Computer programming

Lectures 6

Why use functions

Question:

- **If the program has many functions and a large scale, writing all the code in the main function will make the main function complex, confusing, and difficult to read and maintain**
- **Sometimes when a program needs to implement a certain function multiple times, it needs to repeatedly write the program code that implements this function, which makes the program lengthy and not concise**

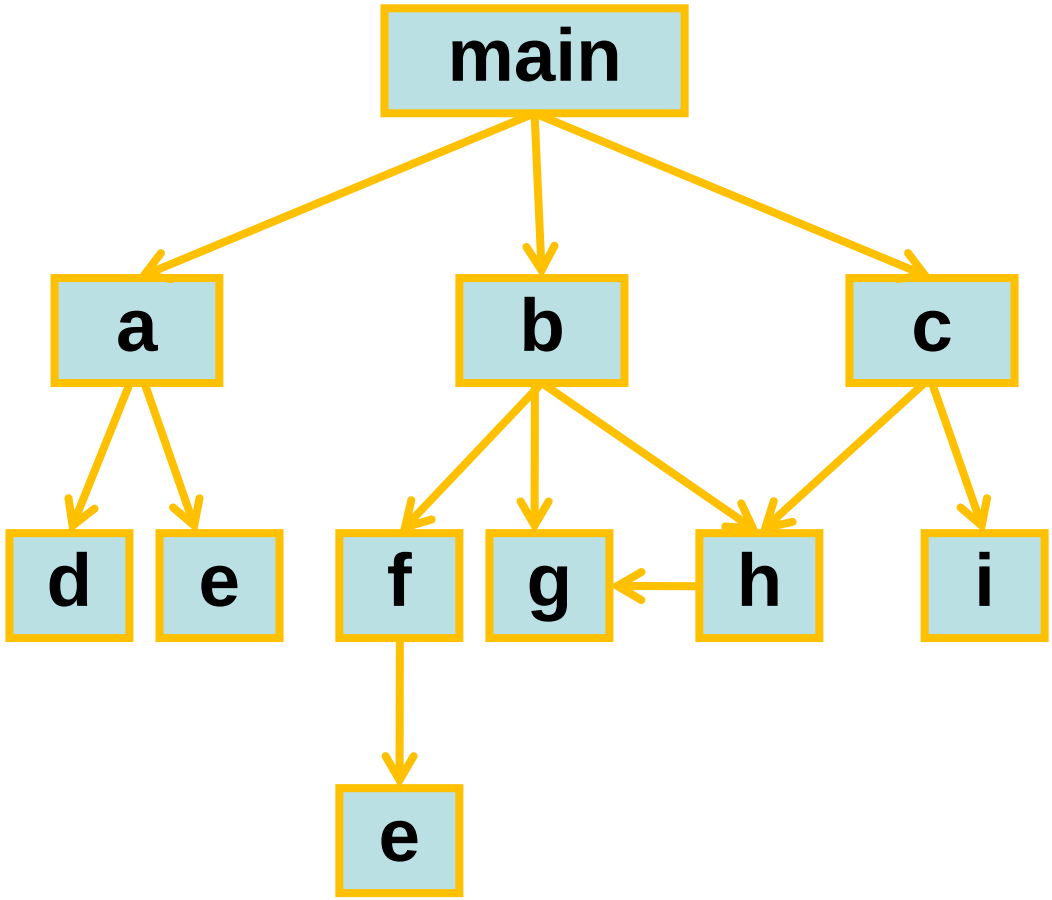
Solution:

- **Using a modular programming approach**
- **Simplify the process of program design by using the method of "assembly"**
- **Prepare a batch of functions to implement various functions in advance**
- **Save them in the function library and use them directly when needed**

C programs can be composed of a main function and several other functions

The main function calls other functions, and other functions can also call each other

The same function can be called any number of times by one or more functions



[example]Output the following results and implement them using function

```
**********
How do you do!
**********
```

Solution ideas:

There is a line of "*" above and below the output text, obviously there is no need to write this code repeatedly. Use a function print_Star to achieve the function of outputting a line of "*".

Write another print_message function outputs the middle line of text information

Call these two functions separately using the main function

```
#include <stdio.h>
int main()
{ void print_star();
   void print_message();
   print_star();   print_message();
   print_star();
   return 0;
}
void print_star()
{  printf("**********\n"); }
void print_message()
{  printf("  How do you do!\n");  }
```

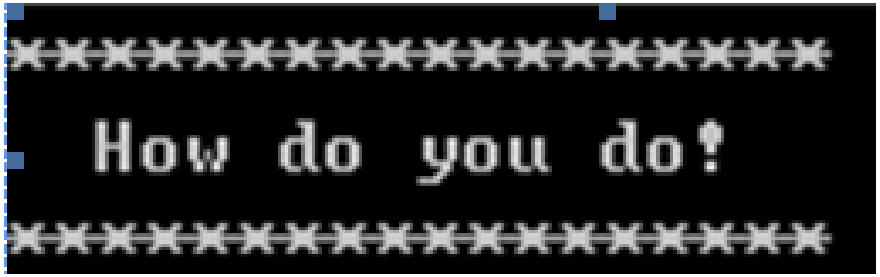
```
#include <stdio.h>
int main()
{  void print_star();
   void print_message();
   print_star();   print_message();
   print_star();
   return 0;
}
void print_star()
{  printf("**********\n"); }
void print_message()
{  printf("  How do you do!\n");  }
```

Declare functions

Call functions

Function Definition

```
#include <stdio.h>
int main()
{ void print_star();
  void print_message();
  print_star();   print_message();
  print_star();
  return 0;
}
void print_star()
{  printf("**********\n"); }
void print_message()
{  printf("  How do you do!\n");  }
```



Explanation:

1.A C program consists of one or more program modules, each of which serves as a source program file. For larger programs, it is generally not desirable to put all the content in one file, but rather to separate them into several source files, which form a C program. This facilitates separate writing and compilation, improving debugging efficiency. A source program file can be shared by multiple C programs.

2. A source program file consists of one or more functions and other related content (such as preprocessing instructions, data declarations and definitions, etc.). A source program file is a compilation unit that is compiled in units of the source program file during program compilation, rather than functions.

3.The execution of C program starts from the main function. If other functions are called in the main function, the process returns to the main function after the call, and the whole program runs in the main function.

Explanation:

4.All functions are parallel, meaning they are defined separately and independently of each other. A function does not belong to another function, that is, a function cannot be nested and defined. Functions can call each other, but the main function cannot be called. The main function is called by the operating system.

5.From the perspective of user usage, there are two types of functions. Library functions are provided by the system, and users do not need to define them themselves to directly use them. It should be noted that the number and functionality of library functions provided by different C language compilation systems may vary, although many basic functions are common. User defined functions. It is a function used to address user specific needs.

6.From the form of functions, there are two types of functions.① No parameter function. Non parametric functions are generally used to perform a specified set of operations. Non parametric functions can bring back or not bring back function values, but generally they do not bring back function values.② Parametric function. When calling a function, the calling function passes data to the called function through parameters. Generally, when executing the called function, a function value is obtained for the calling function to use.

The Definition of Function Structure

```
Return value type function name (formal parameter)
{
    Function body
}
```

```
int max(int x,int y)
{
  int z;
  if (x > y) z = x;
  else z = y;
  return(z);
}
```

No parameter function

```
Return value type function name ()
{
    Function body
}
```

```
Return value type function name (void)
{
    Function body
}
```

empty function

```
void function name (  )
  {          }
```

[Question] Calculate the radius and area of a circle based on the original value of radius r.

problem analysis
1.Function name. Area, indicating the name and meaning
2.The type of function return value. Double type
3.The parameters of the function. The formal parameter of the function is double r
4.Function body. That is, area=π * r * r

```
#define PI 3.14
double area(double r)
{  double s;
s= PI*r*r;
return s;}
```

Similarly, the circumference function of a circle is defined as

```
double circle(double r)
{
    doube l=2*PI*r;
    return l;
}
```

The general form of function calls is:

Function name (actual parameters)

- **If calling a no parameters function, the actual parameters can be omitted, but the parentheses cannot be omitted**
- **If the argument table column contains multiple actual parameters, the parameters are separated by commas**

The execution process of function call statements

- **locates the function definition by comparing the function name.**
- **Actual parameters pass values to their corresponding formal parameters.**
- **Execute the statements in the function body.**
- **returns to the function call statement**

```
#include <stdio.h>
void main()
{
   double r,s,l;
   double area(double r);
   double circle(double r);
   scanf("%lf",&r);
   s=area( r);
   l=circle(r);
   printf("area is %lf,s);
   printf("circle is %lf",l);
  }
```

```
#define PI 3.14
area(double r)
{  double s;
    s= PI*r*r;
  return s;
}
double circle(double r)
{
     double l;
      l=2*PI*r;
      return l;
}
```

[Question] Enter two integers and output the larger one

Solution ideas:

(1)The function name should be 'see the name to know the meaning', and now it is named 'max‘

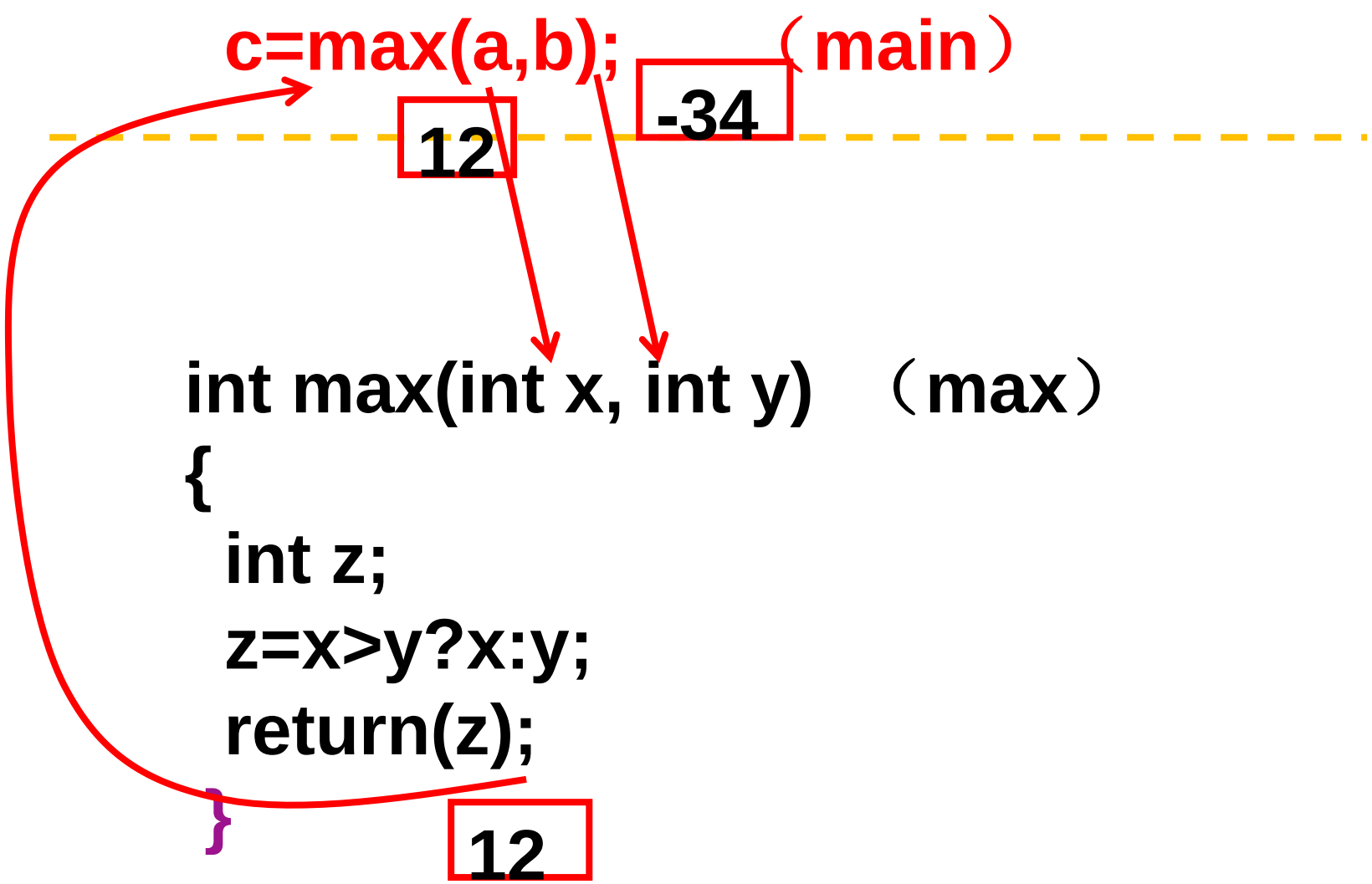
(2)Since the given two numbers are integers, the value of the calling function (i.e. the larger number) returned should be of integer type

(3)The max function should have two parameters in order to receive two integers from the main function, so the type of the parameters should be integer

```
#include <stdio.h>
int main()
{ int max(int x,int y);   int a,b,c;
   printf("two integer numbers: ");
   scanf("%d,%d",&a,&b);
   c=max(a,b);
   printf("max is %d\n",c);
}

int max(int x,int y)
{
    int z;
    z=x>y?x:y;
    return(z);
}
```

The actual parameter can be a constant, variable, or expression


c=max(a,b);
（main）
12
-34
int max(int x, int y)
（max）
{
int z;
z=x>y?x:y;
return(z);
}
12

return statement of a function

1. To return a value, you need to use key words “return” to return it to the calling function, and the return value type needs to be consistent with the function value type. If no return value is required, there is no need for a return statement, and the function type must be void.
2. Usually, it is hoped that the calling function can obtain a certain value through function calls, which is the function value (the return value of the function). There can be more than one return statement in a function, and whichever return statement is executed will take effect. The brackets after the return statement can be omitted.
3. If the type of the function value does not match the value of the expression in the return statement, a data type Data type forced conversion will occur, converting the value of the expression in the return statement to the function type.

```
#include <stdio.h>
int main()
 { int max(float x,float y);
    float a,b;  int c;
    scanf("%f,%f,",&a,&b);
    c=max(a,b);
    printf("max is %d\n",c);
    return 0;
 }
int max(float x,float y)
 {  float z;
     z=x>y?x:y;
     return( z ) ;
}
```

变为2

```
1.5,2.6
max is 2
```

1.5　2.6

2.6 ⇒ 2

There are two general forms of function prototypes:

For example

```
float add (float x, float y);
float add (float, float);
```

The prototype description can be placed at the beginning of the file, where all functions can use this function

Nested calls to functions

Calling another function from one function requires the following conditions:

1.The called function must be an already defined function (either a library function or a user-defined function)

2.If using library functions, the corresponding # include directive should be added at the beginning of this file

3.If you use a self-defined function, and the position of the function is after the calling function, you should declare

[Question] Enter two real numbers and use a function to find their sum.

```
#include <stdio.h>
int main()
{ float add(float x, float y);
   float a,b,c;
   printf("Please enter a and b:");
   scanf("%f,%f",&a,&b);
   c=add(a,b);
   printf("sum is %f\n",c);
   return 0;
}
```

Declare the add function

function value is float

```
float add(float x,float y)
{ float z;
   z=x+y;
   return(z);
}
```

[Question] finding the maximum number among four integers

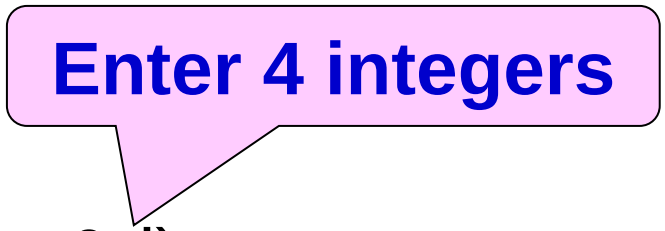
Solution ideas:

- **The max4 function is called in main to find the largest of the four**
- **Call max2 again in max4, find the larger of the two numbers**
- **Call max2 multiple times in max4, find the larger of the four numbers, and then return it as the function value to the main function**
- **Output results in the main function**

Declare the max4 function

```
#include <stdio.h>
int main()
{ int max4(int a,int b,int c,int d);
   int a,b,c,d,max;
   printf(“4 interger numbers:");
   scanf("%d%d%d%d",&a,&b,&c,&d);
   max=max4(a,b,c,d);
   printf("max=%d \n",max);
   return 0;
}
```

```
#include <stdio.h>
int main()
{ int max4(int a,int b,int c,int d);
   int a,b,c,d,max;
   printf(“4 interger numbers:");
   scanf("%d%d%d%d",&a,&b,&c,&d);
   max=max4(a,b,c,d);
   printf("max=%d \n",max);
   return 0;
}
```



```
#include <stdio.h>
int main()
{ int max4(int a,int b,int c,int d);
   int a,b,c,d,m
   printf(“4 inte
   scanf("%d%d%d%
   max=max4(a,b,c,d);
   printf("max=%d \n",max);
   return 0;
}
```

After being called, it must be the largest of the four numbers

Output Largest

Max4 function

```
int max4(int a,int b,int c,int d)
{  int max2(int a,int b);
    int m;
    m=max2(a,b);
    m=max2(m,c);
    m=max2(m,d);
    return(m);
}
```

Declare the max2 function

Max4 function

```
int max4(int a,int b,int c,int d)
{  int max2(int a,int b);
    int m;
    m=max2(a,b);
    m=max2(m,c);
    m=max2(m,d);
    return(m);
}
```

The larger of a and b

The larger of a b c

The larger of a b c d

Max4 function

```
int max4(int a,int b,int c,int d)
{  int max2(int a,int b);
    int m;
    m=max2(a,b);
    m=max2(m,c);
    m=max2(m,d);
    return(m);
}
```

找a,b中较大者

max2函数

```
int max2(int a,int b)
{  if(a>=b)
       return a;
   else
       return b;
}
```

Max4 function

```
int max4(int a,int b,int c,int d)
{  int max2(int a,int b);
    int m;
    m=max2(a,b);
    m=max2(m,c);
    m=max2(m,d);
    return(m);
}
```

Max2 function

```
int max2(int a,int b)
{  if(a>=b)
       return a;
   else
       return b;
}
```

return(a>b?a:b);

Max4 function

```
int max4(int a,int b,int c,int d)
{  int max2(int a,int b);
    int m;
    m=max2(a,b);
    m=max2(m,c);
    m=max2(m,d);
    return(m);
}
```

```
int max2(int a,int b)
{    return(a>b?a:b);   }
```

Max4 function

```
int max4(int a,int b,int c,int d)
{  int max2(int a,int b);
    int m;
    m=max2(a,b);
    m=max2(m,c);
    m=max2(m,d);
    return(m);
}
```

m=max2(max2(a,b),c);

```
int max2(int a,int b)
{    return(a>b?a:b);   }
```

Max4 function

```
int max4(int a,int b,int c,int d)
{  int max2(int a
    int m;
    m=max2(a,b);
    m=max2(m,c);
    m=max2(m,d);
    return(m);
}
```

m=max2(max2(max2(a,b),c),d);

```
int max2(int a,int b)
{    return(a>b?a:b);   }
```

Max4 function

```
int max4(
{  int max2(int a,i
    int m;
    m=max2(a,b);
    m=max2(m,c);
    m=max2(m,d);
    return(m);
}
```

ruturn max2(max2(max2(a,b),c),d);

```
int max2(int a,int b)
{    return(a>b?a:b);   }
```

```
4 interger numbers:12 45 -6 89
max=89
```

```
#include <
int main()
{  ……
    max=max4(a,b,c,d);
   ……
}
```

```
int max4(int a,int b,int c,int d)
{  int max2(int a,int b);
    ruturn max2(max2(max2(a,b),c),d);
}
```

```
int max2(int a,int b)
{    return(a>b?a:b);   }
```

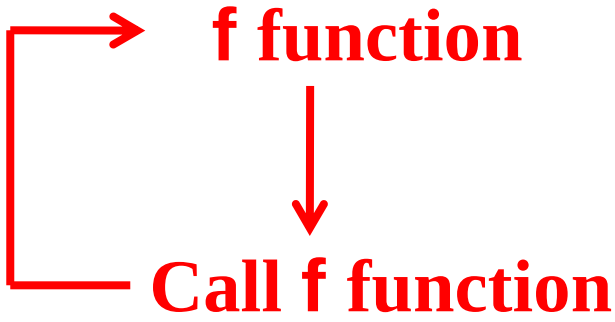
[Problem] Swap the values of two integer variables.

```
void swapValue(int x,int y)
{
    int temp;
    temp=x;
    x=y;
    y=temp;
    printf(“x=%d,y=%d",x,y);
 }
#include <stdio.h>
void main()
{
    int a=10,b=20;
    swapValue(a,b);
     printf("a=%d,b=%d",a,b);
 }
```

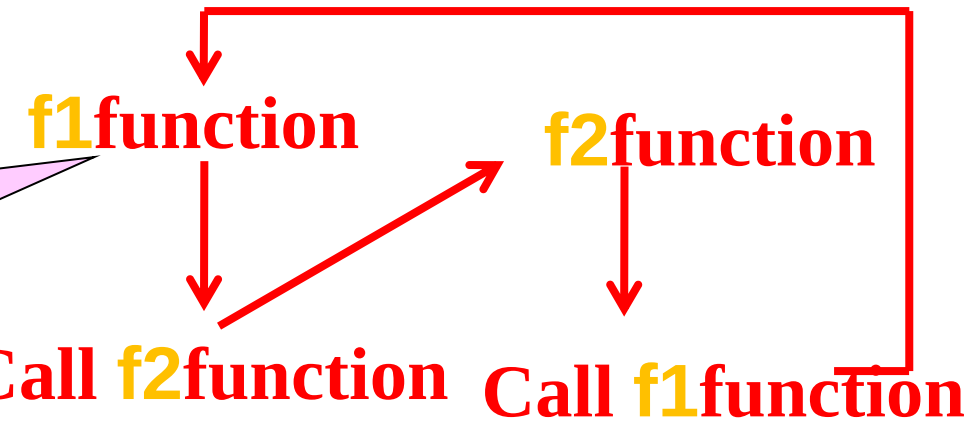
recursive calling of functions

```
int f(int x)
{
    int y,z;
    z=f(y);
    return (2*z);
}
```

Directly call



indirect call



The if statement should be used to control the end call

[Question] There are 5 students sitting together
How old is the fifth student? He said he was two years older than the fourth student
Ask the fourth student the age, he said he is two years older than the third student
Ask the third student and say he said two years older than the second student
Ask the second student, saying he is two years older than the first student
Finally, asked the first student who said he was 10 years old
May I ask how old the fifth student is

Solution ideas:

- **To require a fifth age, one must first know the fourth age**
- **Require that the fourth age must first be known as the third age**
- **The third age depends on the second age**
- **The second age depends on the first age**
- **Each student is 2 years older than their previous student**

age(5)=age(4)+2

age(4)=age(3)+2

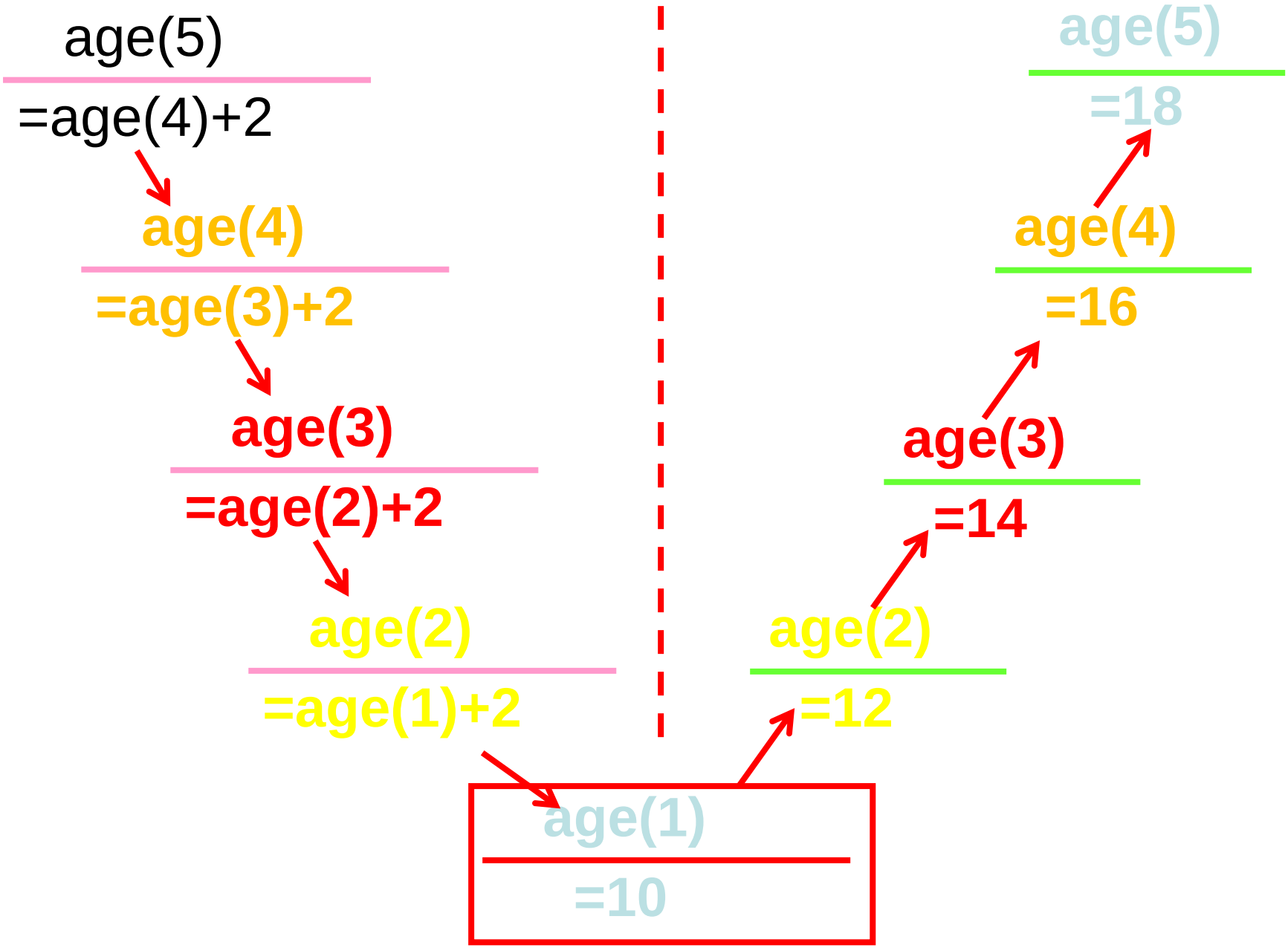
age(3)=age(2)+2

age(2)=age(1)+2

age(1)=10

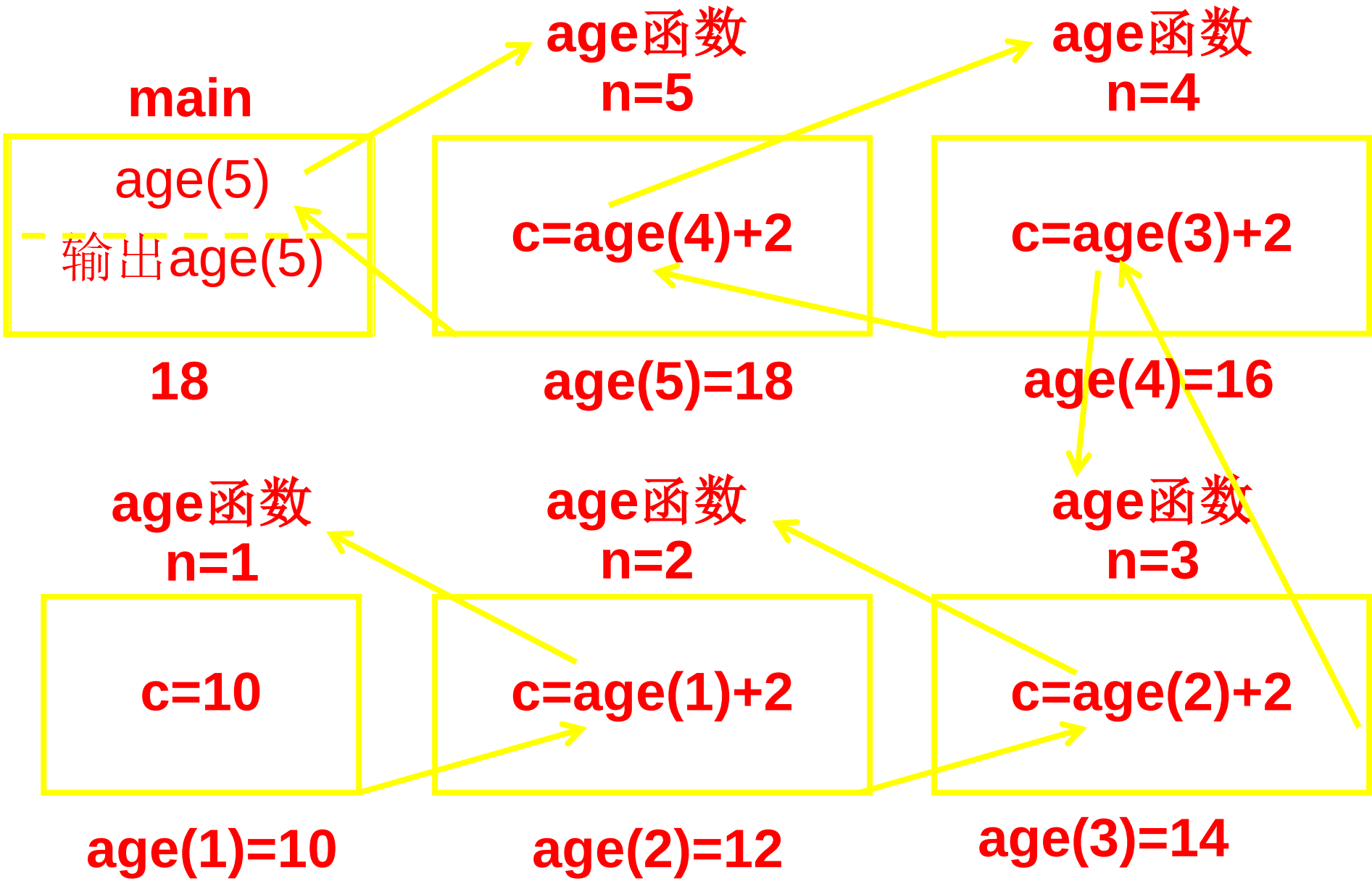
$$age(n) = 10 \qquad (n = 1)$$

$$age(n) = age(n-1) + 2 \qquad (n > 1)$$


age(5)
=age(4)+2
age(4)
=age(3)+2
age(3)
=age(2)+2
age(2)
=age(1)+2
age(1)
=10
age(2)
=12
age(3)
=14
age(4)
=16
age(5)
=18

```
#include <stdio.h>
int main()
{ int age(int n);
   printf("NO.5,age:%d\n",age(5));
   return 0;
}
int age(int n)
{ int c;
   if(n==1)   c=10;
   else    c=age(n-1)+2;
   return(c);
}
```

```
NO.5,age:18
```


main
age(5)
输出age(5)
18
age函数
n=5
c=age(4)+2
age(5)=18
age函数
n=4
c=age(3)+2
age(4)=16
age函数
n=3
c=age(2)+2
age(3)=14
age函数
n=2
c=age(1)+2
age(2)=12
age函数
n=1
c=10
age(1)=10