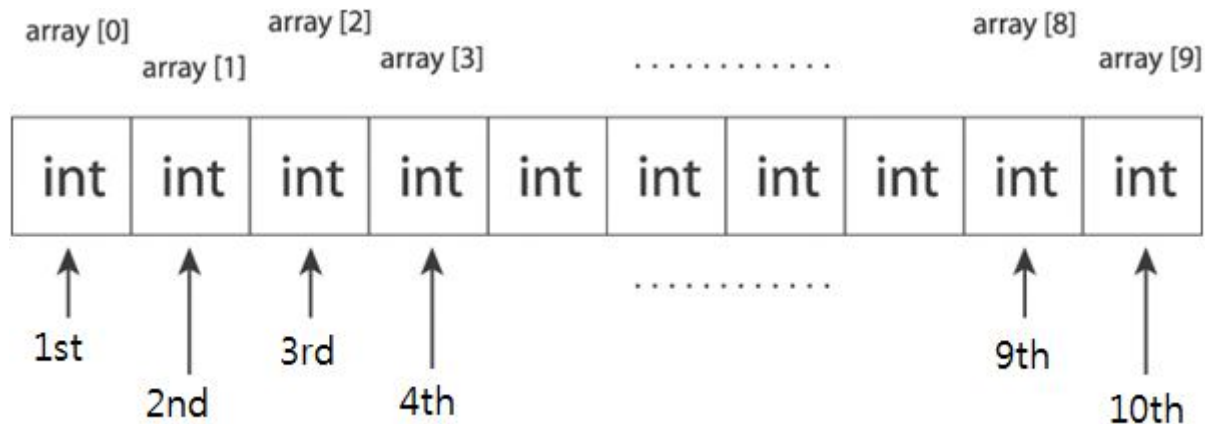
Array basic

- Declare more than one variables which are the same type
- Three elements in declaration of array
 - Length
 - Data type
 - Name



Array basic

- Access 1-dimension array
 - By index : the position of a element on array
 - Index starts with 0



Array basic

- Initialization with declaration

```
int main(void)
{
    int arr1 [5]={1, 2, 3, 4, 5}; ➔ 

|   |   |   |   |   |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|


    int arr2[ ]={1, 3, 5, 7, 9}; ➔ 

|   |   |   |   |   |
|---|---|---|---|---|
| 1 | 3 | 5 | 7 | 9 |
|---|---|---|---|---|


    int arr3[5]={1, 2} ➔ 

|   |   |   |   |   |
|---|---|---|---|---|
| 1 | 2 | 0 | 0 | 0 |
|---|---|---|---|---|


}
```


Array basic

```
/* week 7 example 7*/  
  
#include <stdio.h>  
  
int main(void)  
{  
    double total;  
    double val[5];  
  
    val[0]=1.01;  
    val[1]=2.02;  
    val[2]=3.03;  
    val[3]=4.04;  
    val[4]=5.05;  
  
    total=val[0]+val[1]+val[2]+val[3]+val[4];  
    printf("Average : %lf \n", total/5);  
  
    return 0;  
}
```

```
/* week 7 example 8*/  
  
#include <stdio.h>  
  
int main(void)  
{  
    double total;  
    double val[5]={1.01, 2.02, 3.03, 4.04, 5.05};  
  
    total=val[0]+val[1]+val[2]+val[3]+val[4];  
    printf("Average : %lf \n", total/5);  
  
    return 0;  
}
```

Practice

- Week 7 practice
 - Write a program that calculate 2^n with recursive function call
 - Input number n from user
 - Print the result of 2^n
 - You should solve it with recursive approach
 - ex) when input value : 8, print 256 ($2^8 = 256$)
 - Due date : Today 11:59 PM