Intermediate Programming

— Recursion (1) —

Waseda Univ.

Today's Topics

- Understanding recursive call of functions
  - Recursive functions
  - Some examples of the recursive call
  - Recursion elimination

Recursive call

What is recursive call?

- When some operations are calculated/executed by a function, the C language allows the function to call itself within the same program. This is called the recursive call of a function.
- A recursive function has a trivial result (without recurring) and one or more recursive cases (calls itself).

Advantages and disadvantages

- 【Advantage】 Make the code simple.
- 【Disadvantage】 Difficult for implementation.
- 【Disadvantage】 Performance issues and requires stack space.

Simple example of the recursive call

```
#include <stdio.h>

void recursion( int i ){
  if(i < 10){
    recursion(i+1);
    printf("%d",i);
  }
}
int main(void){
  recursion(0);
  return 0;
}
```

This program prints

9 8 7 6 5 4 3 2 1 0

on the screen.

How to work this program

- First, this program calls the “recursion” function with the parameter 0.
- Since 0 is less than 10, the “recursion” function calls itself with the parameter i+1(=1).
- Next, the “recursion” function is also called with the parameter 2.
- The “recursion” function further calls itself until the parameter becomes 10.
- When the parameter is 10, the “recursion” function returns from the recursive call.
- The “recursion” function prints “9” on the screen and returns.
- This process repeats. The program ends when all “recursion” functions returns from the recursive call.

Example of the recursive call 1

Factorial

- $1! = 1$
- $2! = 1! \times 2$
- $3! = 2! \times 3 = (1! \times 2) \times 3$

For general,

$$\begin{aligned} n! &= (n-1)! \times n \\ &= (n-2)! \times (n-1) \times n \\ &= \cdots \end{aligned}$$

```
#include <stdio.h>

int factorial( int n ){
    int m;
    if( n==0 || n==1 ){
        printf("1");
        return 1;
    }else{
        printf("%d *(",n);
        m = n * factorial( n-1 );
        printf(")");
        return m;
    }
}

int main(void){
    N= 5;
    factorial(N);
    return 0;
}
```

Example of the recursive call 1

Factorial

- $1! = 1$
- $2! = 1! \times 2$
- $3! = 2! \times 3 = (1! \times 2) \times 3$

For general,

$$\begin{aligned} n! &= (n-1)! \times n \\ &= (n-2)! \times (n-1) \times n \\ &= \cdots \end{aligned}$$

```
#include <stdio.h>

int factorial( int n ){
    int m;
    if( n==0 || n==1 ){
        printf("1");
        return 1;
    }else{
        printf("%d *(",n);
        m = n * factorial( n-1 );
        printf(")");
        return m;
    }
}

int main(void){
    N= 5;
    factorial(N);
    return 0;
}
```

Example of the recursive call 2

Fibonacci sequence

- $f(0) = 0$
- $f(1) = 1$
- $f(2) = f(1) + f(0) = 1$
- $f(3) = f(2) + f(1) = 2$
- $f(n) = f(n-1) + f(n-2)$

```
int fibonacci( int n ){
    if( n==0 ){
        return 0;
    } else if( n == 1){
        return 1;
    } else {
        return fibonacci(n-1)+fibonacci(n-2);
    }
}
```

Example of the recursive call 2

Fibonacci sequence

- $f(0) = 0$
- $f(1) = 1$
- $f(2) = f(1) + f(0) = 1$
- $f(3) = f(2) + f(1) = 2$
- $f(n) = f(n-1) + f(n-2)$

```
int fibonacci( int n ){
    if( n==0 ){
        return 0;
    } else if( n == 1){
        return 1;
    } else {
        return fibonacci(n-1)+fibonacci(n-2);
    }
}
```

Recursion elimination

- The recursive programs above can be described without using the recursive call by making use of the looping (for-statement).
- Recursion elimination is the process by which a recursive call is translated to iteration.
- In C language, the iterative program is preferred because it avoids the overhead of recursive function calls.

Example of recursion elimination 1

【Recursive function】

```
int factorial( int n ){
    int m;
    if( n==0 || n==1 ){
        printf("1");
        return 1;
    }else{
        printf("%d *(",n);
        m = n * factorial( n-1 );
        printf(")");
        return m;
     }
 }
```

【Iterative function】

```
int factorial( int n ){
    int i, m=1;

    printf("1");

    for( i=2; i<=n; i++){
        printf("*%d",i );
        m = m * i;
    }
    return m;

 }
```

Example of recursion elimination 2

【Recursive function】

```
int fibonacci( int n ){
    if( n==0 ){
        return 0;
    } else if( n == 1){
        return 1;
    } else {
        return fibonacci(n-1)+fibonacci(n-2);
    }
}
```

【Iterative function】

```
int fibonacci( int n ){
    int i, fn, fn1=1, fn2=0;
    if( n==0 ){
        return 0;
    } else if( n == 1){
         return 1;
    } else {
         for( i=2; i<=n;i++){
              fn=fn1+fn2;   fn2=fn1;    fn1=fn;
         }
         return fn;
    }
}
```

Exercise

Exercise: combination.c

- Write a program that calculates the number of combinations from n things taken r: ${}_nC_r$. The ${}_nC_r$ is defined by

$$ {}_nC_r = {}_{n-1}C_{r-1} + {}_{n-1}C_r,\ {}_nC_0 = {}_nC_n = 1,\ {}_nC_1 = n $$

- Output of this program is as follows:

```
Input n : 20
Input r : 3
 20 C 3 = 1140
```

Answer

```
#include <stdio.h>



int main(void){
  int n, r;

  printf("Input n : ");
  scanf("%d",&n);

  printf("Input r : ");
  scanf("%d",&r);

  printf(" %d C %d =%d \n",n,r,combination(n,r));

  return 0;
}
```

Answer

```
#include <stdio.h>



int main(void){
  int n, r;

  printf("Input n : ");
  scanf("%d",&n);

  printf("Input r : ");
  scanf("%d",&r);

  printf(" %d C %d =%d \n",n,r,combination(n,r));

  return 0;
}
```

Answer

```
#include <stdio.h>



int main(void){
  int n, r;

  printf("Input n : ");
  scanf("%d",&n);

  printf("Input r : ");
  scanf("%d",&r);

  printf(" %d C %d =%d \n",n,r,combination(n,r));

  return 0;
}
```

Answer

```
#include <stdio.h>

int combination(int n, int r){

}

int main(void){
  int n, r;

  printf("Input n : ");
  scanf("%d",&n);

  printf("Input r : ");
  scanf("%d",&r);

  printf(" %d C %d =%d \n",n,r,combination(n,r));

  return 0;
}
```

Answer

```
#include <stdio.h>

int combination(int n, int r){
  if(r==0||r==n){ // nC0 = nCn = 1
    return 1;
  }

}

int main(void){
  int n, r;

  printf("Input n : ");
  scanf("%d",&n);

  printf("Input r : ");
  scanf("%d",&r);

  printf(" %d C %d =%d \n",n,r,combination(n,r));

  return 0;
}
```

Answer

```
#include <stdio.h>

int combination(int n, int r){
  if(r==0||r==n){ // nC0 = nCn = 1
    return 1;
  }
  if(r==1){       // nC1 = n
    return n;
  }

}

int main(void){
  int n, r;

  printf("Input n : ");
  scanf("%d",&n);

  printf("Input r : ");
  scanf("%d",&r);

  printf(" %d C %d =%d \n",n,r,combination(n,r));

  return 0;
}
```

Answer

```
#include <stdio.h>

int combination(int n, int r){
  if(r==0||r==n){ // nC0 = nCn = 1
    return 1;
  }
  if(r==1){       // nC1 = n
    return n;
  }
  return combination(n-1,r-1)+combination(n-1,r);
}

int main(void){
  int n, r;

  printf("Input n : ");
  scanf("%d",&n);

  printf("Input r : ");
  scanf("%d",&r);

  printf(" %d C %d =%d \n",n,r,combination(n,r));

  return 0;
}
```

Summary

- Understanding recursive call of functions
  - Recursive functions
  - Some examples of the recursive call
  - Recursion elimination