C语言程序设计

第6章　函数与预处理命令

北京科技大学　计算机系

2020/4/2

本章主要内容

概述

函数的定义

函数的调用

函数的嵌套调用和递归调用

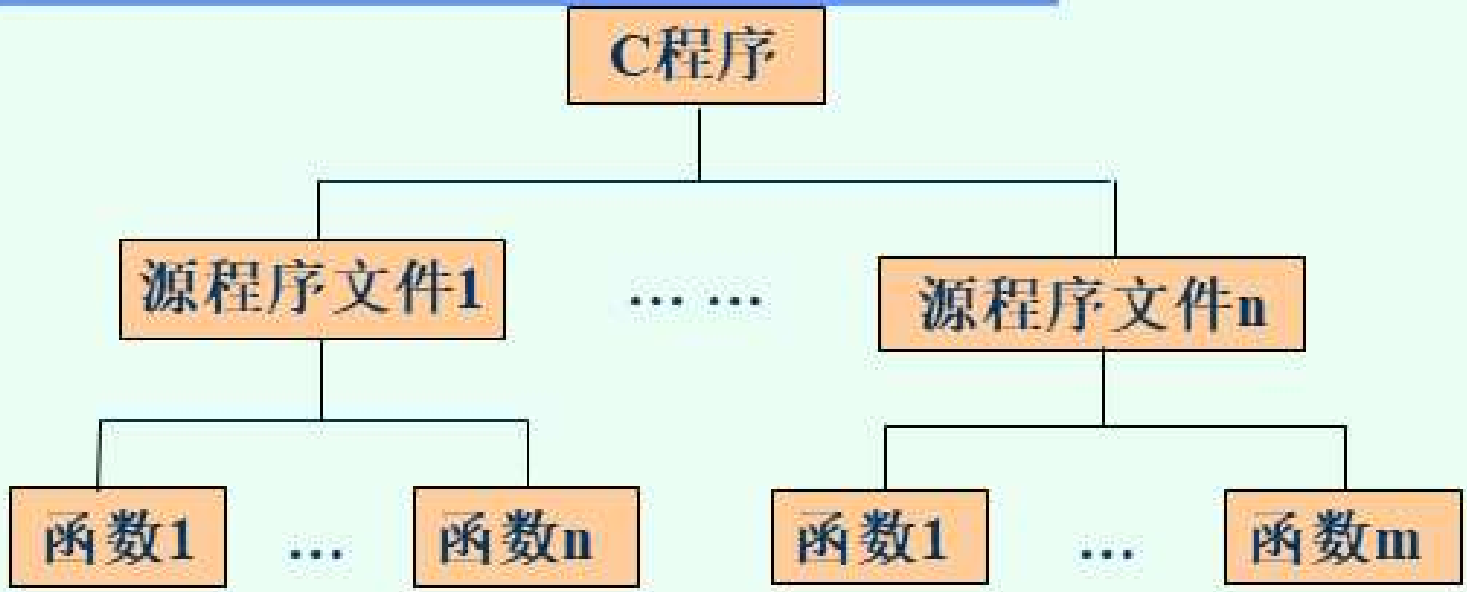
局部变量和全局变量及其作用域

变量的存储类别及变量的生存期

编译预处理[自学]

2020/4/2

6.1 概述



使用函数的好处

① 程序结构清晰，可读性好。

② 减少重复编码的工作量。

③ 可多人共同编制一个大程序，缩短程序设计周期，提高程序设计和调试的效率。

2020/4/2

C语言常规程序结构

声明

上课基本都是一个文件的例子，提倡上课时使用至少两个文件的项目结构，其中一个主文件，调用其它文件中的函数；一个是功能文件，包括各种函数。

2020/4/2

【例6.1】求一个整数的立方。

```
int cube (int x)      /* 函数定义*/
{ return (x * x * x);  }
void main()
{ int f, a;
  printf("\nEnter an integer number:  ");
  scanf("%d", &a);
  f = cube (a);
  printf("%d * %d * %d = %d\n", a, a, a, f);
}
```

主调函数

程序的执行总是从main函数开始

函数调用

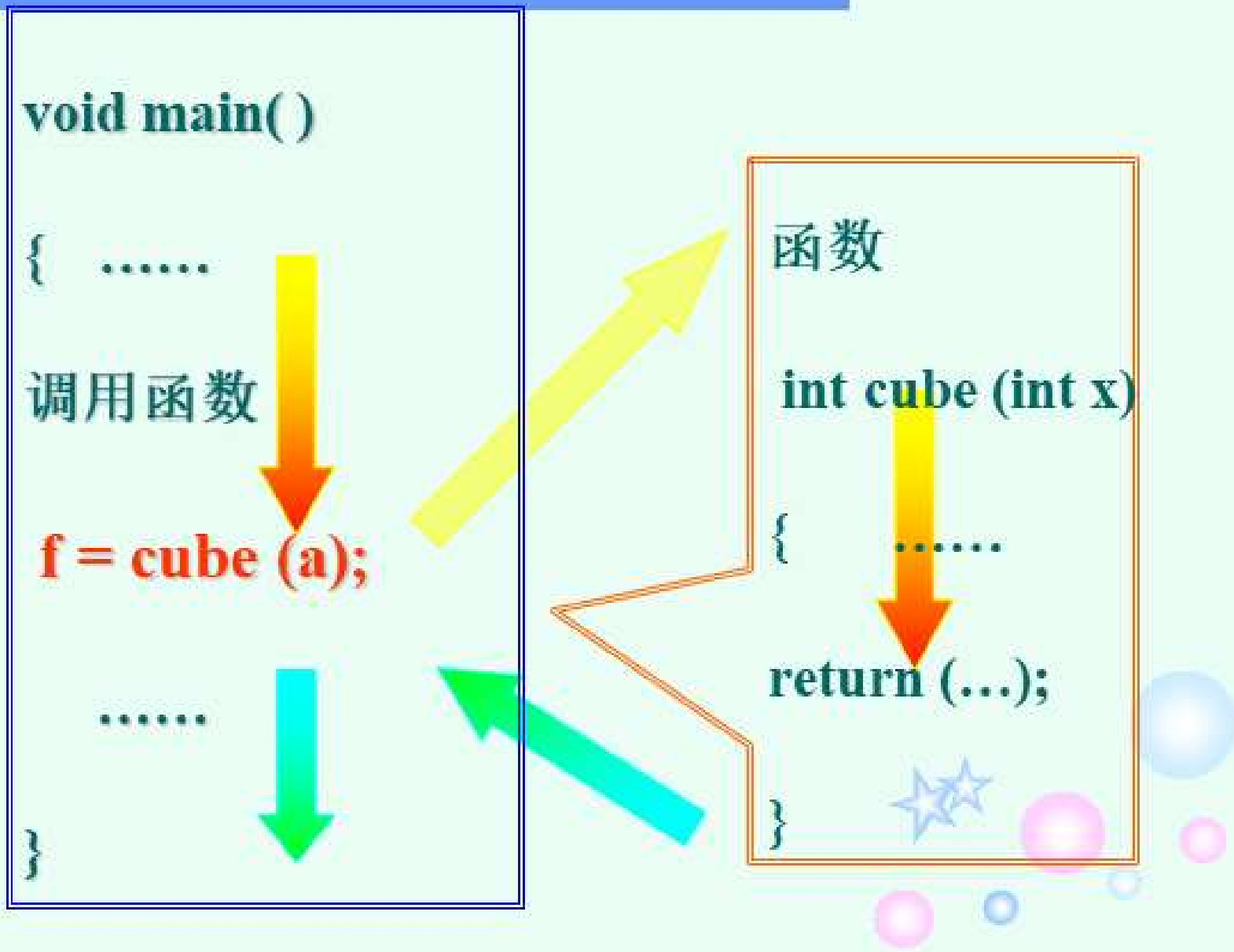
被调函数

程序运行情况如下：

```
Enter an integer number:  2↙
2 * 2 * 2 = 8
```

2020/4/2

【例6.1】求一个整数的立方。



说明

(1) 一个C源程序可以由一个或多个源程序文件组成。C编译系统在对C源程序进行编译时是以文件为单位进行的。

(2) 一个C源程序文件可以由一个或多个函数组成。所有函数都是独立的。主函数可以调用其它函数，其它函数可以相互调用。

(3) 在一个C程序中，有且仅有一个主函数main。C程序的执行总是从main函数开始，调用其它函数后最终回到main函数，在main函数中结束整个程序的运行。

2020/4/2

(4) 函数的种类

从使用的角度看：

① 标准库函数

库函数是由系统提供的。如：getchar( )、sqrt(x)等。在程序中可以直接调用它们。附录A

② 用户自定义函数

如:cube函数。

2020/4/2

(4) 数学库函数

- 由C编译系统定义的一类数学函数，存放在系统函数库中，用户可以根据需要随时调用
- 常用函数

  如：fabs、sqrt、sin、pow、rand

  (常用数学函数参见教材P100 表4-1和附录B)
- 函数调用形式：

  函数名（[参数表]）

  例如：　　sqrt(x)

(4) 函数的种类

double pow(double x, double y);

```
#include <math.h>
#include <stdio.h>
void main( )
{ double x=4.5;
   printf ("%f, %f, %f\n",sqrt(x), pow(x,2), fabs(-x));
}
```

输出结果：

2.121320, 20.250000, 4.500000

(4) 函数的种类

从函数定义形式分：

① 有参函数

在主调函数和被调函数之间通过参数进行数据传递， 如： **int cube (int x) { … }**

② 无参函数

在调用无参函数时，主调函数不需要将数据传递给无参函数。如：**getchar()**

2020/4/2

【例】无参函数的定义与调用。

```
void welcome ( )
{ printf("**************************************\n");
  printf("          Welcome  to  China \n");
  printf("**************************************\n");
}
void  main( )
{ welcome( );}
```

程序的输出结果如下：

```
**************************************
          Welcome  to  China
**************************************
```

2020/4/2

6.2.1 函数的定义

函数定义的一般形式

```
函数类型 函数名(类型名 形式参数1，...)
{ 说明语句
  执行语句
}
```

类型省略时默认为int类型，但不提倡

例如：求两个数的最大值。

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

若没有形式参数为无参函数

2020/4/2

形参也可以这样定义

不提倡

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

√

如下定义都是错误的

```
int max(x,y)
{ int x,y;
……
}
```

或

```
int max(int x,y)
{ …… }
```

×

或

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

×

花括号申也可以为空，这种函数叫空函数。
不能在函数体内定义其他函数，即函数不能嵌套定义。

2020/4/2

6.2.2 函数的调用

函数调用的一般形式：

函数名（实参表列）

在C语言中，函数调用也是一个表达式。因此凡是表达式可以出现的地方都可以出现函数调用。例如：

① welcome( );

② if (iabs (a)>max) max=iabs(a);

③ m=max(c,max(a,b));

2020/4/2

【例6.2】求1～100的累加和。

```
int sum100( )
{ int i, t=0;
  for (i=1; i<=100; i++)
    t+=i;
  return (t);}
```

```
void main( )
{ int s;
  s=sum100( );
  printf("%d\n", s);
}
```

```
void main( )
{ int i,sum;
  i=1; sum=0;
  for (i=1; i<=100; i++)
  { sum=sum+i;
  }
printf("sum=%d\n", sum);

}
```

缺点：只能用于1~100的求和，适用性差！

2020/4/2

【例6.2】求1～100的累加和。

```
int sum100( )
{ int i, t=0;
  for (i=1; i<=100; i++)
    t+=i;
  return (t);}
void main( )
{ int s;
  s=sum100( );
  printf("%d\n", s);
}
```

通用性更强

```
int sum ( int x )
{ int i,t=0;
  for (i=1; i<=x; i++)
    t+=i;
  return (t);
}
void main( )
{ int s;
  s=sum (100);
  printf("%d\n", s);
}
```

通用性强的函数$\sum_1^n x$

2020/4/2

【例6.3】求1～100的累加和。

```
int sum ( int x, int y )
{ int i,t=0;
  for (i=x; i<=y; i++)
    t+=i;
  return (t);
}
```

注意函数的通用性

```
int sum ( int x )
{ int i,t=0;
  for (i=1; i<=x; i++)
    t+=i;
  return (t);
}
void main( )
{ int s;
  s=sum (100);
printf("%d\n", s);
}
```

s=sum (1,100);

```
int sum ( int x, int y )
{ int i,t=0;
  if (x>y)  {t=x;x=y;y=t}
    t=0;
  for (i=x; i<=y; i++)
    t+=i;
  return (t);
}
```

s=sum (100,1);

s=sum (1,100);

通用性强的函数 $\sum_{n_1}^{n_2} x$

2020/4/2

6.2.3 函数参数与函数的返回值

1．函数的形式参数与实际参数

注意：程序调试完毕后，函数功能要独立单一。例如：完成交换功能的函数里面就不提倡出现显示功能。

```
void swap(int x, int y)
{ int z;
  z=x; x=y; y=z;
  printf("\nx=%d,y=%d",x ,y);
}
void main( )
{ int a=10,b=20;
  swap(a,b);
  printf("\na=%d,b=%d\n",a,b);
}
```

形式参数（形参）

实际参数（实参）

单向值传递

程序输出结果：

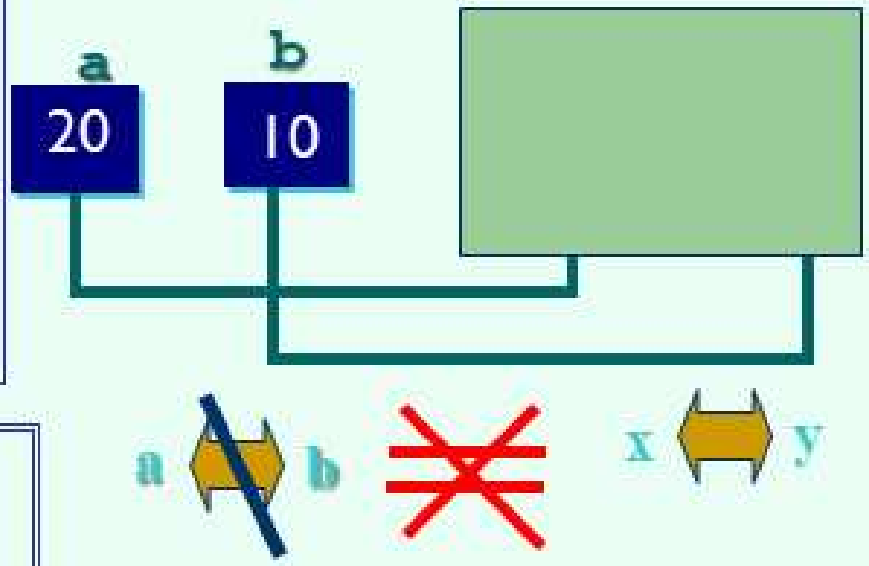
x=20,y=10

a=10,b=20

2020/4/2

有关形参和实参的说明：

```
void swap(int x, int y)
{ int z;
  z=x; x=y; y=z;
  printf("\nx=%d,y=%d",x ,y);
}
```




```
void  main( )
{ int a=20,b=10;
  swap(a,b);
  printf("\na=%d,b=%d\n",a,b);
}
```

形参不能改变实参的值

2020/4/2

有关形参和实参的说明：

① 当函数被调用时才给形参分配内存单元。调用结束，所占内存被释放。

② 实参可以是常量、变量、表达式或函数，但要求它们有确定的值。

③ 实参与形参类型要一致，字符型与整型可以兼容。

④ 实参与形参的个数必须相等。在函数调用时，实参的值赋给与之相对应的形参。“单向值传递”。

“类型一致、位置一致、个数一致”

2020/4/2

函数调用中实参的求值顺序。

```
void  fun(int a,int b)
{ printf("a=%d,b=%d\n",a,b); }
void main( )
{ int m=5;
  fun(3+m, m++);
}
```

程序输出结果：

VC下： a=9,b=5

2020/4/2

2.函数的类型与函数的返回值

(1) 函数的类型

【例】输出两个数中的大数。

```
int max(int x,int y)
{ int z;
  z=x>y?x:y;
  return (z);    /* 返回z的值 */
}
void main( )
{ int a,b,c;
  scanf("%d,%d",&a,&b);
  c=max(a,b);
 printf("max is %d\n",c);
}
```

说明：

①函数的类型决定了函数返回值的类型。

②若省略函数的类型，系统默认其为int型。但是不提倡整型返回值数据类型省略。

③无返回值的函数应将其类型定义为void（空）类型。

2020/4/2

(2) 函数的返回值

函数的返回值是通过return语句带回到主调函数的

return 语句格式：

return (表达式); 或 return 表达式;
或 return;

功能： 终止函数的运行，返回主调函数，若有返回值，将返回值带回主调函数。

说明：

① 若函数没有返回值，return语句可以省略。

② return语句中的表达式类型一般应和函数的类型一致，如果不一致，系统自动将表达式类型转换为函数类型。

2020/4/2

【例6.3】计算并输出圆的面积。

```
int s(int r)
{ return 3.14*r*r;}
void main( )
{ int r,area;
  scanf("%d",&r);
  printf("%d\n",s(r));
}
```

自动转换为int型

程序运行情况如下：

```
2↙
12
```

思考：

若要得到单精度实型的圆面积，程序应如何修改

```
float s(int r)
```

2020/4/2

6.2.4 对被调函数的声明和函数原型

变量要先定义后使用，函数也如此。即被调函数的定义要出现在主调函数的定义之前。如swap函数：

```
void swap(int x, int y)
{...}
void main( )
{ ...
  swap(a,b);
}
```

允许整型函数（且参数也是整型）的定义出现在主调函数之后。如max函数：

```
void main( )
{ ...
  c=max(a,b);
}
int max(int x,int y)
{...}
```

如果非整型函数在主调函数之后定义，则应在主调函数中或主调函数之前对被调函数进行声明。

2020/4/2

6.2.4 对被调函数的声明和函数原型

如果非整型函数在主调函数之后定义，则应在主调函数中或主调函数之前对被调函数进行声明。

函数声明的形式:

函数类型 函数名(参数类型1 参数名1,...);

函数类型 函数名(参数类型1,参数类型2,...);

第2种形式省略了参数名，也称函数的原型。

```
char max(char x, char y);
char min(char , char);
void main( )
{ …
  c=max(a,b);
  d=min(a,b);
}
```

```
void main( )
{ char max(char x, char y);
  char min(char, char);
    …
  c=max(a,b);
  d=min(a,b);
}
```

[例6.4]显示两个数的和/差/积或商(键盘输入)

主调函数中声明 不提倡

键盘输入：4.5 *3
屏幕显示：4.5*3=13.5

```
void main( )  void calc(float,float,char);
{ void calc(float x,float y,char opr);
  float a,b;  char opr;
  printf("\nInput  expression:");
  scanf("%f%c%f",&a,&opr,&b);
  calc(a,b,opr);
}
```

函数设计

- 函数名：calc
- 输入参数几个？是什么？
  3　两个操作数，float
  操作符，char
- 是否需要返回值？如果需要，如何设置？无

```
void calc(float x,float y, char opr)
{  switch(opr)
     { case  '+':printf("%5.2f%c%5.2f=%6.2f\n",x,opr,y,x+y);return;
       case  '-': printf("%5.2f%c%5.2f=%6.2f\n",x,opr,y,x-y);return;
       case  '*': printf("%5.2f%c%5.2f=%6.2f\n",x,opr,y,x*y);return;
       case  '/': printf("%5.2f%c%5.2f=%6.2f\n",x,opr,y,x/y);return;
       default : printf("Operator  err! \n"); }
}
```

【例6.4】计算并输出两个数的和、差、积、商。

或者 void calc(float,float,char);

```
void calc(float x,float y,char opr);
void main( )
{ float a,b;  char opr;
  printf("\nInput  expression:");
  scanf("%f%c%f",&a,&opr,&b);
  calc(a,b,opr);
}
```

主调函数前声明

提倡

```
void calc(float x,float y,char opr)
{ switch(opr)
  { case  '+':printf("%5.2f%c%5.2f=%6.2f\n",x,opr,y,x+y);return;
    case  '-':printf("%5.2f%c%5.2f=%6.2f\n",x,opr,y,x-y);return;
    case  '*':printf("%5.2f%c%5.2f=%6.2f\n",x,opr,y,x*y);return;
    case  '/':printf("%5.2f%c%5.2f=%6.2f\n",x,opr,y,x/y);return;
    default :printf("Operator  err! \n");  }
}
```

【例6.4】计算并输出两个数的和、差、积、商。

对被调函数的声明

主调函数中

主调函数前

```
void main( )
{ void calc(float x,float y,char  opr);
  float a,b;  char opr;
  printf("\nInput  expression:");
  scanf("%f%c%f",&a,&opr,&b);
  calc(a,b,opr);
}
void calc(float x,float y,char opr)
{ switch(opr)
  { case  '+':printf("%5.2f%c%5.2f=%6.2f\n",x,opr,y,x+y);return;
    case  '-':printf("%5.2f%c%5.2f=%6.2f\n",x,opr,y,x-y);return;
    case  '*':printf("%5.2f%c%5.2f=%6.2f\n",x,opr,y,x*y);return;
    case  '/':printf("%5.2f%c%5.2f=%6.2f\n",x,opr,y,x/y);return;
    default :printf("Operator  err! \n");  }
}
```

```
void calc(float x,float y,char opr);
main( )
{ float a,b;  char opr;
  printf("\nInput  expression:");
scanf("%f%c%f",&a,&opr,&b);
  calc(a,b,opr);
}
```

函数的定义：

```
函数类型 函数名(参数类型1 参数名1,...)
{
  ...
}
```

```
double  MyMax(double x, double y)
{ double z=x;
  if (x<y) z=y;
  return z;
}
```

函数的调用：函数名(参数列表);　形参不改变实参的值

```
MyMax(3.5, 4.0);
```

函数的声明：

```
函数类型 函数名(参数类型1 参数名1,...);
函数类型 函数名(参数类型1,...);
```

→ 函数的原型

2020/4/2

应用举例

【例6-5】哥德巴赫猜想之一是任何一个不小于6的大偶数都可以表示为两个素数之和。如：

6=3+3,8=3+5,10=3+7等等，试编程序验证。

分析：设n为大于等于6的任一偶数，将其分解为n1和n2两个数，使得n1+n2=n，分别判断n1和n2是否为素数，若都是，则为一组解。若n1不是素数就不必再检查n2是否为素数。先从n1=3开始，直到n1=n/2为止。

虽需要输入很多数据进行验证，
核心是判断一个数是否是素数

2020/4/2

【例6-5】哥德巴赫猜想之一是任何一个不小于6的大偶数都可以表示为两个素数之和。如：6=3+3，8=3+5，10=3+7等等，试编程序验证。

100 3+97 5+95 7+93
…… 49+51

分析：设n为大于等于6的任一偶数，将其分解为n1和n2两个数，使得n1+n2=n，分别判断n1和n2是否为素数，若都是，则为一组解（不必再找了）。若n1不是素数就不必再检查n2是否为素数。先从n1=3开始，直到n1=n/2为止。

虽需要输入很多数据进行验证，核心是判断一个数是否是素数

数据定义：

N：指定的大偶数

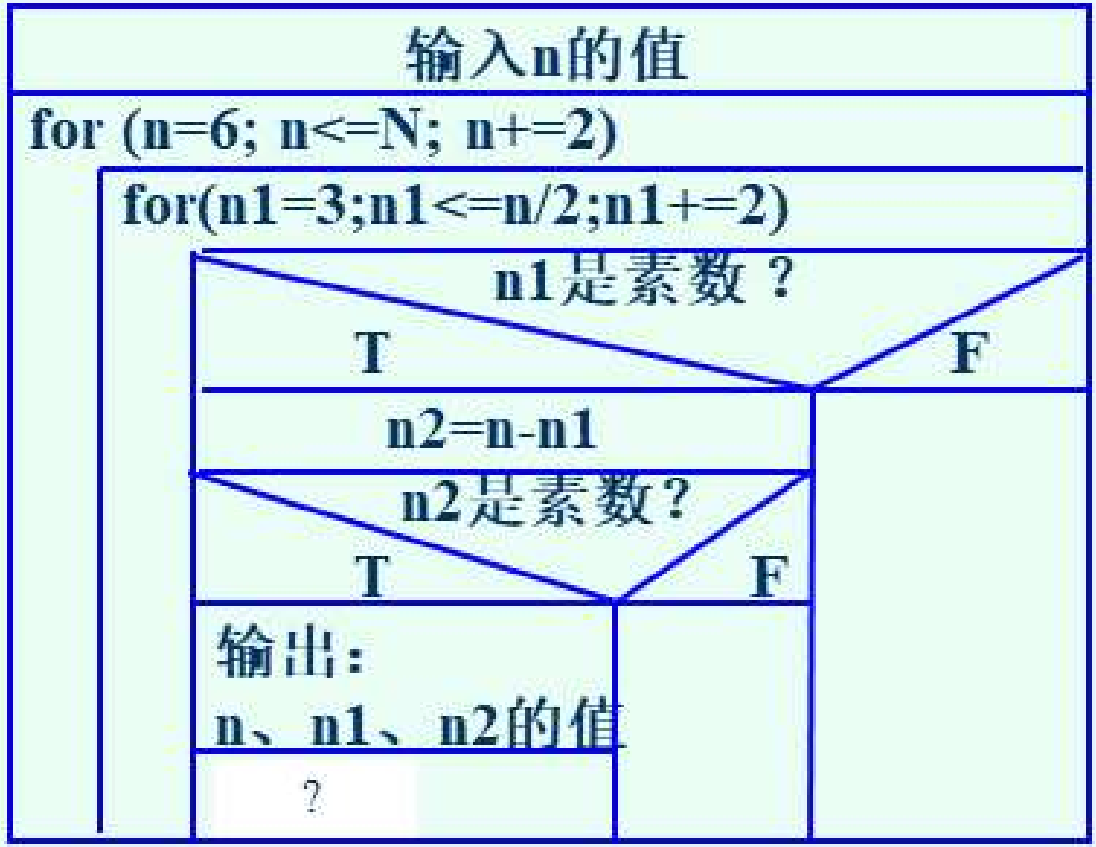
n：待判断的每一个偶数 $\in [6, N]$

n1,n2：分解后的数 n=n1+n2

算法描述

```
for n=6,8, …,N
    for n1=3,5,…, n/2
        if n1是素数
            n2=n-n1
            if n2是素数
                显示n=n1+n2
                退出当前循环
```

【例6.5】 哥德巴赫猜想之一是任何一个大于5的偶数都可以表示为两个素数之和。验证这一论断。



2020/4/2

先尝试一个·数·，例如6

素数判断

```
#include <math.h?
#include <stdio.h>
void main( )
{ int j,m=6,  k;
  k=sqrt(m);
  for (j=2; j<=k; j++)
     if (m%j==0)  break;
  if (j>=k+1)
     printf("%d,YES\n”,m);
  else
     printf(“%d, NO\n”,m);
}
```

```
/* 穷举判断素数 */
int isPrime(int n)
{  int i, k=sqrt(m);
    for (i=2; i<=k; i++)
          if (n%i==0) break;
    if (j>=k+1) return 1;
    else return 0;
}
```

2020/4/2

```
#include <math.h>
int isPrime(int  n);
void main( )
{ int n, n1, n2, N;
  scanf("%d", &N);
  for (n=6; n<=N; n+=2)
     {for (n1=3; n1<=n/2; n1+=2)
        if (isPrime(n1))
        {  n2=n-n1;
           if (isPrime(n2))
           {  printf("%d=%d+%d\n", n, n1, n2);
              break;
           }
        }
}
}
```

```
/* 穷举判断素数 */
int isPrime(int n)
{  int i, k=sqrt(m);;
    for (i=2; i<=k; i++)
          if (n%i==0) break;
    if (j>=k+1) return 1;
    else return 0;
}
```

```
for n=6,8, …,N
    for n1=3,5,…, n/2
        if n1是素数
            n2=n-n1
            if n2是素数
                显示n=n1+n2
                退出当前循环
```

先尝试一个·数·，例如6

素数判断

```
#include <math.h?
#include <stdio.h>
void main( )
{ int j,m=6, k;
  k=sqrt(m);
  for (j=2; j<=k; j++)
     if (m%j==0) break;
  if (j>=k+1)
    printf("%d,YES\n”,m);
 else
    printf(“%d, NO\n”,m);
}
```

函数设计

返回值 函数名（参数列表）

- 起名为。 isPrime
- 输入参数几个？是什么？

要判断是否为素数的数据n，
1个参数，整型数据

- 是否需要返回值？如果需要，如何设置？

需要一个标志位说明n是否为素数。规定1：Y 是0:N 否

int isPrime(int n)

第一步：函数设计

2020/4/2

改造

```
#include <math.h>
#include <stdio.h>
void main( )
{ int j,m=6, k;
  k=sqrt(m);
  for (j=2; j<=k; j++)
     if (m%j==0)  break;
  if (j>k)
    printf("%d,YES\n",m);
 else
    printf("%d,  NO\n",m);
}
```

int isPrime(int n)

```
/* 穷举法判断素数 */
int isPrime(int n)
{ int i, k; //控制循环
   k=sqrt(n);
    for (i=2; i<=k; i++)
       if (n%i==0)  break;

    if (i>k)   return 1;

    else       return 0;

}
```

第二步：素数判断函数的编写

2020/4/2

改造

```
int isPrime(int n)
{ int i,k; //控制循环
   k=sqrt(n);
  for (i=2; i<=k; i++)
      if (n%i==0)
         { return 0; }
   return 1;
}
```

更简洁

int isPrime(int n)

```
/* 穷举法判断素数 */
int isPrime(int n)
{ int i, k; //控制循环
   k=sqrt(n);
    for (i=2; i<=k; i++)
        if (n%i==0)  break;
    if (i>k)   return 1;
    else       return 0;

}
```

第二步：素数判断函数的编写

2020/4/2

素数判断测试

```
int isPrime(int n)
{ int i,k;
  k=sqrt(m);
  for (i=2; i<=k; i++)
     if (n%i==0) return 0;
  return 1;
}
```

```
if (isPrime(n))
   printf("%d: prime\n", n);
```

主调函数思路：

变量：n:待判断的数据（输入还是固定
　　s: 是否素数的标记

调用素数判断算法

显示：if 返回值是1 显示素数

```
#include <math.h>
#include <stdio.h>
int isPrime(int  n);
void main( )
{  int n=6, status;
   s=isPrime(n);
   if (s==1)
      printf("%d: prime\n", n);
}
```

第三步：素数判断函数的调试与测试

2020/4/2

单个大偶数n测试

n=n1+n2

变量：n:待判断的大偶数
n1,n2:因子 n1∈[3,n/2]
n2=n-n1
s1,s2: 因子素数标记

```
int isPrime(int n)
{ int i,k;
  k=sqrt(n);
  for (i=2; i<=k; i++)
     if (n%i==0) return 0;
  return 1;
}
```

思路：

for n1=3,5,7,9,……n/2
计算n1是素数否。
if Yes，
n2=n-n1
计算n2是否素数
if Yes
显示该偶数
并退出循环

第四步：单个大偶数分解测试：
例如n=100，看是否能分解成两个素数

2020/4/2

```
#include <math.h>
#include <stdio.h>
int isPrime(int  n);
void main( )
{  int n=100,n1,n2, s1,s2;
   for (n1=3; n1<=n/2; n1+=2)
    { s1=isPrime(n1);
      if (s1==1)
      {  n2=n-n1;
         s2=isPrime(n2);
         if (s2==1)
         { printf("%d=%d+%d\n",n,n1,\n2);
           break;
         }
      } //end if s1
   }//end of n1
}//end of main
```

n=n1+n2

变量：n:待判断的大偶数
n1,n2:因子 n1∈[3,n/2]
s1,s2: 因子素数标记

for n1=3,5,7,9,……n/2
计算n1是素数否。
if Yes，
n2=n-n1
计算n2是否素数
if Yes
显示该偶数
并退出循环

第四步：单个大偶数分解测试:例如n=100，看是否能分解成两个素数

```
#include <math.h>
#include <stdio.h>
int isPrime(int n);
void main( )
{ int n=100,n1,n2;
  for (n1=3; n1<=n/2; n1+=2)
    {if (isPrime(n1))
     { n2=n-n1;
       if (isPrime(n2))
              { printf("%d=%d+%d\n",n,n1,\n2); break; }
      }//end of if isPrime(n1)
     }//end of n1
}//end of main
```

```
s1=isPrime(n1);
if (s1= =1)
{  n2=n-n1;
   s2=isPrime(n2);
   if (s2= =1)
   { printf("%d=%d+%d\n",n,n1,\n2);
     break;
```

更简洁

第四步：单个大偶数分解测试:例如n=100，看是否能分解成两个素数

```
#include <math.h> #include <stdio.h>
int isPrime(int  n);
void main( )
{ int N,n,n1,n2;
  printf("Input an even num:\n");
  scanf("%d",&N);
  for (n=6; n<=N; n+=2)//多个
  {   for (n1=3; n1<=n/2; n1+=2)
       { if (isPrime(n1))
            { n2=n-n1;
              if (isPrime(n2))
             {printf("%d=%d+%d\n",n,n1,\n2);
               break;  }
       }
      }//end of for n1=3
 } //end of for n=6
}
```

单数判断

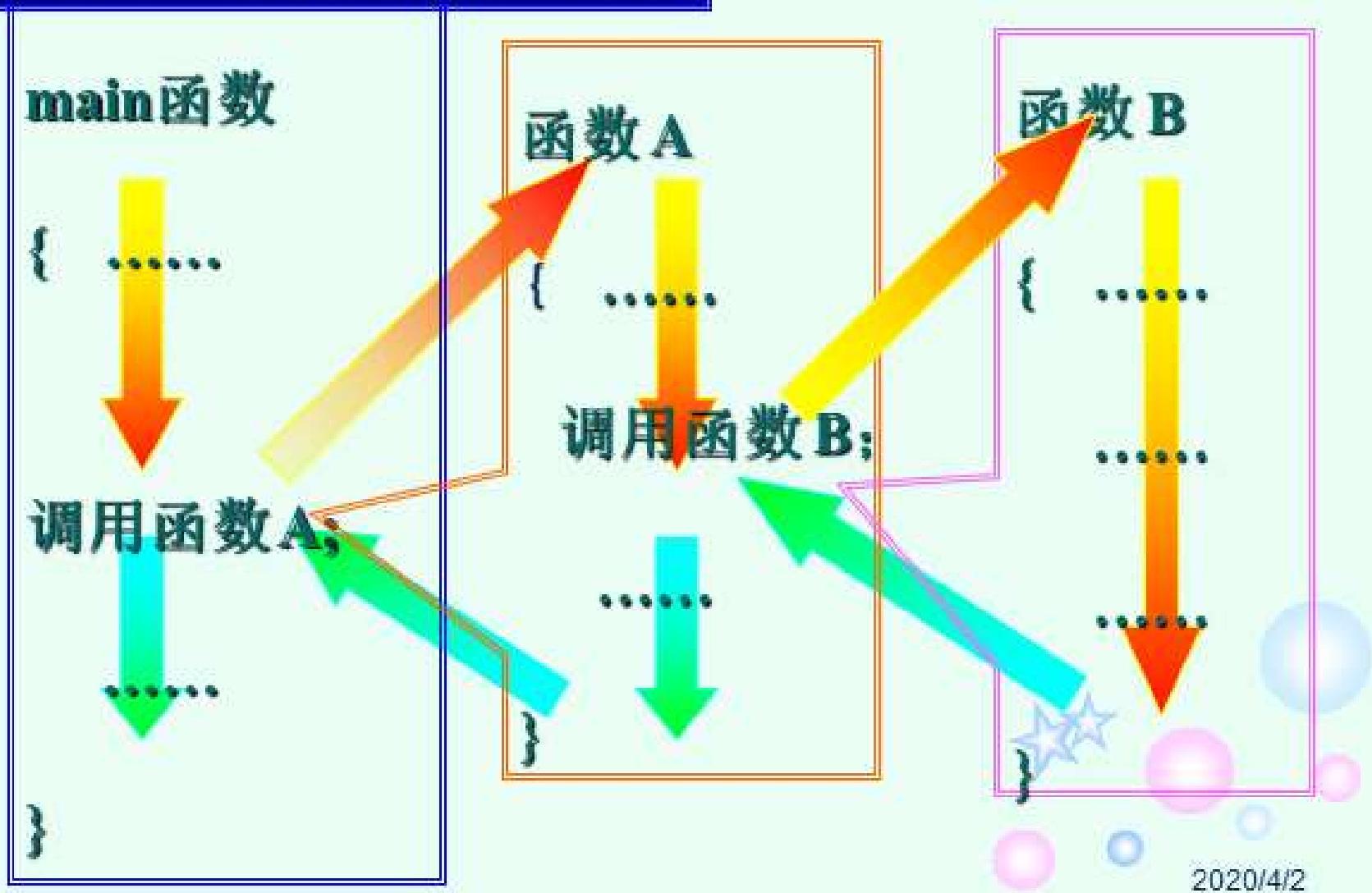
```
int isPrime(int  n);
void main( )
{ int n=100,n1,n2;
  for (n1=3; n1<=n/2; n1+=2)
    {
      if (isPrime(n1);)
         { n2=n-n1;
       if (isPrime(n2);  )
    { printf("%d=%d+%d\n",n,n
    1,\n2); break;  }
      }
     }//end of n1=3
}//end ofmain
```

单个大偶数测试

第五步：多个大偶数分解测试：6~指定范围内的所有偶数

6.4 函数的嵌套调用和递归调用

6.4.1 函数的嵌套调用



【例6.6】函数的嵌套调用

```
void main( )
{ int n=3;
  printf ("%d\n",sub1(n));
}
```

```
int sub1(int n)
{ int i,a=0;
  for (i=n; i>0; i--)
      a+=sub2(i);
  return a ;
}
```

```
int sub2(int n)
{
return   n+1;
}
```

程序输出结果：

9

2020/4/2

6.4.2 函数的递归调用

不要求!

递归调用：直接或间接地调用函数本身

2020/4/2

作业

- 上机作业

练习题：9905 9906 9908/9909（可二选一）
9910/9911（可二选一）

自测练习题：9913 9914 9916 9917(选作)

2020/4/2

6.5 局部变量和全局变量及其作用域

问题：一个变量在程序的所有函数中都能使用吗？

变量的作用域

变量的作用域：变量在程序中可以被使用的范围。

根据变量的作用域可以将变量分为局部变量和全局变量。

局部变量及其作用域

局部变量（内部变量）：在函数内或复合语句内定义的变量以及形参。

作用域：函数内或复合语句内。

【例6.7】分析下面程序的运行结果及变量的作用域。

2020/4/2

```
void sub(int a,int b)
{ int c;
  a=a+b;  b=b+a;  c=b-a;
printf("sub:\ta=%d  b= %d  c= %d\n",a,b,c); }
```

局部变量

```
main: a=1  b= 1  c= 1
sub:  a=2  b= 3  c= 1
main: a=1  b= 1  c= 1
comp: a=2  b= 2  c= 1
main: a=1  b= 1  c= 1
```

```
void main( )
{ int a=1,b=1,c=1;
 printf("main:\ta=%d  b= %d  c= %d\n",a,b,c);
  sub(a,b);
 printf("main:\ta=%d  b= %d  c= %d\n",a,b,c);
 { int a=2,b=2;
    printf("comp:\ta=%d  b= %d  c= %d\n",a,b,c); }
 printf("main:\ta=%d  b= %d  c= %d\n",a,b,c);  }
```

局部变量

局部变量

“分程序”或“程序块”

全局变量及其作用域

全局变量（外部变量）：在函数外部定义的变量。

作用域：从定义变量的位置开始到本源文件结束（文件作用域）。如在其作用域内的函数或分程序中定义了同名局部变量，则在局部变量的作用域内，同名全局变量暂时不起作用。

【例6.8】全局变量和局部变量的作用域。

2020/4/2

```
int a = 5;
void f(int x, int y)
{ int b,c;
  b=a+x;
  c=a-y;
printf("%d\t%d\t%d\n",a,b
    c);
}
```

全局变量

局部变量

```
void main( )
{ int b=6,c=7;
  f(b,c);
  printf("%d\t%d\t%d\n",a,b,c);
  { int a=9,b=8;
    printf("%d\t%d\t%d\n",a,b,c);
     { c=10;
     printf("%d\t%d\t%d\n", a,b,c)
      }
    printf("%d\t%d\t%d\n",a,b,c);
  }
  printf("%d\t%d\t%d\n",a,b,c);
}
```

局部变量

局部变量

Test1.c

程序输出结果:

5	11	-2
5	6	7
9	8	7
9	8	10
9	8	10
5	6	10

2020/4/2

全局变量

```
int a = 5;
void f(int x, int y)
{ int b,c;
  b=a+x;
  c=a-y;
printf("%d\t%d\t%d\n",a,b
   c);
}
```

局部变量

Test1.c

编译通不过。

a的作用范围在Test1.c里面，因此在Main.c中是变量没有定义。

Main.c

```
void main( )
{ int b=6,c=7;
  f(b,c);
  printf("%d\t%d\t%d\n",a,b,c);
  { int a=9,b=8;
    printf("%d\t%d\t%d\n",a,b,c)
     { c=10;
     printf("%d\t%d\t%d\n", a,b,c
      }
    printf("%d\t%d\t%d\n",a,b,c)
  }
  printf("%d\t%d\t%d\n",a,b,c);
}
```

局部变量

局部变量

2020/4/2

Test1.c

```
#include “test1.h”
a = 5;
void f(int x, int y)
{ int b,c;
  b=a+x;
  c=a-y;
printf("%d,%d,%d\n",a,
    b,c);
}
```

不考

Test1.h

```
#ifndef _TEST1_H
#define _TEST1_H
#include <stdio.h>
#ifndef  int_a
#define int_a
int a;
#endif
void f(int a, int b);
#endif
```

Main.c

```
#include “test1.h”
void main( )
{ int b=6,c=7;
  f(b,c);
  printf("%d\t%d\t%d\n",a,b,c);
  { int a=9,b=8;
    printf("%d\t%d\t%d\n",a,b,c);
    { c=10;
      printf("%d\t%d\t%d\n", a,b,c)
    }
    printf("%d\t%d\t%d\n",a,b,c);
  }
  printf("%d\t%d\t%d\n",a,b,c);
```

注意：变量一般不定义在头文件里面,多个编译单元均可以对其进行修改，可维护性差。

6.6 变量的存储类别及变量的生存期

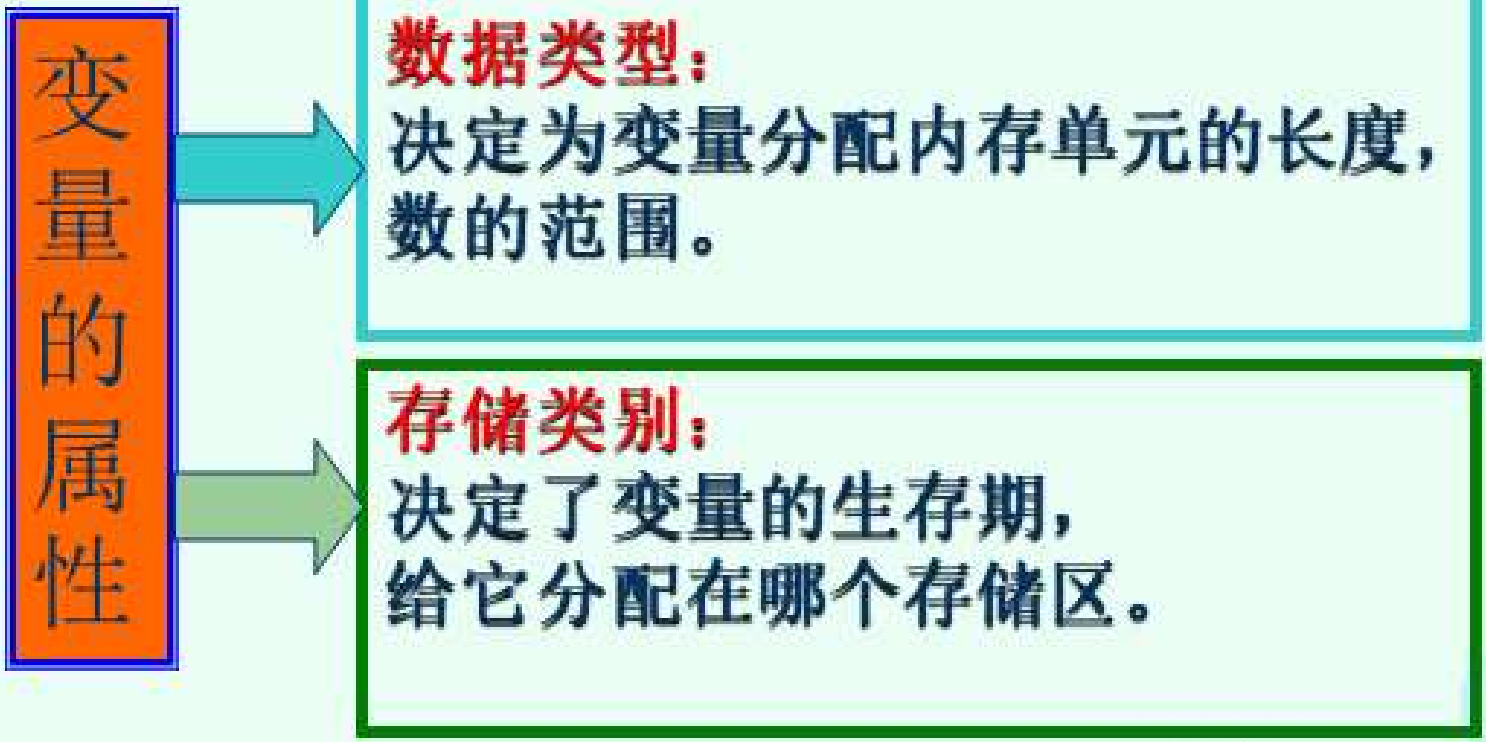
思考： 1. 何时为变量分配内存单元？
2. 将变量分配在内存的什么区域？
3. 变量占据内存的时间（生存期）？

变量的生存期与变量的存储分类

变量的生存期： 变量在内存申占据存储空间的时间。

2020/4/2

变量的存储类别



2020/4/2

变量定义语句的一般形式

存储类别 数据类型 变量名1, …, 变量名n;

auto（自动的）　register（寄存器的）
static（静态的）　extern（外部的）

存储类型只能选择一个

1. 自动变量（auto类别）

局部变量可以定义为自动变量。存储在动态存储区内。

```
void main()
{int x,y;
 …
}
```

等价

自动变量

```
void main()
{auto int x,y;
 …
}
```

可省

2020/4/2

自动变量

(1) 内存分配

调用函数或执行分程序时在**动态存储区**为其分配存储单元，函数或分程序执行结束，所占内存空间即刻释放。

(2) 变量的初值

定义变量时若没赋初值，变量的**初值不确定**；**如果赋初值则每次函数被调用时执行一次赋值操作**。

(3) 生存期

在函数或分程序执行期间。

(4) 作用域

自动变量所在的函数内或分程序内（**所在的花括号所定义的语句块**）。

2020/4/2

观察下列程序运行时变量的存储情况


```
void main()
{ int a,b,c;
  printf("Ente        b:\n");
  scanf("%d%d",&a,
  c=sum(a,b);
  printf("Sum=%d\n",c);
}
int sum(int  a,int b)
{ int  c=0;
  c=a+b;
  return(c);
}
```

2．静态变量（static类别）

除形参外，局部变量和全局变量都可以定义为静态变量。分配在静态存储区内。

静态变量
- 局部静态变量（或称内部静态变量）
- 全局静态变量（或称外部静态变量）

```
static int a;
void main( )
{ float x,y;
  … }
f( )
{ static int b=1;
  ……
}
```

- a：全局静态变量
- x,y：自动变量
- b：局部静态变量

不能省

2020/4/2

静态变量

(1) 内存分配

编译时，将其分配在内存的静态存储区中，程序运行结束释放该单元。

(2) 静态变量的初值

若定义时未赋初值，**在编译时**，系统**自动**赋初值为0；若定义时赋初值，则**仅在编译时赋初值一次**，程序运行后不再给变量赋初值。

(3) 生存期

整个程序的执行期间。

(4) 作用域

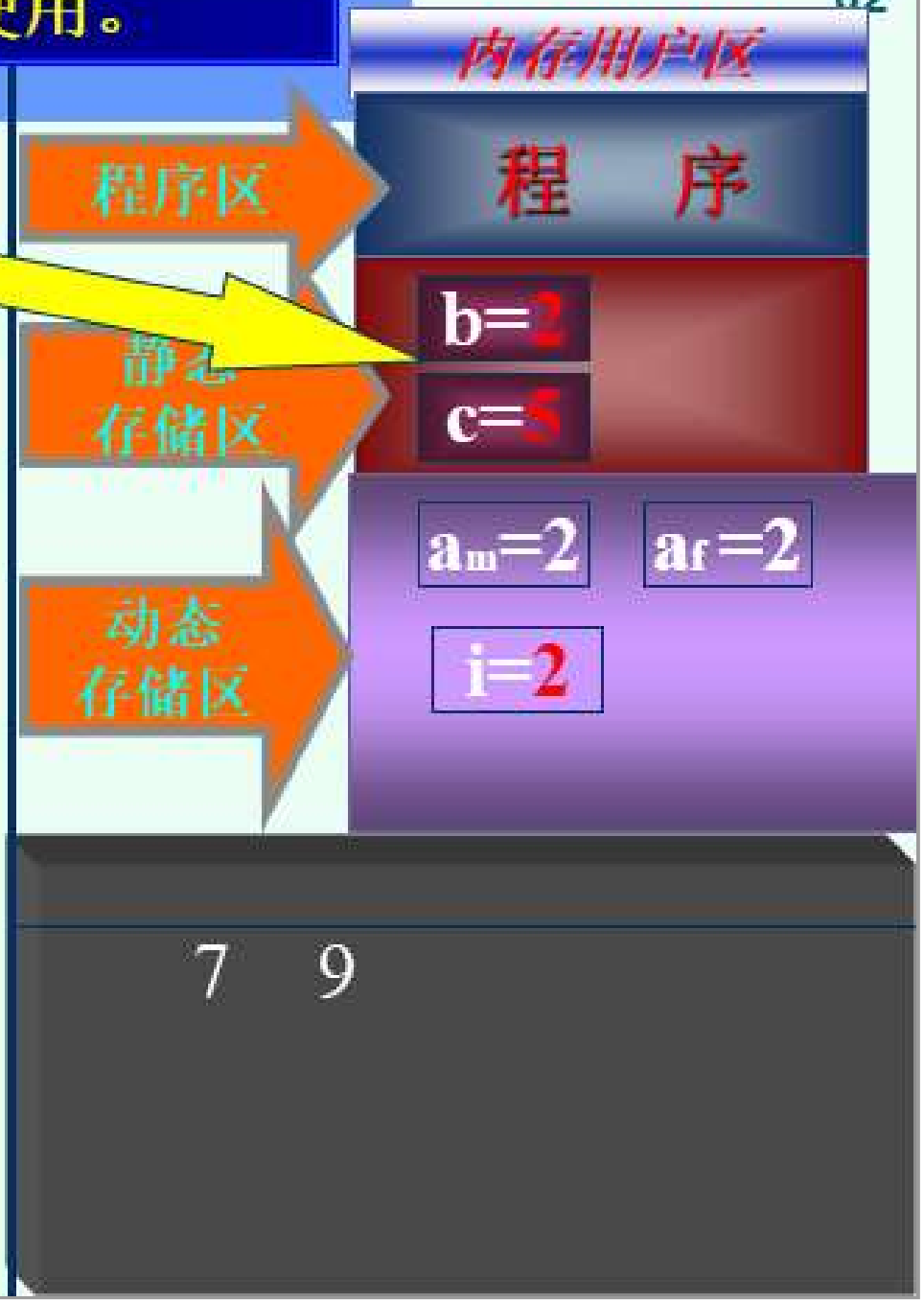
局部静态变量的作用域是它所在的函数或分程序。

全局静态变量的作用域是从定义处开始到***本源文件***(所在的c文件)结束。

2020/4/2

【例6.9】静态变量的使用。

```
static int b=0;
void main( )
{ int     a=2,i;
  for (i=0; i<2; i++)
     printf ("%4d",f(a));
  printf("\n");
}
int f(int a)
{ static int c=3;
  b++;  c++;
  return (a+b+c);
}
```



静态变量例子


```
void main( )
{  int i;
   for(i=1;i<4;i++)
   {  printf("No. %d call:", i);
      compare( );
   }
}
```

```
void compare( )
{  int a=0;
   static int b=0;
   a = a+1;
   b = b+1;
   printf("a=%d,b=%d\n", a, b);
}
```

```
No. 1 call: a=1,b=1
No. 2 call: a=1,b=2
No. 3 call: a=1,b=3
```

main里能否使用b?

2020/4/2

3.外部变量（extern类别）

在函数外定义的变量若没有用static说明，则是外部变量。外部变量只能隐式定义为extern类别，不能显式定义。

```
int c;
static int a;
void main( )
{ float x,y;
  … }
char s;
f( )
{ static int b=1;
  ……
}
```

外部变量

全局静态变量

自动变量

局部静态变量

2020/4/2

外部变量

(1) 内存分配

编译时，将其分配在静态存储区，程序运行结束释放该单元。

(2) 变量的初值

若定义变量时未赋初值，在编译时，系统自动赋初值为0。

(3) 生存期

整个程序的执行期间。

(4) 作用域

从定义处开始到本源文件结束。

此外，还可以用extern进行声明，以使其作用域扩大到该程序的其它文件中。

问题：全局静态变量的作用域可以扩展到本程序的其它文件吗?

2020/4/2

外部变量声明的一般格式

extern 数据类型 变量名1，…，变量名n;

或

extern 变量名1，…，变量名n;

注意：

①外部变量**声明**用关键字extern，而外部变量的**定义不能用**extern，只能隐式定义。

②定义外部变量时，系统要给变量分配存储空间，而对外部变量声明时，系统不分配存储空间，只是让编译系统知道该变量是一个已经定义过的外部变量，与函数声明的作用类似。

2020/4/2

【例6.10】在一个文件内声明外部变量。

```
int p=1,q=5;
float f1(int a)
{ extern char c1,c2;
   ……
}
char c1,c2;
char f2(int x,int y)
{   ……
}
void main( )
{   ……
}
```

定义外部变量

外部变量声明

定义外部变量

思考：在f1函数中声明c1、c2的作用是什么？如何修改程序使所有函数都可以使用外部变量而又不需要声明？

2020/4/2

【例6.11】在多文件的程序中声明外部变量。

file1.c文件中程序如下：

```
int i;
void main(
{ void f1( ),f2( ),f3( );
  i=1;
  f1( );
  printf("\tmain:  i=%d",i);
  f2( );
  printf("\tmain:  i=%d",i);
  f3( );
  printf("\tmain:  i=%d\n",i);
}
void f1( )
{ i++;
  printf("\nf1:  i=%d",i);
}
```

定义外部变量

file2.c文件中程序如下：

```
extern int i;
void f2( )
{ i=i+2;
  printf("\nf2:  i=%d",i);
}
void f3( )
{ i=3;
  printf("\nf3:  i=%d",i);
}
```

声明外部变量

程序输出结果：

```
f1: i=2 main: i=2
f2: i=4 main: i=4
f3: i=3 main: i=3
```

2020/4/2

【例6.12】在多文件的程序中声明外部变量。

file1.c文件中程序如下：

```
int i;
void main( )
{ void f1( ),f2( ),f3( );
  i=1;
  f1( );
  printf("\tmain:  i=%d",i);
  f2( );
  printf("\tmain:  i=%d",i);
  f3( );
  printf("\tmain:  i=%d\n",i);
}
void f1( )
{ i++;
  printf("\nf1:  i=%d",i);
}
```

file2.c文件中程序如下：

```
extern int i;
void f2( )
{ int i=4;
  printf("\nf2:  i=%d",i);
}
void f3( )
{ i=3;
  printf("\nf3:  i=%d",i);
}
```

程序输出结果：

```
f1: i=2 main: i=2
f2: i=4 main: i=2
f3: i=3 main: i=3
```

2020/4/2

4.寄存器变量（register类别）

寄存器变量的值保存在CPU的寄存器中。读写速度快。只有**函数内定义的变量或形参**可以定义为寄存器变量。受寄存器长度的限制，**寄存器变量只能是char、int和指针类型的变量。**

【例6.13】寄存器变量的使用。

```
void main( )
{ long int sum=0;
  register int i;
  for (i=1; i<=1000; i++)
    sum+=i;
  printf("sum=%ld\n ",sum);
}
```

程序输出结果：
sum=500500

2020/4/2

6.6.3 归纳变量的分类

1．按照变量的作用域对变量分类

(1) 局部变量

(2) 全局变量

2．按照变量的生存期对变量分类

(1) 静态存储变量

包括：局部静态变量和全局静态变量、外部变量

(2) 动态存储变量

包括：自动变量

2020/4/2

存储类别	标识符	定义形式(显式/隐式)	存储区	未初始化时的值	作用域	生存期
自动变量	auto	可显可隐	动态~	不定	所在分程序或函数内	函数或分程序的执行期
静态变量	static	显式	静态~	0	局部~：分程序或函数内；全局~：定义处至文件结束	程序执行期
外部变量	extern	隐式	静态~	0	定义处至文件结束	程序执行期
寄存器变量	register	显式	寄存器	不定	同自动变量	同自动变量

2020/4/2

6.1 概述

6.2 函数的定义

6.3 函数的调用

6.4 函数的嵌套调用和递归调用

6.5 局部变量和全局变量及其作用域

6.6 变量的存储类别及变量的生存期

6.7 编译预处理

2020/4/2

编译预处理

预处理命令	格式	功能
#include	#include “头文件名" #include <头文件名>	把指定的文件嵌入到该命令行位置取代该命令行，从而把指定的文件和当前的源程序文件连成一个源文件。
#define	#define 宏名 字符串 #define 宏名(形参表) 字符串	定义宏名来表示一个字符串，编译之前将程序中的宏名替换为字符串，再进行编译。
#undef	#undef宏名	撤销之前定义的宏名
#ifdef	#ifdef标识符 语句 #endif	条件编译，如果已定义了“标识符”，则编译“语句”
#ifndef	#ifndef标识符 语句 #endif	条件编译，如果未定义“标识符”，则编译“语句”

编译预处理命令

- 编译系统对源程序编译之前进行处理
- 掌握两个：
  - 文件包含 include
  - 宏定义 define
    - #define 宏名 字符串
    - #define 宏名(形参表) 字符串
- 注①：是命令，不是语句，无分号
- 注②：带参数的宏定义

2020/4/2

[6.14]宏替换实例

```
#define   MAX(x,y)   x>y?x:y
void main()
{      int a=5, b=2, c=3, d=3, t;
       t = MAX(a+b,c+d) * 10;
       printf("%d\n",t);
}
```

a+b>c+d?a+b:c+d*10

思考：t = 7

33

若int a=2, b=2, c=3, d=3？ t=?

2020/4/2

【6.15】宏替换实例

```
#define   SQA(x,y)   x*x+y*y
void main()
{      int a=5, b=2, c=3, d=3, t;
       t = SQA(a+b,c+d);
       printf("%d\n",t);
}
```

a+b*a+b+c+d*c+d

思考：t = 32

#define SQA(x,y) (x)*(x)+(y)*(y)

(a+b)*(a+b)+(c+d)*(c+d)

t=? 85

2020/4/2

- **上机作业**

  练习题：9907

  自测练习题：9912 9915（选作） 9918 9919

- **课后习题：锐格课后作业**

2020/4/2

- 作业：
- 函数部分的自测练习题

其中第三题9914、第六题9917可以选作

- 数组部分预习：上机练习 10159 10161 10163
- 自测练习： 10166 10167 10169 10170 10171 10173 （非函数部分）
- 其它剩余为数组与函数相关练习。

2020/4/2

1 外部变量的作用域是从定义变量的位置开始到本源文件 的结束。生存期是项目结束

2 如果外部变量定义时未赋初值，则在编译时，系统自动赋初值为 0

3 C语言中 函数内定义的变量的默认存储类别为auto 类型。

4 外部变量、静态变量和全局变量存放在 静态 存储区中

5 外部变量不能显式定义，可以用关键字 extern 进行声明，将其作用域扩大到其他文件中。

2020/4/2

1 下面说法哪个是正确的？

A 在C语言中，函数不可以嵌套定义，但可以嵌套调用。B 当调用函数时，实参是一个数组名，则向函数传送的是数组的首地址。

C 一个函数利用return语句只能返回一个结果.

D 在程序运行过程中，系统分配给实参变量和形参变量的内存单元是不同的。

E 定义函数的首部fun(int *a,int b)则此函数的返回值类型是void。

F 形参与实参的参数传递只能是单向传递。

G 形参是指针/数组名时，函数可以改变实参的值。

H函数可以先声明后被主调函数调用，此时函数可以在主调函数后定义。例如 int myadd(int x, int y); 就是一种函数原型的声明方法

2020/4/2

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

void main()
{  int a=3,b=4,c=5,d;
   d=f(f(a,b),f(a,c));
   printf("%d\n",d);
}
```

f(a,b)→f(3,4) →-2

f(a,c)) →f(3,5) →-1

d=f(-2,-1); →d=3

3

2020/4/2

```
void fun()
{   static int a=1;
    a +=2;
    printf("%d",a);
}

void main()
{  int c;
   for(c=1;c<3;c++)
      fun();
}
```

a=1

c=1 a=3

c=2 a=5

c=3

35

2020/4/2