中国人民大学 信息学院
RENMIN UNIVERSITY OF CHINA SCHOOL OF INFORMATION

程序设计荣誉课程

9. 语法3——函数

授课教师：游伟 副教授、孙亚辉 副教授

授课时间：周二14:00 – 15:30，周四10:00 – 11:30（教学三楼3304）

上机时间：周二18:00 – 21:00（理工配楼二层机房）

课程主页：https://www.youwei.site/course/programming

引子1：实现自己的格式化输入/输出函数

【问题描述】格式化输入函数scanf和格式化输出函数printf是C语言运行时库提供的函数。通过前面课程的学习，我们已经大致了解了scanf和printf的实现机理。现在，让我们动手实现一个my_scanf和my_printf函数。

- 定义全局字符数组in_buf和out_buf，分别模拟输入缓冲区和输出缓冲区
- my_printf函数将各个参数值按给定的格式控制，写到输出缓冲区
- my_scanf函数按给定的格式控制，从输入缓冲区中读取内容至各个参数

```
1.  #include <stdio.h>
2.  #include <stdarg.h>
3.  #define N 10000

4.  int my_scanf(const char *fmt, ...);
5.  int my_printf(const char *fmt, ...);
6.  char in_buf[N], out_buf[N];
```

```
7.  int main()
8.  {
9.      int x, y;
10.     gets(in_buf);
11.     my_scanf("%d %d", &x, &y);
12.     my_printf("%d %d", x, y);
13.     puts(out_buf);
14. }
```

引子2：C语言土味七言绝句

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int people;

int go_far(int psychological_age,
           int *physiological_age) {
  static int year = 0;
  int experience, things = 0;

  psychological_age++; (*physiological_age)++;
  year++, things++;
  experience = (rand()%200 - 100);
  people += experience * year;
  printf("year(%p): %d\n", &year, year);
  printf("people(%p): %d\n", &people, people);
  printf("things(%p): %d\n", &things, things);
  printf("psychological_age: %p\n", psychological_age);
  printf("physiological_age: %p\n", physiological_age);
  if (year < 50) experience +=
    go_far(psychological_age, physiological_age);
  printf("\n");
  return experience;
}
```

```c
int main(int argc, char **argv) {
  int psychological_age = 18;
  int physiological_age = 18;
  int mother_expect;
  srand(time(NULL));
  mother_expect = go_far(psychological_age,
                         &physiological_age);
  printf("psychological_age(%p): %d\n",
         &psychological_age, psychological_age);
  printf("physiological_age(%p): %d\n",
         &physiological_age, physiological_age);
  printf("mother_expect(%p): %d\n",
         &mother_expect, mother_expect);
}
```

```
year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.......
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101
```

```
year(0x565ec00c): 1
people(0x565ec010): 61
things(0xffbeb5e4): 1
psychological_age: 0xffbeb600
physiological_age: 0xffbeb624

year(0x565ec00c): 2
people(0x565ec010): -51
things(0xffbeb5a4): 1
psychological_age: 0xffbeb5c0
physiological_age: 0xffbeb624
.......
year(0x565ec00c): 50
people(0x565ec010): 9059
things(0xffbea9a4): 1
psychological_age: 0xffbea9c0
physiological_age: 0xffbeb624

psychological_age(0xffbeb620): 18
physiological_age(0xffbeb624): 68
mother_expect(0xffbeb628): 460
```

儿行千里母担忧，五十载起伏飘摇。
纵然已物是人非，归来却仍是少年。

目录

为什么要用函数

```
按学号选择排序函数() { 排序代码 }
按学号折半查找函数() { 查找代码 }
输入学生信息函数() { 输出代码 }
输出学生信息函数() { 输出代码 }
```

```
主函数(){
  调用输入学生信息函数()
  调用按学号选择排序函数()
  调用按学号折半查找函数()
  调用输出学生信息函数()
}
```

```
int main()
{
 {
     ⋮    功能1内容
 }

 {
     ⋮    功能2内容
 }

 {
     ⋮    功能1内容
 }

 {
     ⋮    功能2内容
 }
}
```

VS.

```
功能1函数()
{
     ⋮    功能1内容
}

功能2函数()
{
     ⋮    功能2内容
}

int main()
{
     调用功能1
     调用功能2
     调用功能1
     调用功能2
}
```

- 使用函数可以使程序清晰、精炼、简单、灵活。
- 函数就是功能。每一个函数用来实现一个特定的功能。函数名应反映其代表的功能。
- 在设计较大程序时，往往把它分为若干个程序模块，每一个模块包括一个或多个函数，每个函数实现一个特定的功能。
- 一个C程序可由一个主函数和若干个其他函数构成。由主函数调用其他函数，其他函数也可以互相调用。

为什么要用函数

(1) 一个C程序由一个或多个程序模块组成，每一个程序模块作为一个源程序文件。较大的程序，可分别放在若干个源文件中。这样便于分别编写和编译，提高调试效率。一个源程序文件可以为多个C程序共用。

(2) 一个源程序文件由一个或多个函数以及其他有关内容（如指令、数据声明与定义等）组成。一个源程序文件是一个编译单位，在程序编译时是以源程序文件为单位进行编译的，而不是以函数为单位进行编译的。

(3) C程序的执行是从main函数开始的，如果在main函数中调用其他函数，在调用后流程返回到main函数，在main函数中结束整个程序的运行。

(4) 所有函数都是平行的，即在定义函数时是分别进行的，是互相独立的。一个函数并不从属于另一个函数，即函数不能嵌套定义。函数间可以互相调用，但不能调用main函数。main函数是被操作系统调用的。

为什么要用函数

(5) 从用户使用的角度看，函数有两种。

① 库函数，它是由系统提供的，用户不必自己定义，可直接使用它们。应该说明，不同的C语言编译系统提供的库函数的数量和功能会有一些不同，当然许多基本的函数是共同的。

② 用户自己定义的函数。它是用以解决用户专门需要的函数。

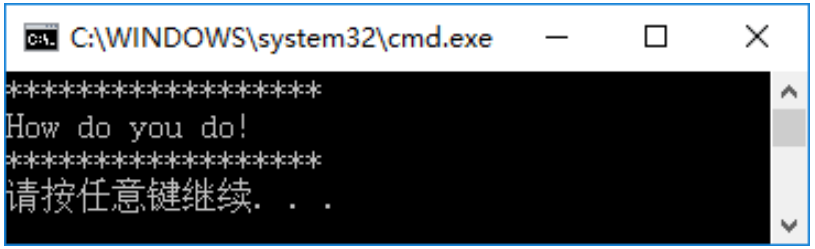
(6) 从函数的形式看，函数分两类。

① 无参函数。在调用无参函数时，主调函数不向被调用函数传递数据。

② 有参函数。在调用函数时，主调函数在调用被调用函数时，通过参数向被调用函数传递数据。

为什么要用函数

- 【例9-1】想输出下图所示的结果，用函数调用实现



print_star和print_message都是用户定义的函数名，分别用来输出一排“*”号和一行文字信息。在定义这两个函数时指定函数的类型为void，意为函数无类型，即无函数值，也就是说，执行这两个函数后不会把任何值带回main函数。

在程序中，**定义**print_star函数和print_message函数的位置是在main函数的后面，在这种情况下，应在main函数之前（或main函数开头），对以上两个函数进行**声明**，以便在**调用**时进行正确性检查。

```
1. #include <stdio.h>

2. void print_star();          //声明print_star函数
3. void print_message();//声明print_message函数

4. int main()  {
5.   print_star();             //调用print_star函数
6.   print_message();          //print_message函数
7.   print_star();             //调用print_star函数
8.   return 0;
9. }

10.void print_star()  {       //定义print_star函数
11. printf("**********\n"); //输出一行*号
12.}

13.void print_message()  { //定义print_message函数
14. printf("How do you do!\n"); //输出一行文字信息
15.}
```

9.1 函数定义/声明/调用

- 函数定义
  - C语言要求，在程序中用到的所有函数，必须"先定义，后使用"
  - 指定函数名字、函数返回值类型、函数参数名字与类型、函数功能代码，将这些信息通知编译系统
- 函数声明（函数原型）
  - 函数声明是为了编译器可以先对函数进行概要浏览
  - 把函数名字，函数返回值类型、函数参数的类型个数和顺序通知编译系统，以便在调用该函数时进行对照检查，函数声明不包括函数功能代码
  - 函数声明的作用是告诉编译器与该函数有关的信息，让编译器知道函数的存在及存在的形式，即使函数暂时没有定义，编译器也知道如何使用它
- 函数调用
  - 将函数调用作为一个语句或者表达式的一部分
  - 知识要点：参数和返回值的传递

9.1.1 函数定义

定义无参函数

```
类型名  函数名()
{
        函数体
}
```

或

```
类型名  函数名(void)
{
        函数体
}
```

函数名后面括号内的void表示“空”，即函数没有参数。

函数体包括**声明部分**和**语句部分**。

在定义函数时要用“类型标识符”(即类型名)指定函数值的类型，即指定函数带回来的值的类型。

定义有参函数

```
类型名  函数名(形式参数表列)
{
        函数体
}
```

```
int max(int x,int y)  {
        int z;              //声明部分
        z=x>y?x:y;          //执行语句部分
        return(z);
}
```

定义空函数

```
类型名  函数名()
{ }
```

函数体为空，什么也不做。

9.1.2 函数声明

- 在一个函数中调用另一个函数需具备如下条件
  - 被调用的函数必须是已经定义的函数（是库函数或用户自己定义的函数）。
  - 如果使用库函数，应该在本文件开头用#include指令将调用有关库函数时所需用到的信息“包含”到本文件中来。
  - 如果使用用户自己定义的函数，而该函数的位置在调用它的函数（即**主调函数**）的后面（在同一个文件中），则应该在主调函数中对被调用的函数（即**被调函数**）作声明(declaration)。声明的作用是把函数名、函数参数的个数和参数类型等信息通知编译系统，以便在遇到函数调用时，编译系统能正确识别函数并检查调用是否合法。

9.1.2 函数声明

函数类型 函数名(参数类型1 参数名1, 参数类型2 参数名2, …, 参数类型n 参数名n);

或 **函数类型 函数名(参数类型1, 参数类型2, …, 参数类型n);**

在函数声明中的形参名可以省写，而只写形参的类型。

```
float add(float x, float y);
float add(float, float);          //不写参数名，只写参数类型
float add(float a, float b);      //参数名不用x,y，而用a,b。合法
```

如果已在文件的开头(在所有函数之前)，已对本文件中所调用的函数进行了声明，则在各函数中不必对其所调用的函数再作声明。

```
//所有函数之前，且在函数外部进行函数声明
char letter(char, char); float f(float,float); int i (float,float);

//在main函数中要调用letter，f和i函数，不必再对所调用的这3个函数进行声明
int main() { … }

char letter(char c1,char c2) { … }                    //定义letter函数
float f(float x,float y) { … }                        //定义f函数
int i(float j,float k) { … }                          //定义i函数
```

注意

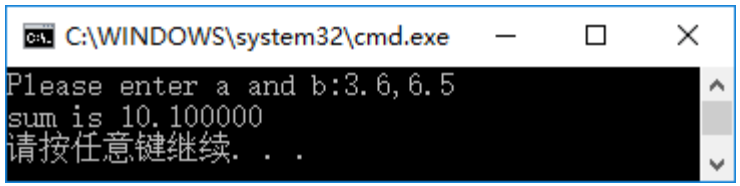
- 对函数的“定义”和“声明”不是同一回事。函数的定义是指对函数功能的确立，包括指定函数名、函数值类型、形参及其类型以及函数体等，它是一个完整的、独立的函数单位。而函数的声明的作用则是把函数的名字、函数类型以及形参的类型、个数和顺序通知编译系统，以便在调用该函数时系统按此进行对照检查，它不包含函数体。

9.1.2 函数声明

- 【例9-2】输入两个实数，用一个函数求出它们之和

```
1.  #include <stdio.h>

2.  float add(float x, float y); //对add函数作声明

3.  int main()  {
4.     float a,b,c;
5.     printf("Please enter a and b:"); //提示输入
6.     scanf("%f,%f",&a,&b);          //输入两个实数
7.     c=add(a,b);                    //调用add函数
8.     printf("sum is %f\n",c);       //输出两数之和
9.     return 0;
10. }
```

```
11. float add(float x,float y)//定义add函数
12. {
13.    float z;
14.    z=x+y;
15.    return(z);       //把变量z的值作为函数值返回
16. }
```



函数的声明和函数定义中的第1行（函数首部）基本上是相同的，只差一个分号(函数声明比函数定义中的首行多一个分号)。
函数的首行(即函数首部)称为**函数原型**(function prototype)。
因为在函数的首部包含了检查调用函数是否合法的基本信息(它包括了函数名、函数值类型、参数个数、参数类型和参数顺序)，因此在函数调用时检查函数原型是否与函数声明一致。这样就能保证函数的正确调用。

9.1.3 函数调用

函数名(实参表列)

```
print_star();      //调用无参函数
c=max(a,b);        //调用有参函数
```

1. 函数调用语句

把函数调用单独作为一个语句。如printf_star();

这时不要求函数带回值，只要求函数完成一定的操作。

2. 函数表达式

函数调用出现在另一个表达式中，如c=max(a,b);

这时要求函数带回一个确定的值以参加表达式的运算。

参数

- 在调用有参函数时，主调函数和被调用函数之间有**数据传递**关系
- 在定义函数时，函数名后面括号中的变量名称之为“**形式参数**”（简称“**形参**”）。
- 在主调函数中调用一个函数时，函数名后面括号中的参数称之为“**实际参数**”（简称“**实参**”）。 实参可以是常量、变量或表达式，但要求它们有确定的值。
- 实参与形参的**类型应相同或赋值兼容**。赋值兼容指实参与形参类型不同时能按不同类型数值的赋值规则进行转换。

返回值

通常，希望通过函数调用使主调函数能得到一个确定的值，这就是**函数值**(函数的返回值)。

(1) **函数的返回值是通过函数中的return语句获得的。**一个函数中可以有一个以上的return语句，执行到哪一个return语句，哪一个return语句就起作用。return语句后面的括号可以不要，如“return z;”与“return(z);”等价。return后面的值可以是一个表达式。

(2) **函数值的类型。**函数值的类型在定义函数时指定。

```
int max (float x，float y)  //函数值为整型
char letter (char c1,char c2)  //函数值为字符型
double min (int x,int y)  //函数值为双精度型
```

(3) **在定义函数时指定的函数类型一般应该和return语句中的表达式类型一致。**

如果函数值的类型和return语句中表达式的值不一致，则以函数类型为准。对数值型数据，可以自动进行类型转换。即**函数类型决定返回值的类型**。

(4) **对于不带回值的函数，应当用定义函数为“void类型”**（或称“空类型”）。这样，系统就保证不使函数带回任何值，即禁止在调用函数中使用被调用函数的返回值。此时在函数体中不得出现return语句。

函数调用和返回的过程

(1) 在定义函数中指定的**形参**，在未出现函数调用时，它们并不占内存中的存储单元。在发生函数调用时，函数的形参才被临时分配内存单元。

(2) 将**实参**的值传递给对应形参。

(3) 在执行函数期间，由于形参已经有值，就可以利用形参进行有关的运算。

(4) 通过return语句将**函数值**带回到主调函数。应当注意返回值的类型与函数类型一致。如果函数不需要返回值，则不需要return语句。这时函数的类型应定义为void类型。

(5) 调用结束，形参单元被释放。注意: 实参单元仍保留并维持原值，没有改变。如果在执行一个被调用函数时，形参的值发生改变，不会改变主调函数的实参的值。因为实参与形参是两个不同的存储单元。

注意

- 实参向形参的数据传递是“值传递”，单向传递，只能由实参传给形参，而不能由形参传给实参。实参和形参在内存中占有不同的存储单元,实参无法得到形参的值。

函数调用和返回的过程

- 【例9-3】输入两个整数，要求输出其中值较大者。要求用函数来找到大数

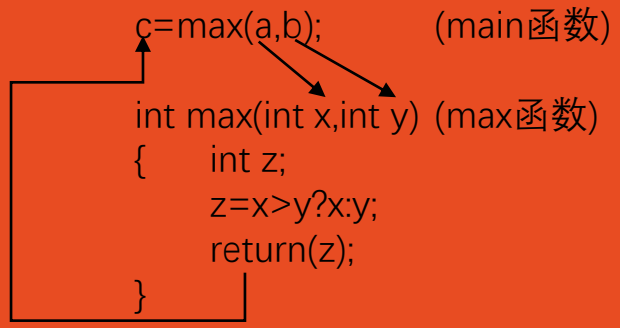
```
1. #include <stdio.h>

2. int max(int x,int y)    //定义max函数，有两个参数
3. {
4.   int z;          //定义临时变量z
5.   z=x>y?x:y;      //把x和y中大者赋给z
6.   return(z);      //把z作为max函数的值带回main函数
7. }
```

```
8. int main() {
9.   int a,b,c;
10. printf("please enter two integer numbers:");
11. scanf("%d,%d",&a,&b); //输入两个整数
12. c=max(a,b);          //调用max函数，返回值赋给变量c
13. printf("max is %d\n",c);  //输出大数c
14. return 0;
15.}
```

定义函数，名为max，函数类型为int。指定两个形参x和y，形参的类型为int。

主函数中包含了一个函数调用max(a,b)。max后面括号内的a和b是实参。a和b是在main函数中定义的变量，x和y是函数max的形参。

通过函数调用，在两个函数之间发生数据传递，实参a和b的值传递给形参x和y，在max函数中把x和y中的大者赋给变量z，z的值作为函数值返回main函数，赋给变量c。



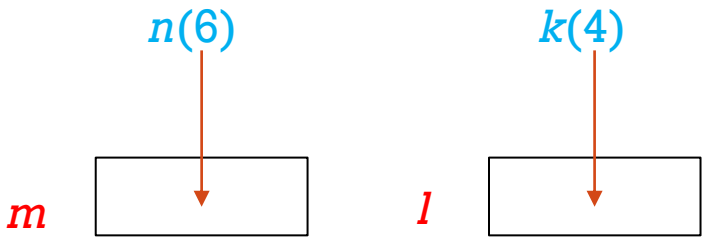
函数调用和返回的过程

- 【例9-4】求解$\sum_{x=1}^{n} x^k$，假定 n=6，k=4
- 思路：
  - 该式可分解为$1^4 + 2^4 + 3^4 + 4^4 + 5^4 + 6^4$
  - 定义函数 $power(p, q) = p^q$，让$q = 4$, $p = 1,2, \ldots, 6$，可表示1^4, 2^4,..., 6^4
  - 定义函数 $SOP(m, l) = \sum_{i=1}^{m} power(i, l)$，调用$SOP(\mathrm{n}, \mathrm{k})$即可得解

```
1. int power(int p,int q) {   //定义power函数
2.   int i, product = 1;
3.   for (i = 1; i <= q; i++)   product *= p;
4.   return product;
5. }

6. int SOP(int m, int l)  {   //定义SOP函数
7.   int i, sum = 0;
8.   for (i = 1; i <= m; i++) sum += Power(i, l);
9.   return sum;
10.}
```

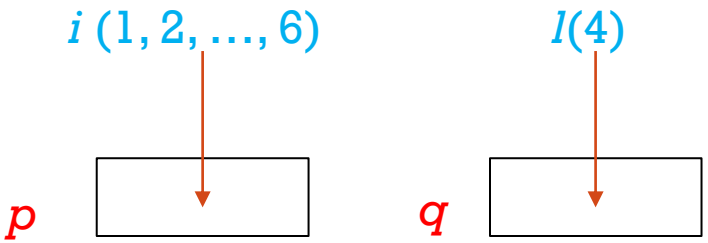
```
11.#include <stdio.h>
12.#define n 6
13.#define k 4

14.int main() {
15.   int result = SOP(n, k);
16.   printf("Sum of %d the powers of integers
17.           from 1 o %d is %d\n",
18.           k, n, SOP(n, k));
19.   return 0;
20.}
```

main函数调用SOP(*n*, *k*)，这时，*n*, *k*为实参，*n*的值为6，*k*的值为4。在SOP函数定义中，int SOP(*m*, *l*) 中的 *m*, *l* 为形参，在SOP被调用时，系统给 *m*, *l* 这两个形式参数分配内存单元。之后，*n* 的值 6 赋给 *m*；*k* 的值 4 赋给 *l*。

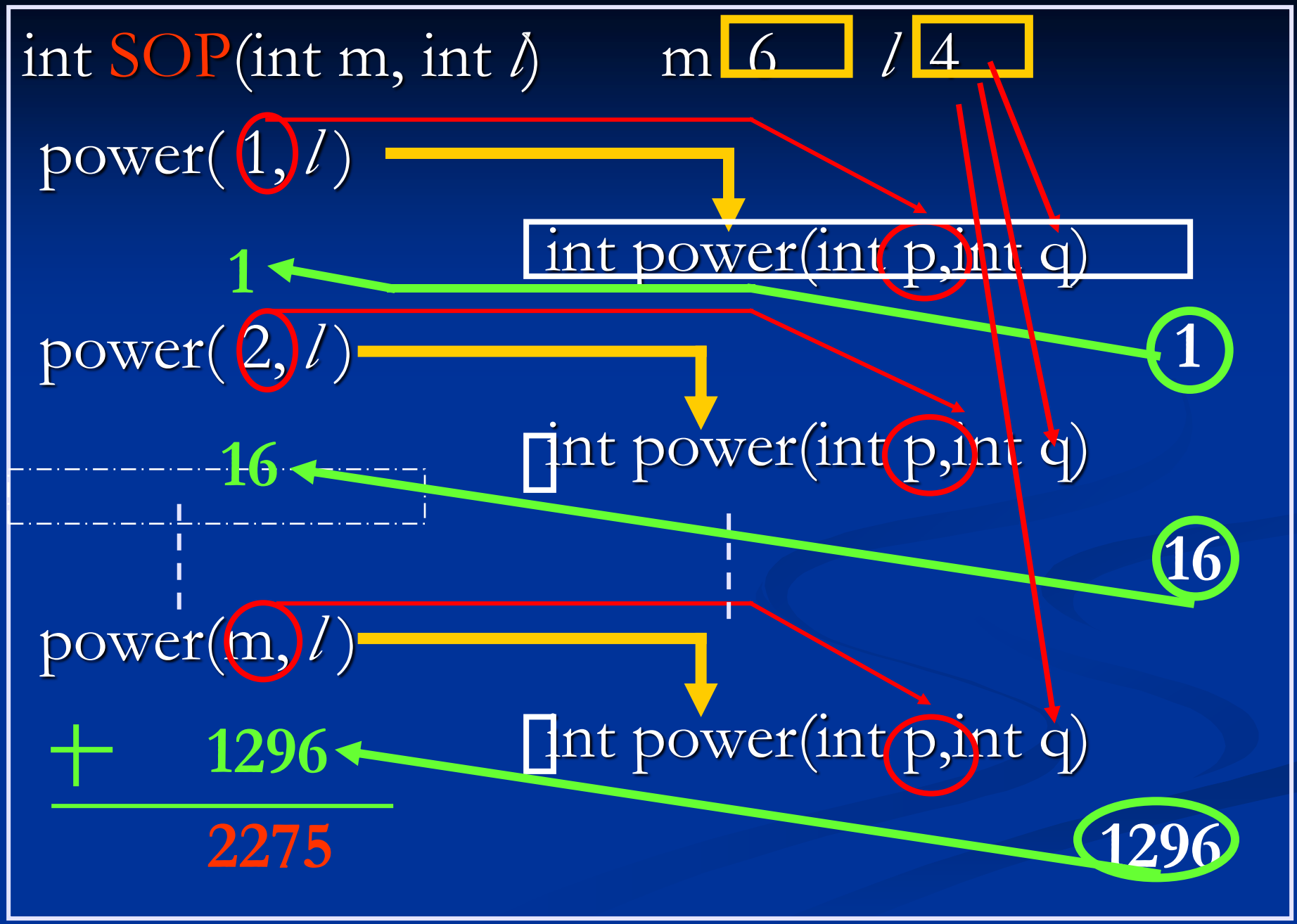


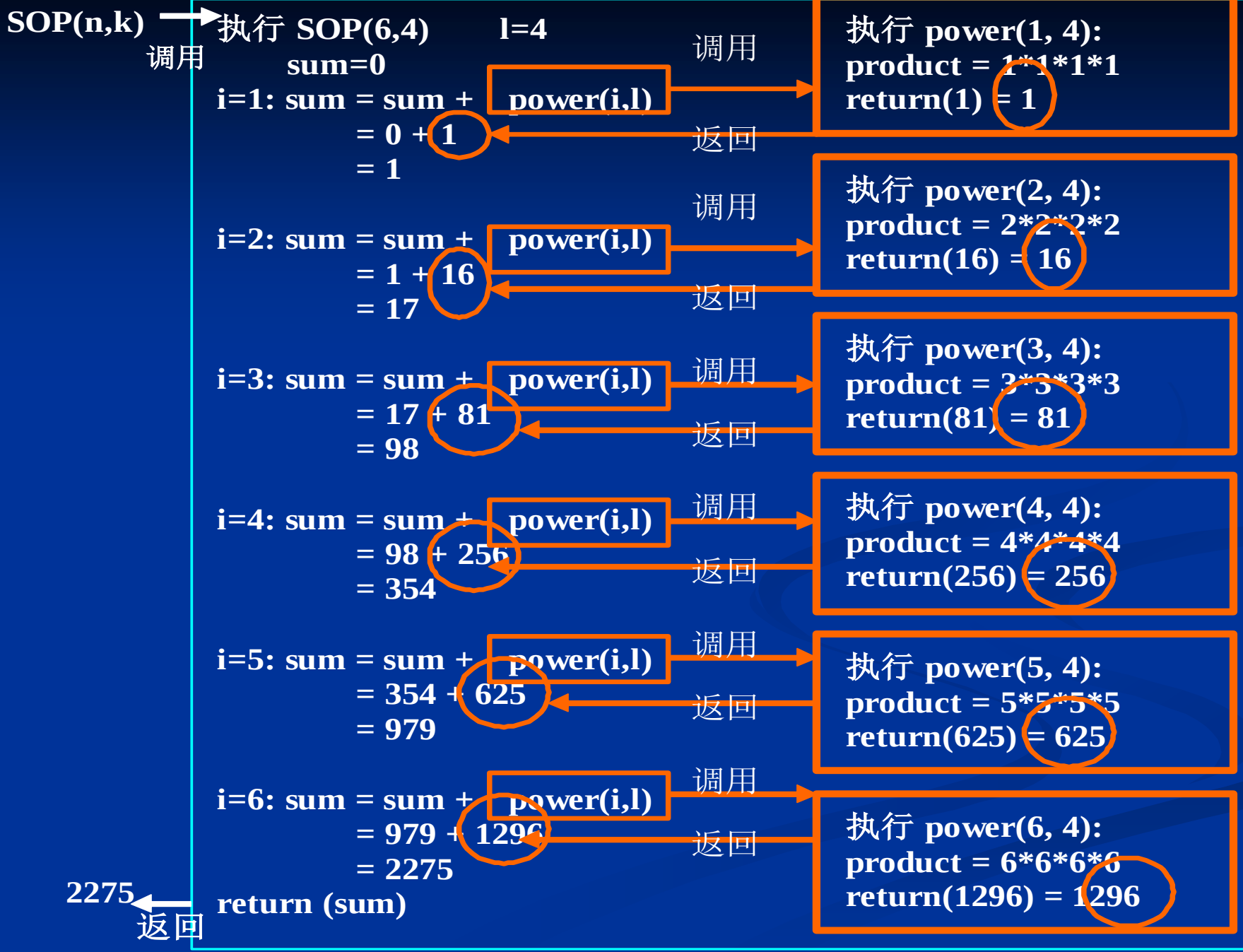
main calls **SOP**

SOP函数调用power(*i*, *l*)，这时，*i*, *l*为实参。在power函数定义中，int power(*p*, *q*) 中的 *p*, *q*为形参，在power被调用时，系统给 *p*, *q* 这两个形式参数分配了内存单元。之后，*i*的值(每轮循环中分别取值1, 2, …, 6) 赋给 *p*; *l*的值 4 赋给 *q*。



SOP calls **power**


int SOP(int m, int l)
m 6
l 4
power(1, l)
int power(int p,int q)
1
1
power(2, l)
int power(int p,int q)
16
16
power(m, l)
int power(int p,int q)
+ 1296
2275
1296


SOP(n,k)
调用
执行 SOP(6,4)
l=4
sum=0
i=1: sum = sum + power(i,l)
= 0 + 1
= 1
i=2: sum = sum + power(i,l)
= 1 + 16
= 17
i=3: sum = sum + power(i,l)
= 17 + 81
= 98
i=4: sum = sum + power(i,l)
= 98 + 256
= 354
i=5: sum = sum + power(i,l)
= 354 + 625
= 979
i=6: sum = sum + power(i,l)
= 979 + 1296
= 2275
return (sum)
2275
返回
调用
返回
执行 power(1, 4):
product = 1*1*1*1
return(1) = 1
执行 power(2, 4):
product = 2*2*2*2
return(16) = 16
执行 power(3, 4):
product = 3*3*3*3
return(81) = 81
执行 power(4, 4):
product = 4*4*4*4
return(256) = 256
执行 power(5, 4):
product = 5*5*5*5
return(625) = 625
执行 power(6, 4):
product = 6*6*6*6
return(1296) = 1296

嵌套调用

C语言的函数定义是互相平行、独立的，也就是说，在定义函数时，一个函数内不能再定义另一个函数，即不能嵌套定义，但可以嵌套调用函数，即在调用一个函数的过程中，又调用另一个函数。

① 执行main函数的开头部分；

② 遇函数调用语句，调用函数a，流程转去a函数；

③ 执行a函数的开头部分；

④ 遇函数调用语句，调用函数b，流程转去函数b；

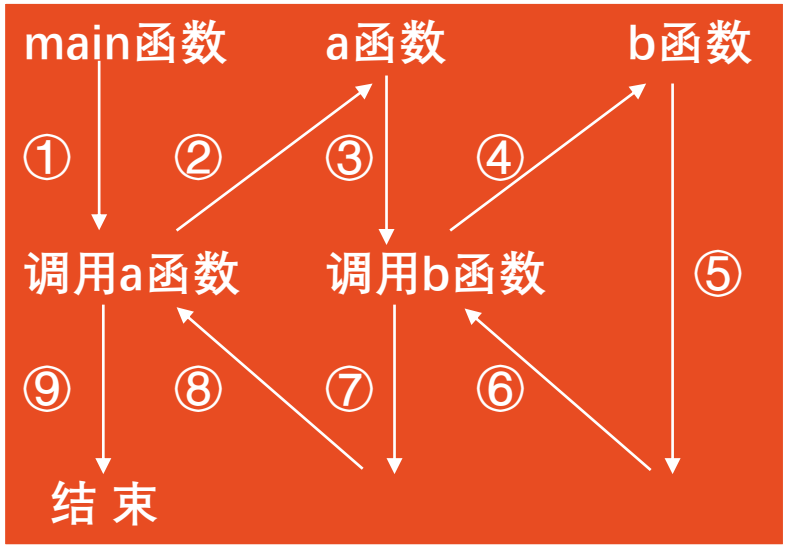
⑤ 执行b函数，如果再无其他嵌套的函数，
则完成b函数的全部操作；

⑥ 返回到a函数中调用b函数的位置；

⑦ 继续执行a函数中尚未执行的部分，直到a函数结束；

⑧ 返回main函数中调用a函数的位置；

⑨ 继续执行main函数的剩余部分直到结束。



嵌套调用

- 【例9-5】输入4个整数，找出其中最大的数。

```
1. #include <stdio.h>
2. int max4(int a,int b,int c,int d); //对max4的函数声明
3. int max2(int a,int b);             //对max2的函数声明

4. int main()
5. {
6.   int a,b,c,d,max;
7.   printf("Please enter 4 interger numbers:");
8.   scanf("%d %d %d %d",&a,&b,&c,&d);      //输入4个数
9.   max=max4(a,b,c,d); //调用max4函数，得到4个数中的最大者
10.  printf("max=%d \n",max);      //输出4个数中的最大者
11.  return 0;
12.}
```

```
13.int max4(int a,int b,int c,int d) { //定义max4函数
14.  int m;
15.  m=max2(a,b);//调用max2函数，得到a和b中的大者，放在m中
16.  m=max2(m,c);//调用max2函数，得到a,b,c中的大者，放在m中
17.  m=max2(m,d);//调用max2函数，得到a,b,c,d中的大者，放在m中
18.  return(m);     //把m作为函数值带回main函数
19.}

20.int max2(int a,int b) {   //定义max2函数
21.  if(a>=b) return a;       //若a≥b，将a作为函数返回值
22.  else return b;      //若a<b，将b作为函数返回值
23.}
```

- 在主函数中要调用max4函数，因此在主函数的定义之前要对max4函数作声明。在max4函数中3次调用max2函数，因此在max4函数的定义之前要对max2函数作声明。
- max4函数执行过程: 第1次调用max2函数得到的函数值是a和b中的大者，把它赋给变量m，第2次调用max2得到m和c中的大者，也就是a,b,c中的最大者，再把它赋给变量m。第3次调用max2得到m和d中的大者，也就是a,b,c,d中的最大者，再把它赋给变量m。这是一种**递推**方法，先求出2个数的大者；再以此为基础求出3个数的大者；再以此为基础求出4个数的大者。m的值一次一次地变化，直到实现最终要求。

嵌套调用

程序改进

(1) 可以将max2函数的函数体改为只用一个return语句，返回一个条件表达式的值:

```
int max2(int a,int b)  //定义max2函数
{return(a>=b?a:b);}  //返回条件表达式的值，即a和b中的大者
```

(2) 在max4函数中，3个调用max2的语句可以用以下一行代替:

```
m=max2(max2(max2(a,b),c),d); //把函数调用作为函数参数
```

甚至可以取消变量m，max4函数可写成

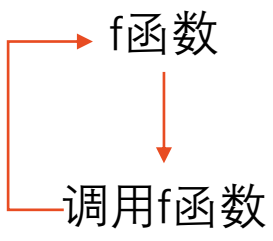
```
int max4(int a,int b,int c,int d)  {
    int max2(int a,int b);      //对max2的函数声明
    return max2(max2(max2(a,b),c),d);
}
```

- 先调用“max2(a,b)”，得到a和b中的大者。
- 再调用“max2(max2(a,b),c)”(其中max2(a,b)为已知)，得到a,b,c三者中的大者。
- 最后由“max2(max2(max2(a,b),c),d)”求得a,b,c,d四者中的大者。

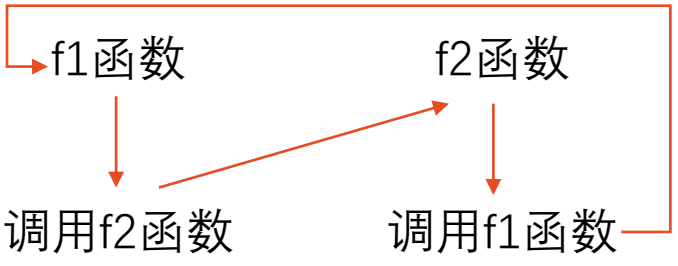
递归调用

在调用一个函数的过程中又出现**直接或间接地调用该函数本身，称为函数的递归调用。**

```
int f(int x)
{
    int y,z;
    z=f(y);     //在执行f函数过程中又调用f函数
    return (2*z);
}
```



直接递归

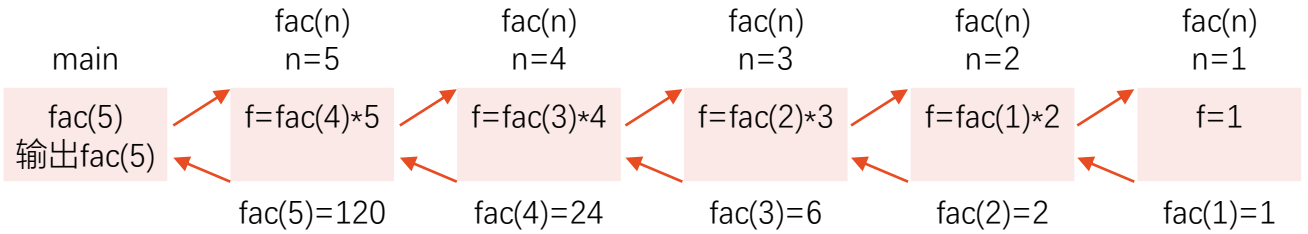


间接递归

程序中不应出现无终止的递归调用，而只应出现有限次数的、有终止的递归调用，这可以用if语句来控制，只有在某一条件成立时才继续执行递归调用；否则就不再继续。

注：递归调用需要设置一个终止条件，已使得递归能正常结束。

递归调用



- 【例9-6】用递归方法求n!

解题思路:

$$n!=\begin{cases} n!=1 & (n=0,1) \\ n\cdot(n-1)! & (n>1) \end{cases}$$

```
1. #include <stdio.h>
2. int fac(int n);          //fac函数声明

3. int main()
4. {
5.   int n;
6.   int y;
7.   printf("input an integer number:");
8.   scanf("%d",&n);   //输入要求阶乘的数
9.   y=fac(n);
10.  printf("%d!=%d\n",n,y);
11.  return 0;
12.}
```

```
13.int fac(int n)           //定义fac函数
14.{
15.  int f;
16.  if(n<0)                //n不能小于0
17.       printf("n<0,data error!");
18.  else if(n==0||n==1)    //n=0或,1时n!=1
19.       f=1;               //递归终止条件
20.  else
21.       f=fac(n-1)*n;      //n>1时, n!=n*(n-1)

22.  return(f);
23.}
```

注意

- 程序中的变量是int型，多数C编译系统为int型数据分配4个字节，能表示的最大数为2 147 483 647，当n=12时，运行正常，输出为479 001 600。如果输入13，企图求13!，则得不到预期结果，因为求出的结果超过了int型数据的最大值。可将f,y和fac函数定义为float或double型。

9.2 数组参数

形式参数	实在参数
变量	常量、变量、表达式、数组元素
数组名、指针变量	数组名

- 数组元素作为函数参数
  - 数组元素可以用作函数实参，但是不能用作形参。因为形参是在函数被调用时临时分配存储单元的，不可能为一个数组元素单独分配存储单元(数组是一个整体，在内存中占连续一段存储单元)
  - 在用数组元素作实参时，把实参的值传给形参，是“值传递”方式。数据传递的方向是从实参传到形参，单向传递
- 数组名作为函数参数
  - 数组名可以用作函数的形参和实参
  - 用数组名做实参时，向形参（数组名或指针变量）传递的是数组首元素的地址，是“地址传递”方式。在被调函数中操作形参指向的数组区域，直接影响主调函数中的数组实参

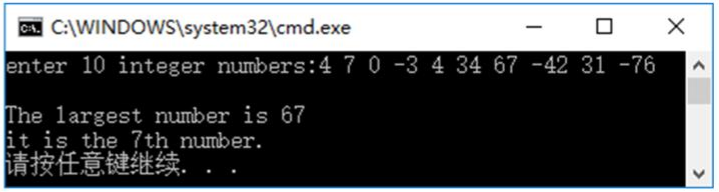
9.2.1 数组元素作函数参数

- 【例9-7】输入10个数，输出其中值最大的元素和该数是第几个数

```
#include <stdio.h>
int main()
{    int max(int x,int y);              //函数声明
     int a[10],m,n,i;
     printf("enter 10 integer numbers:");
     for(i=0;i<10;i++)                  //输入10个数给a[0]~a[9]
          scanf("%d",&a[i]);
     printf("\n");
     for(i=1,m=a[0],n=0;i<10;i++)
     {    if(max(m,a[i])>m)             //若max函数返回的值大于m
          {    m=max(m,a[i]);     //max函数返回的值取代m原值
               n=i;               //把此数组元素的序号记下来，放在n中
          }
     }
     printf("The largest number is %d\nit is the %dth number.\n",m,n+1);
}

int max(int x,int y)                    //定义max函数
{    return(x>y?x:y);                   //返回x和y中的大者
}
```

从键盘输入10个数给a[0]~a[9]。变量m用来存放当前已比较过的各数中的最大者。开始时设m的值为a[0]，然后依次将m与a[i]比，如果a[i]大于m，就以a[i]的值取代m的原值。下一次以m的新值与下一个a[i]比较。经过9轮循环的比较，m最后的值就是10个数的最大数。

请注意分析怎样得到最大数是10个数中第几个数。当每次出现以max(m,a[i])的值取代m的原值时，就把i的值保存在变量n中。n最后的值就是最大数的序号(注意序号从0开始)，如果要输出“最大数是10个数中第几个数”，应为n+1。因为数组元素序号从0开始。



注意

- 用**数组元素**作实参时，向形参变量传递的是数组元素的**值**，而用**数组名**作函数实参时，向形参(数组名或指针变量) 传递的是**数组首元素的地址**。

9.2.2 一维数组名作函数参数

- 【例9-8】有一个一维数组score，内放10个学生成绩，求平均成绩

```
#include <stdio.h>
int main()
{    float average(float array[10]);      //函数声明
     float score[10],aver;
     int i;
     printf("input 10 scores:\n");
     for(i=0;i<10;i++)
          scanf("%f",&score[i]);
     printf("\n");
     aver=average(score);                 //调用average函数
     printf("average score is %5.2f\n",aver);
     return 0;
}

float average(float array[10])       //定义average函数
{    int i;
     float aver,sum=array[0];
     for(i=1;i<10;i++)
          sum=sum+array[i];       //累加学生成绩
     aver=sum/10;
     return(aver);
}
```

```
C:\WINDOWS\system32\cmd.exe
input 10 scores:
100 56 78 98 67.5 99 54 88.5 76 58

average score is 77.50
请按任意键继续. . .
```

(1) 用数组名作函数参数，应该在主调函数和被调用函数分别定义数组。
(2) 实参数组与形参数组类型必须一致。
(3) 在定义average函数时，声明形参数组的大小为10，但在实际上，指定其大小是不起任何作用的，因为C语言编译系统并不检查形参数组大小，只是将实参数组的首元素的地址传给形参数组名。
(4) 形参数组可以不指定大小，在定义数组时在数组名后面跟一个空的方括号。 `float average(float array[])`

9.2.2 一维数组名作函数参数

- 【例9-9】有两个班级，分别有35和30名学生，调用average函数，分别求这两个班的学生的平均成绩

```
#include <stdio.h>
int main()
{    float average(float array[],int n);
     float score1[5]={98.5,97,91.5,60,55};   //定义长度为5的数组
     float score2[10]={67.5,89.5,99,69.5,77,89.5,76.5,54,60,99.5};
     //定义长度为10的数组
     printf("The average of class A is %6.2f\n",average(score1,5));
     //用数组名score1和5作实参
     printf("The average of class B is %6.2f\n",average(score2,10));
     //用数组名score2和10作实参
     return 0;
}

float average(float array[],int n) //定义average函数，未指定形参数组长度
{    int i;
     float aver,sum=array[0];
     for(i=1;i<n;i++)
          sum=sum+array[i];   //累加n个学生成绩
     aver=sum/n;
     return(aver);
}
```

```
C:\WINDOWS\system32\cmd.exe
The average of class A is  80.40
The average of class B is  78.20
请按任意键继续. . .
```

注意

- 用数组名作函数实参时,不是把数组元素的值传递给形参，而是把实参数组的首元素的地址传递给形参数组，这样两个数组就共占同一段内存单元。

	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]
起始地址1000	2	4	6	8	10	12	14	16	18	20
	b[0]	b[1]	b[2]	b[3]	b[4]	b[5]	b[6]	b[7]	b[8]	b[9]

9.2.3 多维数组名作函数参数

可以用多维数组名作为函数的实参和形参，在被调用函数中对形参数组定义时可以指定每一维的大小，也可以省略第一维的大小说明。

```
int array[3][10]; 或 int array[][10];        //二者等价
```

```
int array[][]; 或 int array[3][ ]; //必须指定列数
```

在定义二维数组时，必须指定列数(即一行中包含几个元素），由于形参数组与实参数组类型相同，所以它们是由具有相同长度的一维数组所组成的。不能只指定第1维(行数)而省略第2维(列数)。

在第2维大小相同的前提下，形参数组的第1维可以与实参数组不同。例如，实参数组定义为int score[5][10];而形参数组定义为int array[ ][10];或int array[8][10];均可以。这时形参数组和实参数组都是由相同类型和大小的一维数组组成的。C语言编译系统不检查第一维的大小。

9.2.1 数组参数

多维数组名作函数参数

可以用多维数组名作为函数的实参和形参，在被调用函数中对形参数组定义时可以指定每一维的大小，也可以省略第一维的大小说明。

```
int array[3][10]; 或 int array[][10];        //二者等价
```

```
int array[][]; 或 int array[3][ ];           //必须指定列数
```

在定义二维数组时，必须指定列数(即一行中包含几个元素)，由于形参数组与实参数组类型相同，所以它们是由具有相同长度的一维数组所组成的。不能只指定第1维(行数)而省略第2维(列数)。

在第2维大小相同的前提下，形参数组的第1维可以与实参数组不同。例如，实参数组定义为int score[5][10];而形参数组定义为int array[ ][10];或int array[8][10];均可以。这时形参数组和实参数组都是由相同类型和大小的一维数组组成的。C语言编译系统不检查第一维的大小。

9.3 可变参数

- 可变参数：函数的参数数量是可变的，不同的调用实例给定的实参个数不同，一个典型例子：printf(const char *fmt, ...)
- 在C语言中，变长参数通过三个宏（va_start、va_end、va_arg）和一个类型（va_list）实现的

void va_start (va_list ap, paramN);
参数：ap: 可变参数列表地址　　paramN: 确定的参数
功能：初始化可变参数列表（把函数在 paramN 之后的参数地址放到 ap 中）。

void va_end (va_list ap);
参数：ap: 可变参数列表地址
功能：关闭初始化列表（将 ap 置空）。

type va_arg (va_list ap, type);
参数：ap: 可变参数列表地址　　type：下一个参数的类型
功能：返回下一个参数的值。

va_list ： 存储参数的类型信息。

#include <stdarg.h>

```
// stdarg.h
#define va_start _crt_va_start
#define va_arg _crt_va_arg
#define va_end _crt_va_end
// vadefs.h
typedef char *  va_list;
#define _crt_va_start(ap,v)  ( ap = (va_list)_ADDRESSOF(v) + _INTSIZEOF(v) )
#define _crt_va_arg(ap,t)    ( *(t *)((ap += _INTSIZEOF(t)) - _INTSIZEOF(t)) )
#define _crt_va_end(ap)      ( ap = (va_list)0 )
#define _ADDRESSOF(v)   ( &(v) )
#define _INTSIZEOF(n)   ( (sizeof(n) + sizeof(int) - 1) & ~(sizeof(int) - 1) )
```

_INTSIZEOF：向上取整成sizeof(int)的整数倍

（详参：https://baijiahao.baidu.com/s?id=1617444332889746919）

9.3 可变参数

- 【例9-6】求多个数值的和

```c
#include <stdio.h>
#include <stdarg.h>

int va_sum (int first_num, ...);

int main ()
{
    int sum_val = va_sum(1, 2, 3, 4, 5, 0);
    printf("%d", sum_val);
    return 0;
}
```

以特殊值0作为变长参数的结束标志

```c
int va_sum(int first_num, ...) {
    int result = first_num, tmp;
    va_list ap;                     // (1) 定义参数列表
    va_start(ap, first_num);        // (2) 初始化参数列表

    while (1) {
        tmp = va_arg(ap, int);      // (3) 获取参数值
        if (tmp == 0) break;
        result += tmp;
    }

    va_end(ap);                     // (4) 关闭参数列表
    return result;
}
```

```c
#include <stdio.h>
#include <stdarg.h>

int va_sum (int cnt, ...);

int main ()
{
    int sum_val = va_sum(5, 1, 2, 3, 4, 5);
    printf("%d", sum_val);
    return 0;
}
```

第一个变长参数指定后续变长参数的数量

```c
int va_sum(int cnt, ...) {
    int result = 0, tmp, i;
    va_list ap;                     // (1) 定义参数列表
    va_start(ap, cnt);              // (2) 初始化参数列表

    for (i = 0; i < cnt; i++) {
        tmp = va_arg(ap, int);      // (3) 获取参数值
        result += tmp;
    }

    va_end(ap);                     // (4) 关闭参数列表
    return result;
}
```

深入解析printf

- printf示例：printf("int: %5d, char: %-8c, float: %5.3f\n", d, c, f)
- 如何确定可变参数的个数？（根据%的个数确定可变参数个数）
- 如何确定每个可变参数的类型？（根据格式字符确定可变参数类型）

```
1.   int is_alphabeta(char ch) { return (ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z'); }
2.   int my_printf(const char *fmt, ...) {
3.     int i; int d; char c; float f;
4.     va_list ap;                    // (1) 定义参数列表
5.     va_start(ap, fmt);             // (2) 初始化参数列表
6.     for (i = 0; i < strlen(fmt); i++) {
7.       if (fmt[i] == '%') {
8.         if (fmt[i+1] == '%') { printf("%"); i++; }        //%%代表输出一个%
9.         else {
10.          do { i++; } while (i < strlen && !is_alphabeta(fmt[i]));    //找到%后第一个字母
11.          switch (fmt[i]) {
12.            case 'd': d = va_arg(ap, int); break;       // (3) 获取参数值
13.            case 'c': c = va_arg(ap, char); break;      // (3) 获取参数值
14.            case 'f': f = va_arg(ap, float); break;     // (3) 获取参数值
15.            ……
16.          }
17.        }
18.      } else printf("%c", fmt[i]);
19.    }
20.    va_end(ap);                    // (4) 关闭参数列表
21.  }
```

函数综合应用举例

- 【例9-10】输入n个整数，找出其中最大的数（嵌套调用实现）

```
1. #include <stdio.h>
2. #define N 100
3. int max2(int a, int b);              //对max2的函数声明

4. int main()
5. {
6.    int n, i, max;
7.    int arr[N];
8.    scanf("%d", &n);
9.    for (i = 0; i < n; i++) scanf("%d", &arr[i]);      //输入n个数
10.   max = arr[0];   //初始设置max为arr数组第0个元素
11.   for (i = 1; i < n; i++) max = max2(max, arr[i]); //嵌套调用n-1次max2
12.   printf("final max: %d\n", max);     //输出n个数中的最大者
13.   return 0;
14.}

15.int max2(int a,int b) {  //定义max2函数
16. if(a>=b) return a;        //若a≥b，将a作为函数返回值
17. else return b;        //若a<b，将b作为函数返回值
18.}
```

输入：

```
6
3 5 1 7 9 6
```

输出：

函数综合应用举例

- 【例9-11】输入n个整数，找出其中最大的数（递归调用实现）

```
1. #include <stdio.h>
2. int max4(int a,int b,int c,int d); //对max4的函数声明
3. int max2(int a,int b);             //对max2的函数声明

4. int main()
5. {
6.    int n, i, max;
7.    int arr[N];
8.    scanf("%d", &n);
9.    for (i = 0; i < n; i++) scanf("%d", &arr[i]);
10.   max = maxm(arr, n);
11.   printf("final max: %d\n", max); //输出n个数中的最大者
12.}
```

```
13.int maxm(int arr[], int m)
14.{
15.     int max;
16.     if (m == 1) max = arr[0];   //递归终止条件
17.     else
18.     {
19.         max = maxm(arr, m - 1);
20.         if (arr[m - 1] > max) max = arr[m - 1];
21.     }

22.     printf("current max: %d\n", max);
23.     return max;
24.}
```

输入：

```
6
3 5 1 7 9 6
```

输出：

```
current max: 3
current max: 5
current max: 5
current max: 7
current max: 9
current max: 9
final max: 9
```

函数综合应用举例

- 【例9-12】输入n个整数，找出其中最大的数（变长参数实现）

```
1. #include <stdio.h>
2. #include <stdarg.h>

3. int va_sum (int cnt, ...);

4. int main ()
5. {
6.     int sum_val = va_max(6, 3, 5, 1, 7, 9, 6);
7.     printf("%d", sum_val);
8.     return 0;
9. }
```

```
10.int va_max(int cnt, ...)
11.{
12.    int result = 0, tmp, i, max = -32768;
13.    va_list ap;
14.    va_start(ap, cnt);

15.    for (i = 0; i < cnt; i++) {
16.        tmp = va_arg(ap, int);
17.        if (tmp > max) max = tmp;
18.    }

19.    va_end(ap);
20.    return result;
21.}
```

9.4 变量的作用域、存储类别和生命周期

- 作用域
  - 局部变量
  - 全局变量
- 存储类别
  - 静态存储方式
  - 动态存储方式
- 生命周期
  - 函数调用实例
  - 程序运行实例

9.4.1 变量的作用域

- 定义变量可能有3种情况:
  - (1) 在函数的开头定义（局部变量）：在一个函数内部定义的变量只在本函数范围内有效，也就是说只有在本函数内才能引用它们，在此函数以外不能使用这些变量;
  - (2) 在函数内的复合语句内定义（局部变量）：在复合语句内定义的变量只在本复合语句范围内有效，只有在本复合语句内才能引用它们。在该复合语句以外不能使用这些变量;
  - (3) 在函数的外部定义（全局变量）：程序的编译单位是源程序文件,一个源文件可以包含一个或若干个函数。在函数之外定义的变量称为外部变量,外部变量是全局变量(也称全程变量)。全局变量可以为本文件中其他函数所共用。它的有效范围为从定义变量的位置开始到本源文件结束

局部变量

```
float f1(int a)          //定义函数f1
{int b,c;                //在函数f1中定义b,c
    ⋮                    } a，b，c有效
}
char f2(int x,int y)     //定义函数f2
{int i,j;
    ⋮                    } x，y，i，j有效
}
int main()               //主函数
{int m,n;
    ⋮                    } m，n有效
return 0;
}
```

(1) 主函数中定义的变量也只在主函数中有效。主函数也不能使用其他函数中定义的变量。

(2) 不同函数中可以使用同名的变量，它们代表不同的对象，互不干扰。

(3) 形式参数也是局部变量。只在定义它的函数中有效。其他函数中不能直接引用形参。

(4) 在一个函数内部，可以在复合语句中定义变量，这些变量只在本复合语句中有效，这种复合语句也称为“**分程序**”或“**程序块**”。

```
int main ()
{    int a,b;                                  } a,b在此范围内有效
       ⋮
     {    int c;
          c=a+b;                               } c在此复合语句内有效
            ⋮
     }
       ⋮
}
```

全局变量

```
int p=1,q=5;            //定义外部变量
float f1(int a)         //定义函数f1
{
    int b,c;            //定义局部变量
      ⋮
}
char c1,c2;             //定义外部变量
char f2 (int x, int y)  //定义函数f2
{
    int i,j;
      ⋮
}
int main()              //主函数
{
    int m,n;
      ⋮
    return 0;
}
```

全局变量c1,c2的作用范围

全局变量p,q的作用范围

设置全局变量的作用是增加了函数间数据联系的渠道。由于同一文件中的所有函数都能引用全局变量的值，因此如果在一个函数中改变了全局变量的值，就能影响到其他函数中全局变量的值。相当于各个函数间有直接的传递通道。

由于函数的调用只能带回一个函数返回值，因此有时可以利用全局变量来增加函数间的联系渠道，通过函数调用能得到一个以上的值。

*为了便于区别全局变量和局部变量，在C程序设计人员中有一个习惯（但非规定），将全局变量名的第1个字母用大写表示。

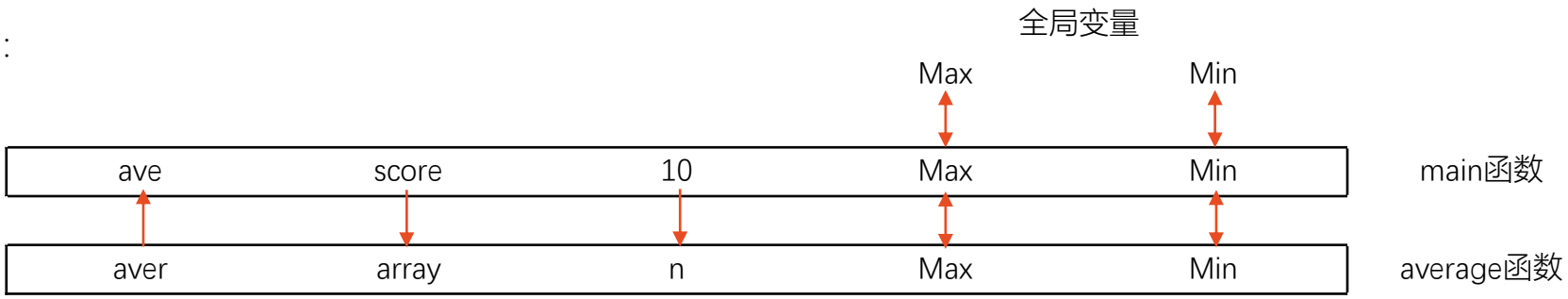
局部变量与全局变量实例

- 【例9-13】有一个一维数组，内放10个学生成绩，写一个函数，当主函数调用此函数后，能求出平均分、最高分和最低分。

```
1.  #include <stdio.h>
2.  float Max=0,Min=0;        //定义全局变量Max,Min

3.  int main()
4.  {
5.    float ave, score[10];
6.    int i;
7.    printf("Please enter 10 scores:");
8.    for(i=0;i<10;i++) scanf("%f",&score[i]);
9.    ave = average(score,10);
10.   printf("max=%6.2f\nmin=%6.2f\naverage=%6.2f\n",
11.          Max, Min, ave);
12.   return 0;
13. }
```

```
14. float average(float array[],int n)    //定义函数，有一形参是数组
15. {
16.   int i;
17.   float aver,sum=array[0];
18.   Max = Min = array[0];
19.   for(i=1; i<n; i++)
20.   {
21.     if (array[i]>Max) Max = array[i];
22.     else if (array[i]<Min) Min = array[i];
23.     sum = sum + array[i];
24.   }
25.   aver=sum/n;
26.   return(aver);
27. }
```

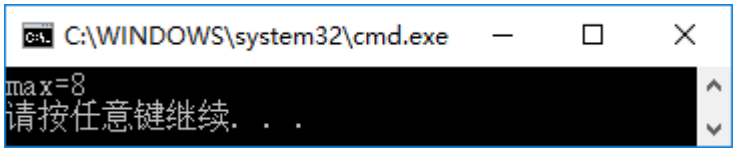
变量的关系：



局部变量与全局变量实例

- 【例9-14】若外部变量与局部变量同名，分析结果。

```
1.   #include <stdio.h>

2.   int a=3,b=5;                    //a,b是全局变量
3.   int max(int a,int b);           //函数声明。a,b是形参

4.   int main()
5.   {
6.     int a=8;                      //a是局部变量
7.     printf("max=%d\n",max(a,b));
8.     return 0;
9.   }

10.  int max(int a,int b)            //a,b是函数形参
11.  {
12.    int c;
13.    c=a>b?a:b;                    //把a和b中的大者存放在c中
14.    return(c);
15.  }
```



程序第2行定义了全局变量a和b，并对其初始化。

第3行是main函数，在main函数中(第6行)定义了一个局部变量a。局部变量a的作用范围为第6~8行。在此范围内全局变量a被局部变量a屏蔽，相当于全局变量a在此范围内不存在(即它不起作用)，而全局变量b在此范围内有效。因此第6行中max(a,b)的实参a应是局部变量a，所以max(a,b)相当于max(8,5)。它的值为8。

第10行起定义max函数，形参a和b是局部变量。全局变量a和b在max函数范围内不起作用，所以函数max中的a和b不是全局变量a和b，而是形参a和b，它们的值是由实参传给形参的，即8和5。

9.4.2 变量的存储类别和生命周期

- 静态存储方式与动态存储方式
  - 静态存储方式是指在程序运行期间由系统分配固定的存储空间的方式。静态存储的变量在程序运行的整个过程都是存在的
  - 动态存储方式则是在程序运行期间根据需要进行动态的分配存储空间的方式。动态存储的变量在函数调用结束后其存储单元就马上释放，变量不存在了
- C语言每一个变量和函数都有两个属性：数据类型和数据的存储类别
  - 存储类别指的是数据在内存中存储的方式（如静态存储和动态存储）
  - 在定义和声明变量和函数时，一般应同时指定其数据类型和存储类别，也可采用默认方式指定（即若用户不指定，系统会隐含指定为某一种存储类别）
  - C语言的存储类别包括4种：**自动的**（auto）、**静态的**（static）、**寄存器的**（register）、**外部的**（extern）。根据变量的存储类别，可以知道变量的作用域和生存期。

9.4.2 变量的存储类别和生命周期

数据分别存放在静态存储区和动态存储区中。全局变量全部存放在静态存储区中，在程序开始执行时给全局变量分配存储区，程序执行完毕就释放。在程序执行过程中它们占据固定的存储单元，而不是动态地进行分配和释放。

在动态存储区中存放以下数据:

① 函数形式参数。在调用函数时给形参分配存储空间。

② 函数中定义的没有用关键字static声明的变量，即自动变量。

③ 函数调用时的现场保护和返回地址等。

对以上这些数据，在函数调用开始时分配动态存储空间，函数结束时释放这些空间。在程序执行过程中，这种分配和释放是动态的，如果在一个程序中两次调用同一函数，而在此函数中定义了局部变量，在两次调用时分配给这些局部变量的存储空间的地址可能是不相同的。

如果一个程序中包含若干个函数，每个函数中的局部变量的生存期并不等于整个程序的执行周期，它只是程序执行周期的一部分。在程序执行过程中，先后调用各个函数，此时会动态地分配和释放存储空间。

局部变量的存储类别

- 自动变量（auto变量）

函数中的局部变量，如果不专门声明为static（静态）存储类别，都是动态地分配存储空间的，数据存储在动态存储区中。函数中的形参和在函数中定义的局部变量（包括在复合语句中定义的局部变量），都属于此类。在调用该函数时，系统会给这些变量分配存储空间，在函数调用结束时就自动释放这些存储空间。因此这类局部变量称为**自动变量**。自动变量用关键字auto作存储类别的声明。

```
int f(int a)                    //定义f函数，a为形参
{
      auto int b,c=3;           //定义b,c为自动变量
        ⋮
}
```

实际上，关键字auto可以省略，**不写auto则隐含指定为“自动存储类别”**，它属于动态存储方式。程序中大多数变量属于自动变量。

```
auto int b,c=3;       //等价于int b,c=3;
```

局部变量的存储类别

- 静态局部变量（static局部变量）

(1) 静态局部变量属于静态存储类别，在静态存储区内分配存储单元。在程序整个运行期间都不释放。而自动变量（即动态局部变量）属于动态存储类别，分配在动态存储区空间而不在静态存储区空间，函数调用结束后即释放。

(2) 对静态局部变量是在编译时赋初值的，即只赋初值一次，在程序运行时它已有初值。以后每次调用函数时不再重新赋初值而只是保留上次函数调用结束时的值。而对自动变量赋初值，不是在编译时进行的，而是在函数调用时进行的，每调用一次函数重新给一次初值，相当于执行一次赋值语句。

(3) 如果在定义局部变量时不赋初值的话，则对静态局部变量来说，编译时自动赋初值0（对数值型变量）或空字符'\0'（对字符变量）。而对自动变量来说，它的值是一个不确定的值。这是由于每次函数调用结束后存储单元已释放，下次调用时又重新另分配存储单元，而所分配的单元中的内容是不可知的。

(4) 虽然静态局部变量在函数调用结束后仍然存在，但其他函数是不能引用它的。因为它是局部变量，只能被本函数引用，而不能被其他函数引用。

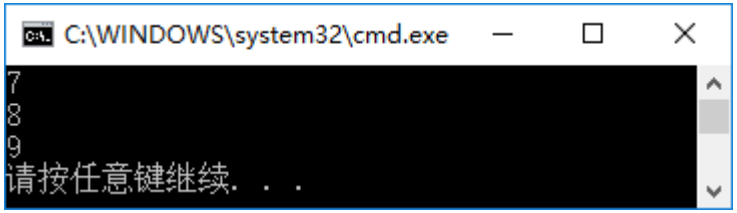
局部变量的存储类别

- 【例9-15】考察静态局部变量的值。

```
1.   #include <stdio.h>
2.   int f(int);                          //函数声明
3.   int main()
4.   {
5.     int a=2,i;                         //自动局部变量
6.     for(i=0;i<3;i++)
7.       printf("%d\n",f(a));             //输出f(a)的值
8.     return 0;
9.   }

10.  int f(int a)
11.  {
12.    auto int b=0;                      //自动局部变量
13.    static int c=3;                    //静态局部变量
14.    b = b+1;
15.    c = c+1;
16.    return (a+b+c);
17.  }
```

静态变量与自动变量的值的比较分析

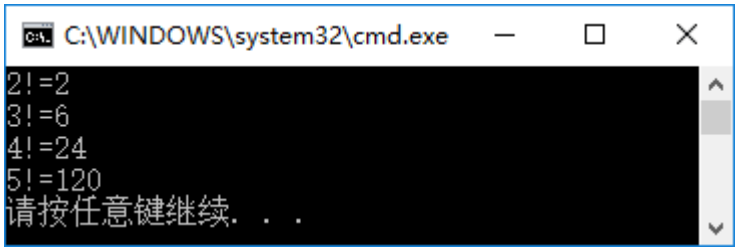
第几次调用	调用时初值		调用结束时的值		
	b	c	b	c	a+b+c
第1次	0	3	1	4	7
第2次	0	4	1	5	8
第3次	0	5	1	6	9



局部变量的存储类别

- 【例9-16】输出1到5的阶乘值。

```
1.   #include <stdio.h>
2.   int fac(int n);

3.   int main()
4.   {
5.     int i;
6.     for(i=1;i<=5;i++)    //先后5次调用fac函数
7.       printf("%d!=%d\n",i,fac(i));       //每次计算并输出i!的值
8.     return 0;
9.   }

10.  int fac(int n)
11.  {
12.    static int f=1;            //f保留了上次调用结束时的值
13.    f=f*n;                     //在上次的f值的基础上再乘以n
14.    return(f);                 //返回值f是n!的值
15.  }
```



(1) 每次调用fac(i)，输出一个i!，同时保留这个i!的值以便下次再乘(i+1)。

(2) 如果函数中的变量只被引用而不改变值，则定义为静态局部变量(同时初始化)比较方便，以免每次调用时重新赋值。

注意

- 用静态存储要多占内存（长期占用不释放，而不能像动态存储那样一个存储单元可以先后为多个变量使用，节约内存），而且降低了程序的可读性，当调用次数多时往往弄不清静态局部变量的当前值是什么。因此，若非必要，不要多用静态局部变量。

局部变量的存储类别

寄存器变量(register变量)

一般情况下，变量（包括静态存储方式和动态存储方式）的值是存放在内存中的。当程序中用到哪个变量的值时，由控制器发出指令将内存中该变量的值送到运算器中。 经过运算器进行运算，如果需要存数，再从运算器将数据送到内存存放。

如果有一些变量使用频繁（例如，在一个函数中执行10 000次循环，每次循环中都要引用某局部变量），则为存取变量的值要花费不少时间。为提高执行效率，允许将局部变量的值放在CPU中的寄存器中，需要用时直接从寄存器取出参加运算，不必再到内存中去存取。由于对寄存器的存取速度远高于对内存的存取速度，因此这样做可以提高执行效率。这种变量叫做寄存器变量，用关键字register作声明。如

```
register int  f;        //定义f为寄存器变量
```

由于现在的计算机的速度愈来愈快，性能愈来愈高， 优化的编译系统能够识别使用频繁的变量，从而自动地将这些变量放在寄存器中，而不需要程序设计者指定。因此，现在实际上用register声明变量的必要性不大。

注意

- 3种局部变量的存储位置是不同的: 自动变量存储在动态存储区；静态局部变量存储在静态存储区；寄存器存储在CPU中的寄存器中。

全局变量的存储类别

- 全局变量都是存放在静态存储区中的。因此它们的生存期是固定的，存在于程序的整个运行过程
- 一般来说，外部变量是在函数的外部定义的全局变量，它的作用域是从变量的定义处开始，到本程序文件的末尾。在此作用域内，全局变量可以为程序中各个函数所引用
- 但有时程序设计人员希望能扩展外部变量的作用域，比如在一个文件内扩展外部变量的作用域，将外部变量的作用域扩展到其他文件，或者将外部变量的作用域限制在本文件中

全局变量的存储类别

- 在一个文件内扩展外部变量的作用域

如果外部变量不在文件的开头定义，其有效的作用范围只限于定义处到文件结束。 在定义点之前的函数不能引用该外部变量。

如果由于某种考虑，在定义点之前的函数需要引用该外部变量，则应该在引用之前用关键字**extern**对该变量作“**外部变量声明**”，表示把该外部变量的作用域扩展到此位置。

有了此声明，就可以从“声明”处起，合法地使用该外部变量。

全局变量的存储类别

- 【例9-17】调用函数，求3个整数中的大者。

注意

- 提倡将外部变量的定义放在引用它的所有函数之前，这样可以避免在函数中多加一个extern声明。
- 用extern声明外部变量时，类型名可以写也可以省写。例如，“extern int A,B,C;”也可以写成“extern A,B,C;”。因为它不是定义变量，可以不指定类型，只须写出外部变量名即可。

```
1.   #include <stdio.h>

2.   extern int A,B,C;      //把外部变量A,B,C的作用域扩展到从此处开始
3.   int main() {
4.     printf("Please enter three integer numbers:");
5.     scanf("%d %d %d",&A,&B,&C);          //输入3个整数给A,B,C
6.     printf("max is %d\n",max());
7.     return 0;
8.   }
9.   int A,B,C;              //定义外部变量A,B,C
10.  int max() {
11.    int m;
12.    m = A>B ? A : B;     //把A和B中的大者放在m中
13.    if(C>m) m=C;         //将A,B,C三者中的大者放在m中
14.    return(m);           //返回m的值
15.  }
```

全局变量的存储类别

将外部变量的作用域扩展到其他文件

如果一个程序包含两个文件，在两个文件中都要用到同一个外部变量Num，不能分别在两个文件中各自定义一个外部变量Num，否则在进行程序的连接时会出现“重复定义”的错误。正确的做法是：在任一个文件中定义外部变量Num，而在另一文件中用extern对Num作“外部变量声明”，即“extern Num; ”。

在编译和连接时，系统会由此知道Num有“外部链接”，可以从别处找到已定义的外部变量Num，并将在另一文件中定义的外部变量Num的作用域扩展到本文件，在本文件中可以合法地引用外部变量Num。

extern既可以用来扩展外部变量在本文件中的作用域，又可以使外部变量的作用域从一个文件扩展到程序中的其他文件，系统在编译过程中遇到extern时，

- 先在本文件中找外部变量的定义，如果找到，就在本文件中扩展作用域；
- 如果找不到，就在连接时从其他文件中找外部变量的定义。如果从其他文件中找到了，就将作用域扩展到本文件；
- 如果再找不到，就按出错处理。

全局变量的存储类别

- 【例9-18】给定b的值，输入a和m，求a*b和am的值。

file1.c

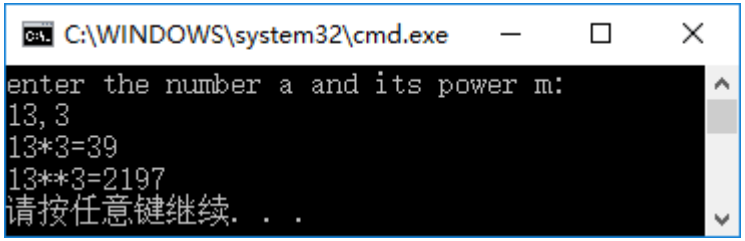
```
#include <stdio.h>
int A;                          //定义外部变量
int power(int);                 //函数声明
int main() {
  int b=3,c,d,m;
  printf("enter the number a and its power m:\n");
  scanf("%d,%d",&A,&m);
  c=A*b;
  printf("%d*%d=%d\n",A,b,c);
  d=power(m);
  printf("%d**%d=%d\n",A,m,d);
  return 0;
}
```

file2.c

```
extern A;
//把file1中定义的外部变量的作用域扩展到本文件
int power(int n)
{
  int i,y=1;
  for(i=1;i<=n;i++) y*=A;
  return(y);
}
```

注意

- 用这种方法扩展全局变量的作用域应十分慎重，因为在执行一个文件中的操作时，可能会改变该全局变量的值，会影响到另一文件中全局变量的值，从而影响该文件中函数的执行结果。



全局变量的存储类别

- 将外部变量的作用域限制在本文件中

有时在程序设计中希望某些外部变量只限于被本文件引用，而不能被其他文件引用。这时可以在定义外部变量时加一个static声明。

file1.c

```
static int A;
int main()
{
        ⋮
}
```

file2.c

```
extern A;
void fun(int n)
{
        ⋮
    A=A*n;        //出错
        ⋮
}
```

这种加上static声明、只能用于本文件的外部变量称为**静态外部变量**。在程序设计中，常由若干人分别完成各个模块，各人可以独立地在其设计的文件中使用相同的外部变量名而互不相干。只须在每个文件中定义外部变量时加上static即可。这就为程序的模块化、通用性提供方便。如果已确认其他文件不需要引用本文件的外部变量，就可以对本文件中的外部变量都加上static，成为静态外部变量，以免被其他文件误用。至于在各文件中在函数内定义的局部变量，本来就不能被函数外引用，更不能被其他文件引用，因此是安全的。

全局变量的存储类别

- 将外部变量的作用域限制在本文件中

不要误认为对外部变量加static声明后才采取静态存储方式（存放在静态存储区中），而不加static的是采取动态存储（存放在动态存储区）。声明局部变量的存储类型和声明全局变量的存储类型的含义是不同的。

对于局部变量来说，声明存储类型的作用是指定变量存储的区域(静态存储区或动态存储区)以及由此产生的生存期的问题，而对于全局变量来说，由于都是在编译时分配内存的，都存放在静态存储区，声明存储类型的作用是变量作用域的扩展问题。

用static声明一个变量的作用是:

(1) 对局部变量用static声明，把它分配在静态存储区，该变量在整个程序执行期间不释放，其所分配的空间始终存在。

(2) 对全局变量用static声明，则该变量的作用域只限于本文件模块(即被声明的文件中)。

注意

- 用auto，register和static声明变量时，是在定义变量的基础上加上这些关键字，而不能单独使用。

```
int a;//先定义整型变量a
static a;    //企图再将变量a声明为静态变量
```

重新定义

变量的作用域、存储类别和生命周期小结

对一个数据的定义，需要指定两种属性: **数据类型**和**存储类别**，分别使用两个关键字。

```
static int a;        //静态局部整型变量或静态外部整型变量
auto char c;         //自动变量，在函数内定义
register int d;      //寄存器变量，在函数内定义
```

此外，可以用extern声明已定义的外部变量。

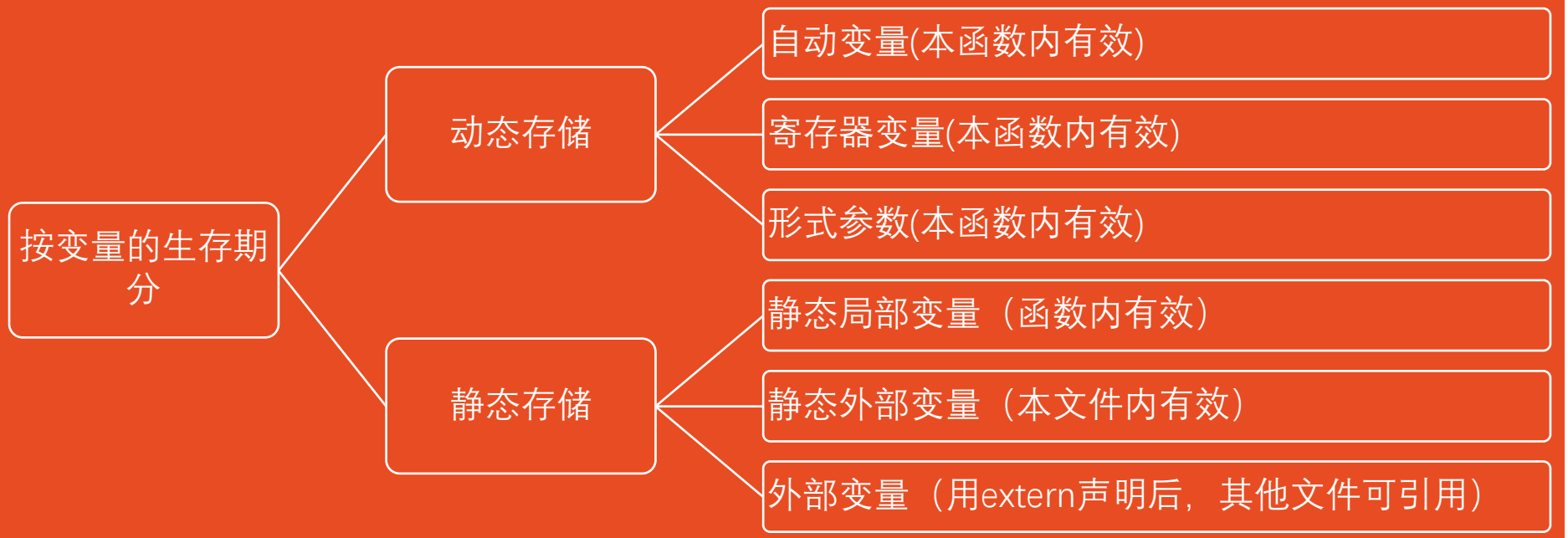
```
extern b;            //将已定义的外部变量b的作用域扩展至此
```

变量的作用域、存储类别和生命周期小结

(1) 从作用域角度分，有局部变量和全局变量。它们采用的存储类别如下:

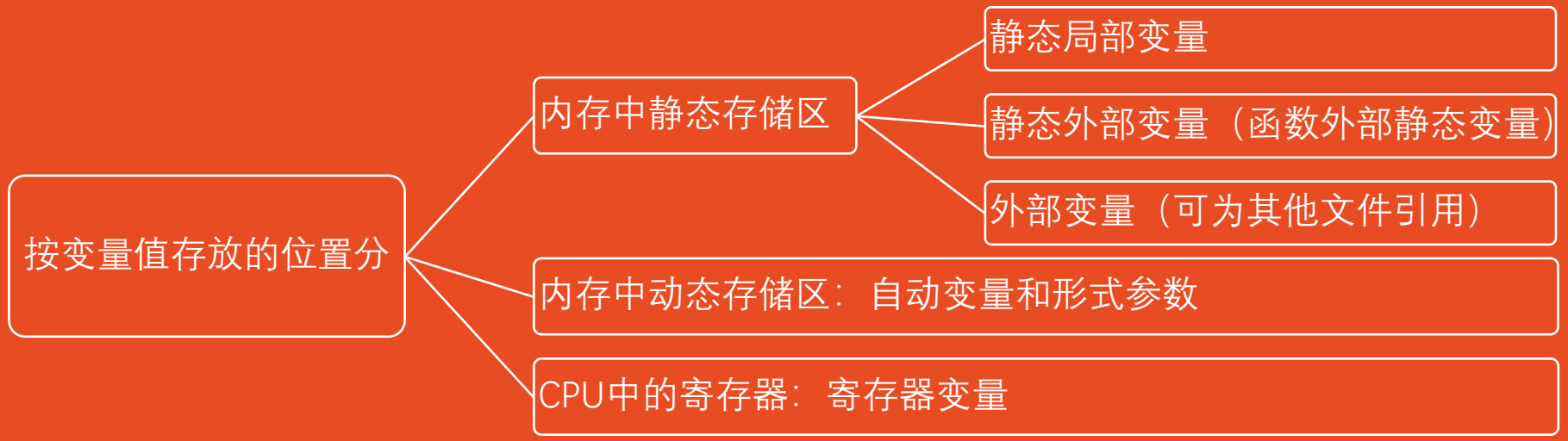
变量的作用域、存储类别和生命周期小结

(2)从变量存在的时间(生存期)来区分,有动态存储和静态存储两种类型。静态存储是程序整个运行时间都存在,而动态存储则是在调用函数时临时分配单元。



变量的作用域、存储类别和生命周期小结

(3)从变量值存放的位置来区分,可分为:



变量的作用域、存储类别和生命周期小结

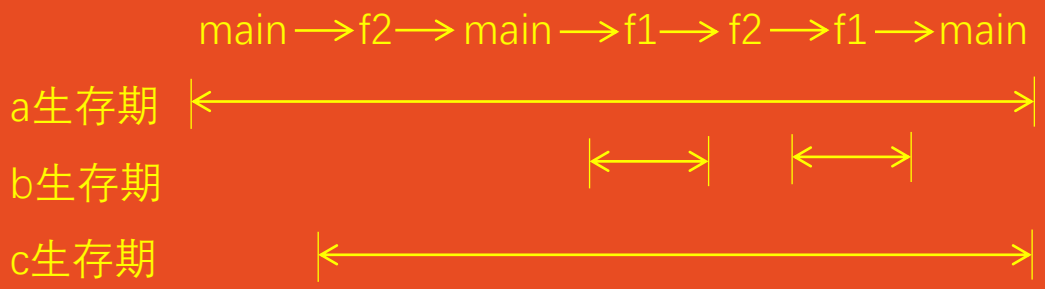
```
int a;
int main()
{
        f2();
        f1();
}
void f1()
{
        auto int b;
        f2();
}
void f2()
{
        static int c;
}
```

a作用域

b作用域

c作用域

(4)关于作用域和生存期的概念。从前面叙述可以知道，对一个变量的属性可以从两个方面分析，一是变量的作用域，一是变量值存在时间的长短，即生存期。前者是从空间的角度，后者是从时间的角度。二者有联系但不是同一回事。



如果一个变量在某个文件或函数范围内是有效的，就称该范围为该变量的**作用域**，在此作用域内可以引用该变量，在专业书中称变量在此作用域内“**可见**”，这种性质称为变量的**可见性**。如果一个变量值在某一时刻是存在的，则认为这一时刻属于该变量的**生存期**，或称该变量在此时刻“**存在**”。

变量的作用域、存储类别和生命周期小结

(5)static对局部变量和全局变量的作用不同。

对局部变量来说，它使变量由动态存储方式改变为静态存储方式。

而对全局变量来说，它使变量局部化(局部于本文件)，但仍为静态存储方式。

从作用域角度看，凡有static声明的，其作用域都是局限的，或者局限于本函数内(静态局部变量)，或者局限于本文件内(静态外部变量)。

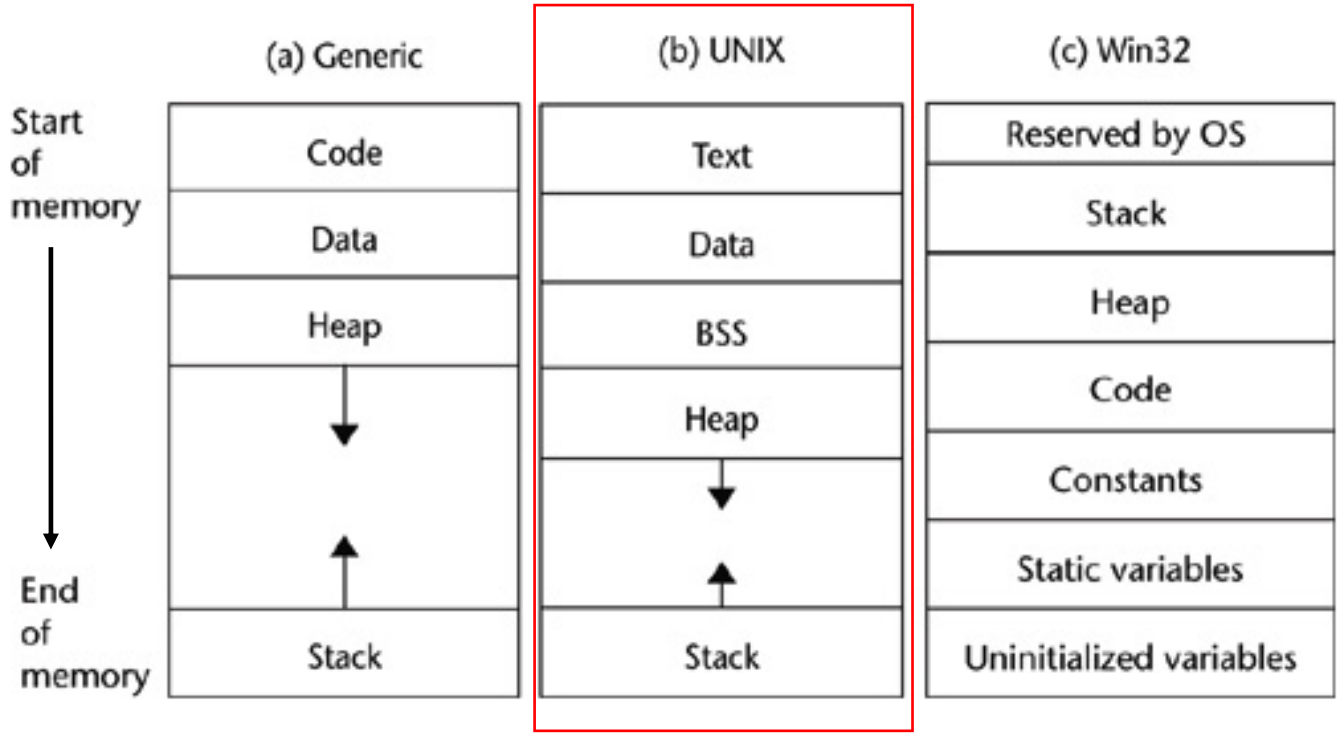
变量的作用域、存储类别和生命周期小结

各种类型变量的作用域和存在性的情况

变量存储类别	函数内		函数外	
	作用域	存在性	作用域	存在性
自动变量和寄存器变量	√	√	×	×
静态局部变量	√	√	×	√
静态外部变量	√	√	√ (只限本文件)	√
外部变量	√	√	√	√

9.5 函数调用与变量存储揭秘

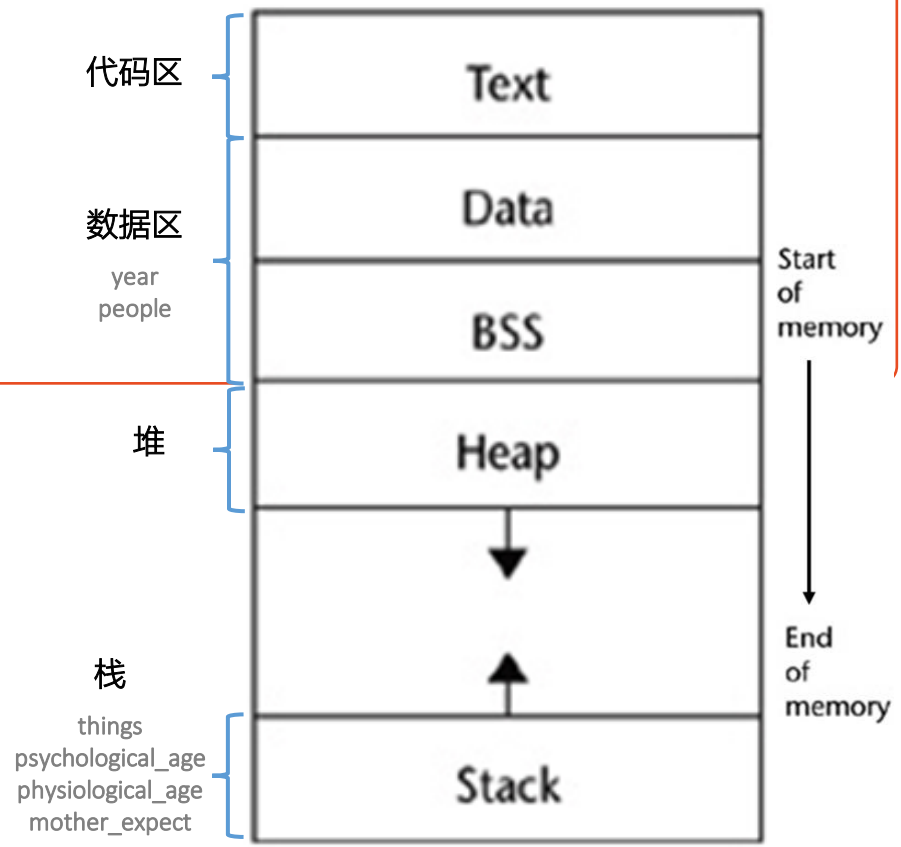
- 各平台上进程内存组织示意



```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int people;

int go_far(int psychological_age,
           int *physiological_age) {
  static int year = 0;
  int experience, things = 0;

  psychological_age++; (*physiological_age)++;
  year++, things++;
  experience = (rand()%200 - 100);
  people += experience * year;
  printf("year(%p): %d\n", &year, year);
  printf("people(%p): %d\n", &people, people);
  printf("things(%p): %d\n", &things, things);
  printf("psychological_age: %p\n", psychological_age);
  printf("physiological_age: %p\n", physiological_age);
  if (year < 50) experience +=
    go_far(psychological_age, physiological_age);
  printf("\n");
  return experience;
}
```

```c
int main(int argc, char **argv) {
  int psychological_age = 18;
  int physiological_age = 18;
  int mother_expect;
  srand(time(NULL));
  mother_expect = go_far(psychological_age,
                         &physiological_age);
  printf("psychological_age(%p): %d\n",
         &psychological_age, psychological_age);
  printf("physiological_age(%p): %d\n",
         &physiological_age, physiological_age);
  printf("mother_exprect(%p): %d\n",
         &mother_expect, mother_expect);
}
```

```
year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.....
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101
```

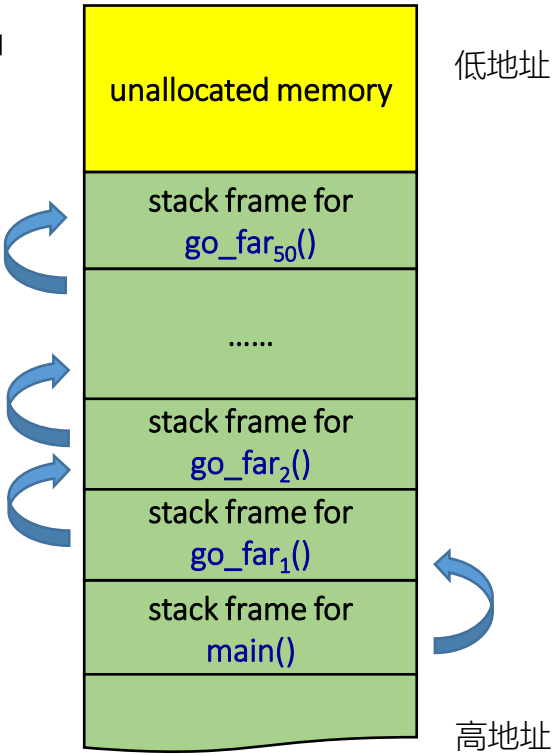


以32位x86体系
UNIX平台为例

9.5.1 栈管理

- 程序的栈（stack）用于存储以下信息保证程序的正常执行：
  - 调用函数的返回地址
  - 函数参数，局部变量
- 栈桢（frame）：进程中每个函数被调用时在栈中压入的一段数据
  - 在Intel架构下，当前帧的地址被存储在ebp寄存器中
  - 在调用一个函数时，调用函数的帧地址也需要入栈

```
go_far()
{
   go_far();
}

main()
 {
   go_far();
}
```

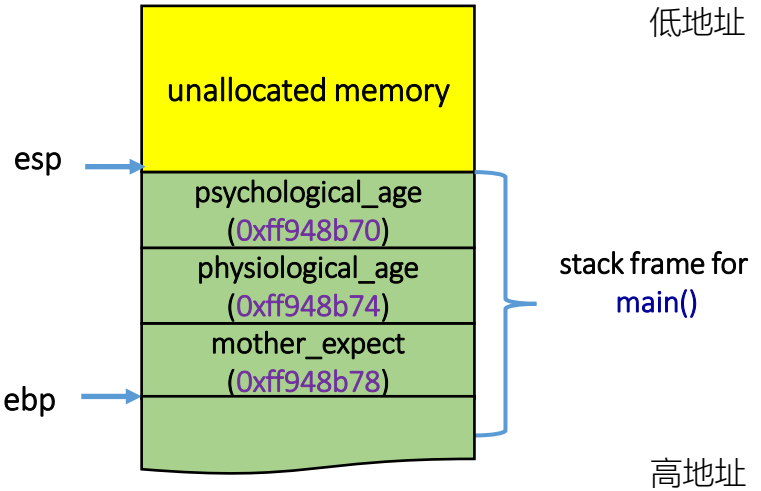


以32位x86体系
UNIX平台为例

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int people;

int go_far(int psychological_age,
           int *physiological_age) {
  static int year = 0;
  int experience, things = 0;

  psychological_age++; (*physiological_age)++;
  year++, things++;
  experience = (rand()%200 - 100);
  people += experience * year;
  printf("year(%p): %d\n", &year, year);
  printf("people(%p): %d\n", &people, people);
  printf("things(%p): %d\n", &things, things);
  printf("psychological_age: %p\n", psychological_age);
  printf("physiological_age: %p\n", physiological_age);
  if (year < 50) experience +=
    go_far(psychological_age, physiological_age);
  printf("\n");
  return experience;
}
```

```c
int main(int argc, char **argv) {
  int psychological_age = 18;
  int physiological_age = 18;
  int mother_expect;
  srand(time(NULL));
  mother_expect = go_far(psychological_age,
                         &physiological_age);
  printf("psychological_age(%p): %d\n",
         &psychological_age, psychological_age);
  printf("physiological_age(%p): %d\n",
         &physiological_age, physiological_age);
  printf("mother_exprect(%p): %d\n",
         &mother_expect, mother_expect);
}
```

```
year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.....
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101
```



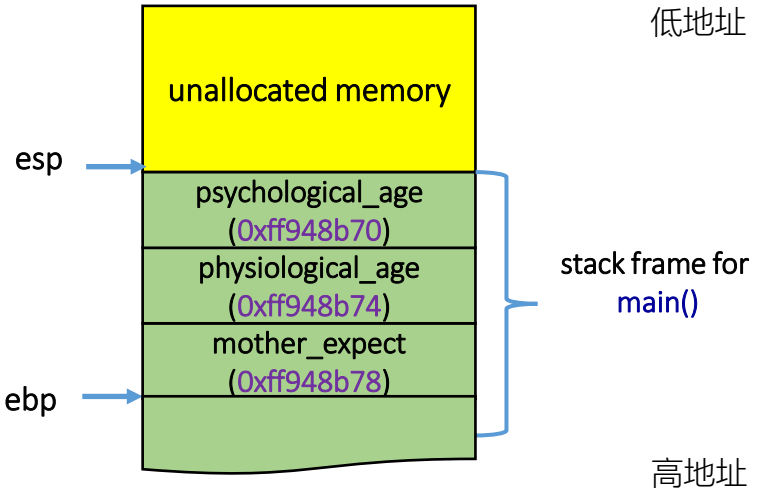
以32位x86体系
UNIX平台为例

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int people;

int go_far(int psychological_age,
           int *physiological_age) {
  static int year = 0;
  int experience, things = 0;

  psychological_age++; (*physiological_age)++;
  year++, things++;
  experience = (rand()%200 - 100);
  people += experience * year;
  printf("year(%p): %d\n", &year, year);
  printf("people(%p): %d\n", &people, people);
  printf("things(%p): %d\n", &things, things);
  printf("psychological_age: %p\n", psychological_age);
  printf("physiological_age: %p\n", physiological_age);
  if (year < 50) experience +=
    go_far(psychological_age, physiological_age);
  printf("\n");
  return experience;
}
```

```c
int main(int argc, char **argv) {
  int psychological_age = 18;
  int physiological_age = 18;
  int mother_expect;
  srand(time(NULL));
  mother_expect = go_far(psychological_age,
                         &physiological_age);
  printf("psychological_age(%p): %d\n",
         &psychological_age, psychological_age);
  printf("physiological_age(%p): %d\n",
         &physiological_age, physiological_age);
  printf("mother_exprect(%p): %d\n",
         &mother_expect, mother_expect);
}
```

main调用go_far_1

```
year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.....
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101
```



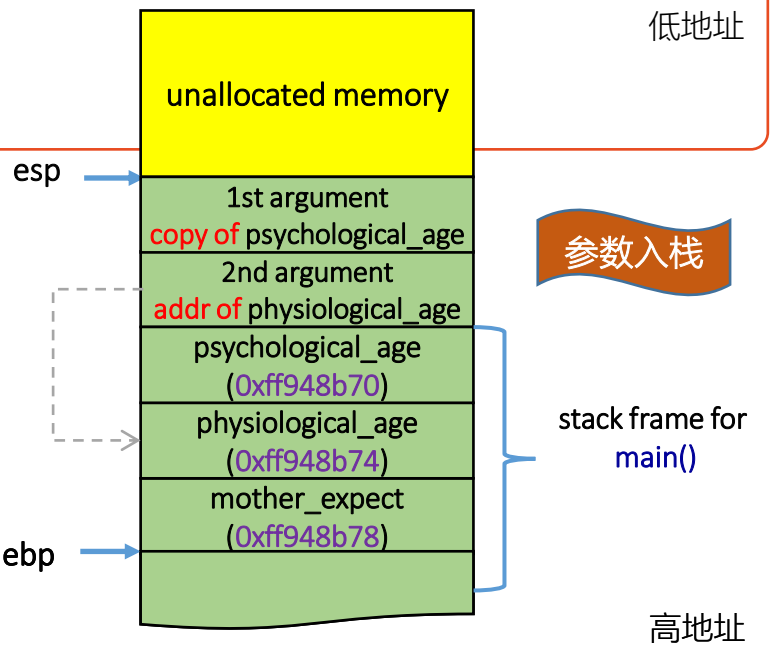
以32位x86体系
UNIX平台为例

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int people;

int go_far(int psychological_age,
           int *physiological_age) {
  static int year = 0;
  int experience, things = 0;

  psychological_age++; (*physiological_age)++;
  year++, things++;
  experience = (rand()%200 - 100);
  people += experience * year;
  printf("year(%p): %d\n", &year, year);
  printf("people(%p): %d\n", &people, people);
  printf("things(%p): %d\n", &things, things);
  printf("psychological_age: %p\n", psychological_age);
  printf("physiological_age: %p\n", physiological_age);
  if (year < 50) experience +=
    go_far(psychological_age, physiological_age);
  printf("\n");
  return experience;
}
```

```c
int main(int argc, char **argv) {
  int psychological_age = 18;
  int physiological_age = 18;
  int mother_expect;
  srand(time(NULL));
  mother_expect = go_far(psychological_age,
                         &physiological_age);
  printf("psychological_age(%p): %d\n",
         &psychological_age, psychological_age);
  printf("physiological_age(%p): %d\n",
         &physiological_age, physiological_age);
  printf("mother_exprect(%p): %d\n",
         &mother_expect, mother_expect);
}
```

```
year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.....
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101
```

main调用go_far$_1$



以32位x86体系
UNIX平台为例

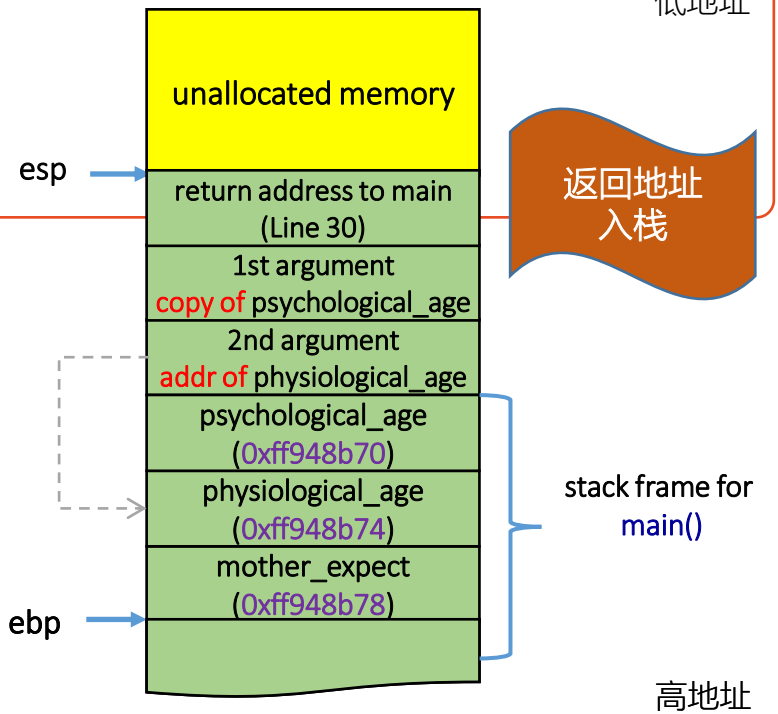
```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int people;

int go_far(int psychological_age,
           int *physiological_age) {
  static int year = 0;
  int experience, things = 0;

  psychological_age++; (*physiological_age)++;
  year++, things++;
  experience = (rand()%200 - 100);
  people += experience * year;
  printf("year(%p): %d\n", &year, year);
  printf("people(%p): %d\n", &people, people);
  printf("things(%p): %d\n", &things, things);
  printf("psychological_age: %p\n", psychological_age);
  printf("physiological_age: %p\n", physiological_age);
  if (year < 50) experience +=
    go_far(psychological_age, physiological_age);
  printf("\n");
  return experience;
}
```

```c
int main(int argc, char **argv) {
  int psychological_age = 18;
  int physiological_age = 18;
  int mother_expect;
  srand(time(NULL));
  mother_expect = go_far(psychological_age,
                         &physiological_age);
  printf("psychological_age(%p): %d\n",
         &psychological_age, psychological_age);
  printf("physiological_age(%p): %d\n",
         &physiological_age, physiological_age);
  printf("mother_exprect(%p): %d\n",
         &mother_expect, mother_expect);
}
```

```
year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.....
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101
```

以32位x86体系
UNIX平台为例

main调用go_far$_1$



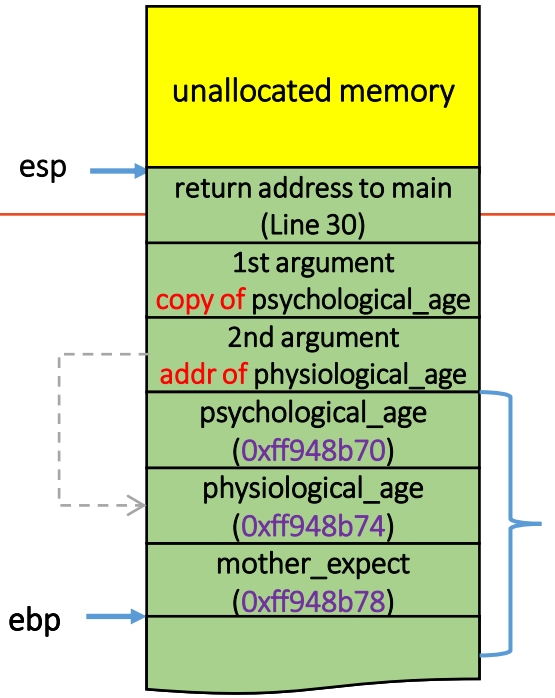
```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <time.h>

4.  int people;

5.  int go_far(int psychological_age,
6.              int *physiological_age) {
7.    static int year = 0;
8.    int experience, things = 0;

9.    psychological_age++; (*physiological_age)++;
10.   year++, things++;
11.   experience = (rand()%200 - 100);
12.   people += experience * year;
13.   printf("year(%p): %d\n", &year, year);
14.   printf("people(%p): %d\n", &people, people);
15.   printf("things(%p): %d\n", &things, things);
16.   printf("psychological_age: %p\n", psychological_age);
17.   printf("physiological_age: %p\n", physiological_age);
18.   if (year < 50) experience +=
19.     go_far(psychological_age, physiological_age);
20.   printf("\n");
21.   return experience;
22. }
```

```
23. int main(int argc, char **argv) {
24.   int psychological_age = 18;
25.   int physiological_age = 18;
26.   int mother_expect;
27.   srand(time(NULL));
28.   mother_expect = go_far(psychological_age,
29.                          &physiological_age);
30.   printf("psychological_age(%p): %d\n",
31.          &psychological_age, psychological_age);
32.   printf("physiological_age(%p): %d\n",
33.          &physiological_age, physiological_age);
34.   printf("mother_exprect(%p): %d\n",
35.          &mother_expect, mother_expect);
36. }
```

```
year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.....
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101
```

进入go_far$_1$

低地址



高地址

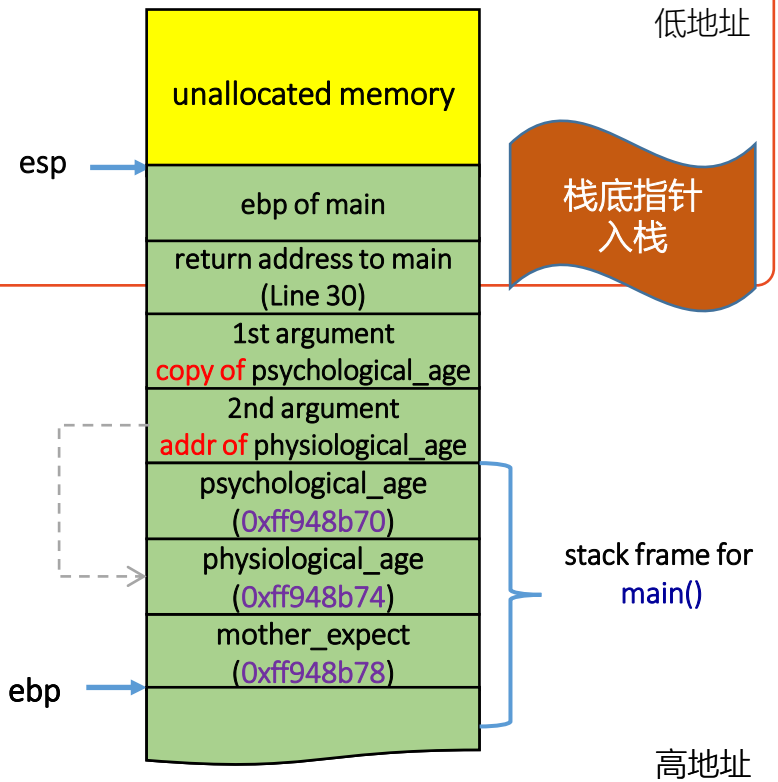
以32位x86体系
UNIX平台为例

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int people;

int go_far(int psychological_age,
           int *physiological_age) {
  static int year = 0;
  int experience, things = 0;

  psychological_age++; (*physiological_age)++;
  year++, things++;
  experience = (rand()%200 - 100);
  people += experience * year;
  printf("year(%p): %d\n", &year, year);
  printf("people(%p): %d\n", &people, people);
  printf("things(%p): %d\n", &things, things);
  printf("psychological_age: %p\n", psychological_age);
  printf("physiological_age: %p\n", physiological_age);
  if (year < 50) experience +=
    go_far(psychological_age, physiological_age);
  printf("\n");
  return experience;
}
```

```c
int main(int argc, char **argv) {
  int psychological_age = 18;
  int physiological_age = 18;
  int mother_expect;
  srand(time(NULL));
  mother_expect = go_far(psychological_age,
                         &physiological_age);
  printf("psychological_age(%p): %d\n",
         &psychological_age, psychological_age);
  printf("physiological_age(%p): %d\n",
         &physiological_age, physiological_age);
  printf("mother_exprect(%p): %d\n",
         &mother_expect, mother_expect);
}
```

```
year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.....
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101
```

进入go_far$_1$



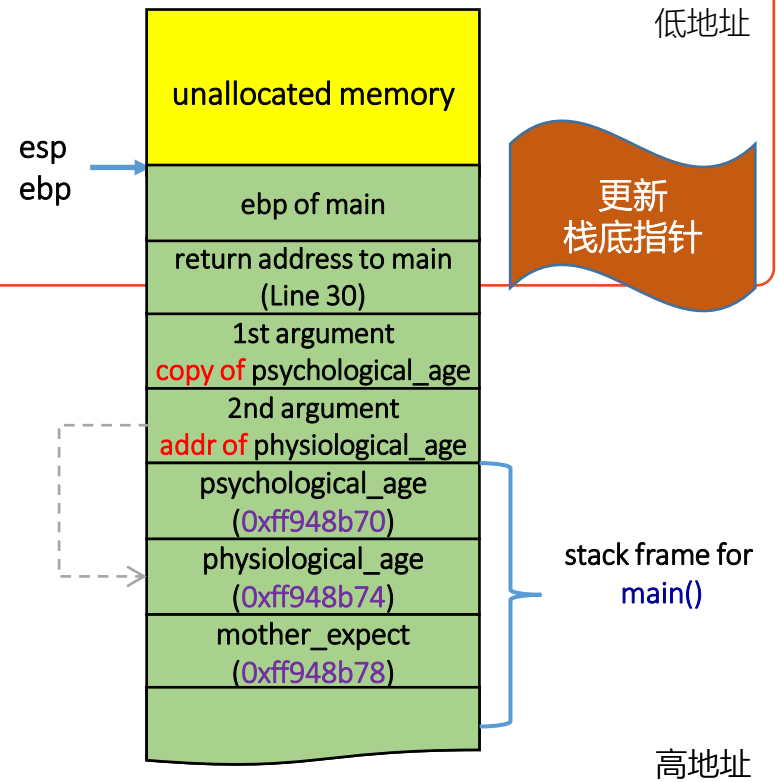
以32位x86体系
UNIX平台为例

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <time.h>

4.  int people;

5.  int go_far(int psychological_age,
6.              int *physiological_age) {
7.    static int year = 0;
8.    int experience, things = 0;

9.    psychological_age++; (*physiological_age)++;
10.   year++, things++;
11.   experience = (rand()%200 - 100);
12.   people += experience * year;
13.   printf("year(%p): %d\n", &year, year);
14.   printf("people(%p): %d\n", &people, people);
15.   printf("things(%p): %d\n", &things, things);
16.   printf("psychological_age: %p\n", psychological_age);
17.   printf("physiological_age: %p\n", physiological_age);
18.   if (year < 50) experience +=
19.     go_far(psychological_age, physiological_age);
20.   printf("\n");
21.   return experience;
22. }
```

```
23. int main(int argc, char **argv) {
24.   int psychological_age = 18;
25.   int physiological_age = 18;
26.   int mother_expect;
27.   srand(time(NULL));
28.   mother_expect = go_far(psychological_age,
29.                          &physiological_age);
30.   printf("psychological_age(%p): %d\n",
31.          &psychological_age, psychological_age);
32.   printf("physiological_age(%p): %d\n",
33.          &physiological_age, physiological_age);
34.   printf("mother_exprect(%p): %d\n",
35.          &mother_expect, mother_expect);
36. }
```

```
year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.....
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101
```

进入go_far$_1$



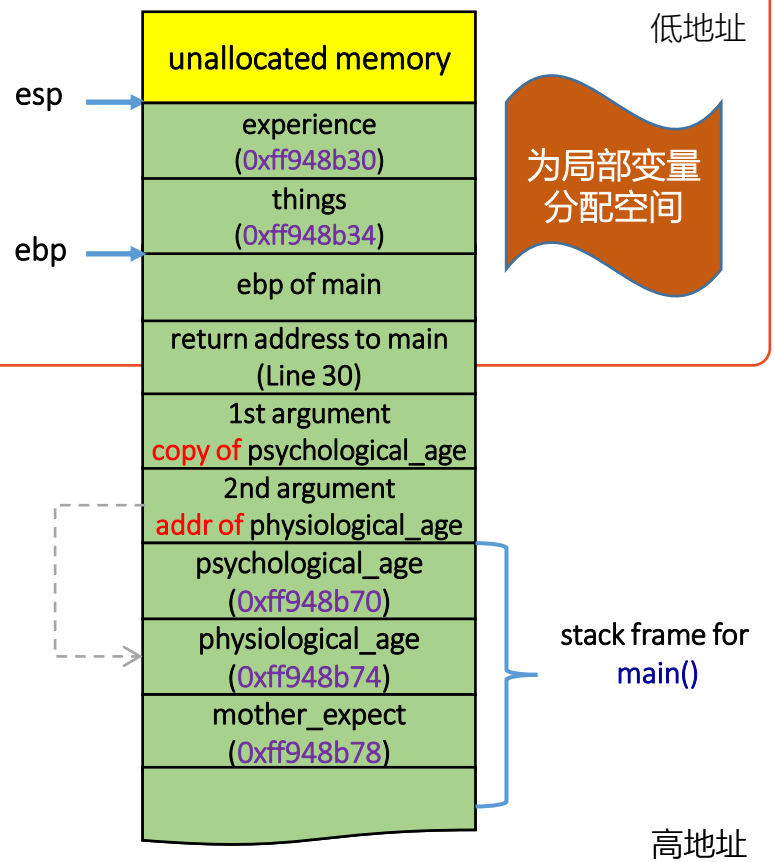
以32位x86体系
UNIX平台为例

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int people;

int go_far(int psychological_age,
           int *physiological_age) {
  static int year = 0;
  int experience, things = 0;

  psychological_age++; (*physiological_age)++;
  year++, things++;
  experience = (rand()%200 - 100);
  people += experience * year;
  printf("year(%p): %d\n", &year, year);
  printf("people(%p): %d\n", &people, people);
  printf("things(%p): %d\n", &things, things);
  printf("psychological_age: %p\n", psychological_age);
  printf("physiological_age: %p\n", physiological_age);
  if (year < 50) experience +=
    go_far(psychological_age, physiological_age);
  printf("\n");
  return experience;
}
```

```c
int main(int argc, char **argv) {
  int psychological_age = 18;
  int physiological_age = 18;
  int mother_expect;
  srand(time(NULL));
  mother_expect = go_far(psychological_age,
                         &physiological_age);
  printf("psychological_age(%p): %d\n",
         &psychological_age, psychological_age);
  printf("physiological_age(%p): %d\n",
         &physiological_age, physiological_age);
  printf("mother_exprect(%p): %d\n",
         &mother_expect, mother_expect);
}
```

```
year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.....
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101
```

进入go_far$_1$



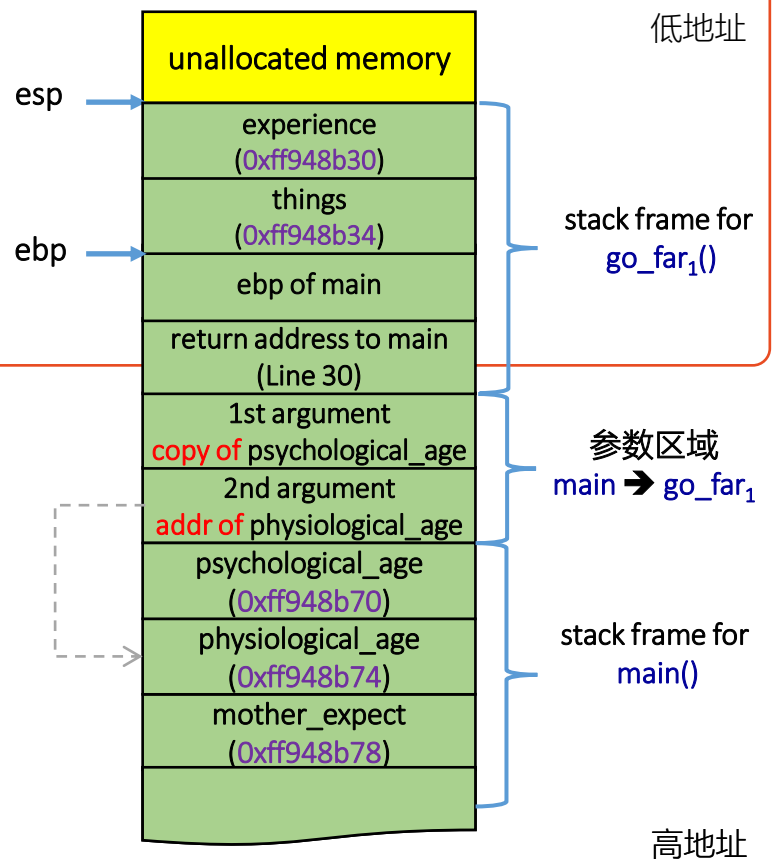
以32位x86体系
UNIX平台为例

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int people;

int go_far(int psychological_age,
           int *physiological_age) {
  static int year = 0;
  int experience, things = 0;

  psychological_age++; (*physiological_age)++;
  year++, things++;
  experience = (rand()%200 - 100);
  people += experience * year;
  printf("year(%p): %d\n", &year, year);
  printf("people(%p): %d\n", &people, people);
  printf("things(%p): %d\n", &things, things);
  printf("psychological_age: %p\n", psychological_age);
  printf("physiological_age: %p\n", physiological_age);
  if (year < 50) experience +=
    go_far(psychological_age, physiological_age);
  printf("\n");
  return experience;
}
```

```c
int main(int argc, char **argv) {
  int psychological_age = 18;
  int physiological_age = 18;
  int mother_expect;
  srand(time(NULL));
  mother_expect = go_far(psychological_age,
                         &physiological_age);
  printf("psychological_age(%p): %d\n",
         &psychological_age, psychological_age);
  printf("physiological_age(%p): %d\n",
         &physiological_age, physiological_age);
  printf("mother_exprect(%p): %d\n",
         &mother_expect, mother_expect);
}
```

```
year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.....
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101
```

进入go_far$_1$



以32位x86体系
UNIX平台为例

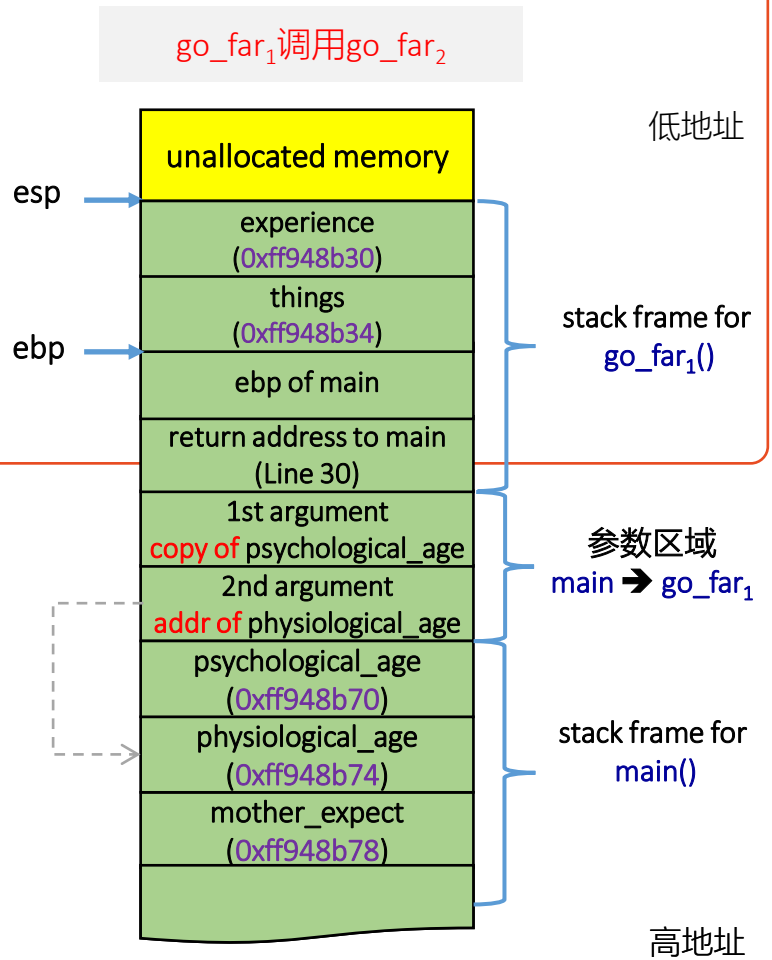
```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <time.h>

4.  int people;

5.  int go_far(int psychological_age,
6.              int *physiological_age) {
7.    static int year = 0;
8.    int experience, things = 0;

9.    psychological_age++; (*physiological_age)++;
10.   year++, things++;
11.   experience = (rand()%200 - 100);
12.   people += experience * year;
13.   printf("year(%p): %d\n", &year, year);
14.   printf("people(%p): %d\n", &people, people);
15.   printf("things(%p): %d\n", &things, things);
16.   printf("psychological_age: %p\n", psychological_age);
17.   printf("physiological_age: %p\n", physiological_age);
18.   if (year < 50) experience +=
19.     go_far(psychological_age, physiological_age);
20.   printf("\n");
21.   return experience;
22. }
```

year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.....
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101

以32位x86体系
UNIX平台为例



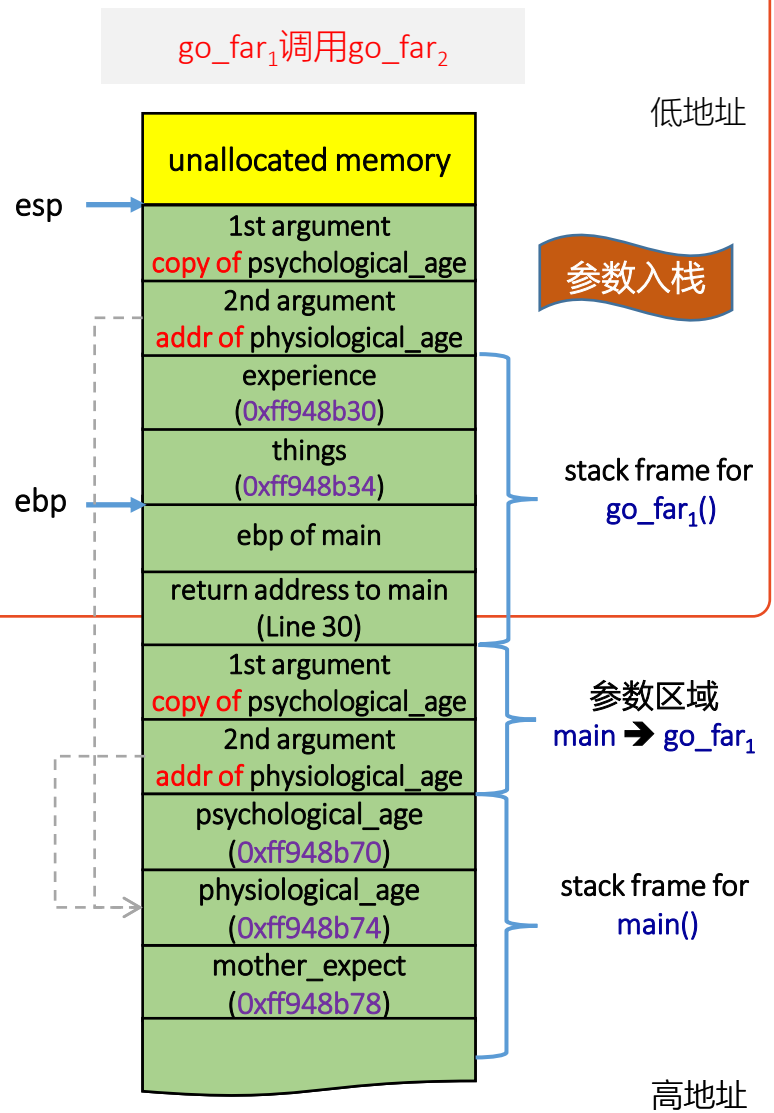
```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <time.h>

4.  int people;

5.  int go_far(int psychological_age,
6.              int *physiological_age) {
7.    static int year = 0;
8.    int experience, things = 0;

9.    psychological_age++; (*physiological_age)++;
10.   year++, things++;
11.   experience = (rand()%200 - 100);
12.   people += experience * year;
13.   printf("year(%p): %d\n", &year, year);
14.   printf("people(%p): %d\n", &people, people);
15.   printf("things(%p): %d\n", &things, things);
16.   printf("psychological_age: %p\n", psychological_age);
17.   printf("physiological_age: %p\n", physiological_age);
18.   if (year < 50) experience +=
19.     go_far(psychological_age, physiological_age);
20.   printf("\n");
21.   return experience;
22. }
```

year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.....
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101

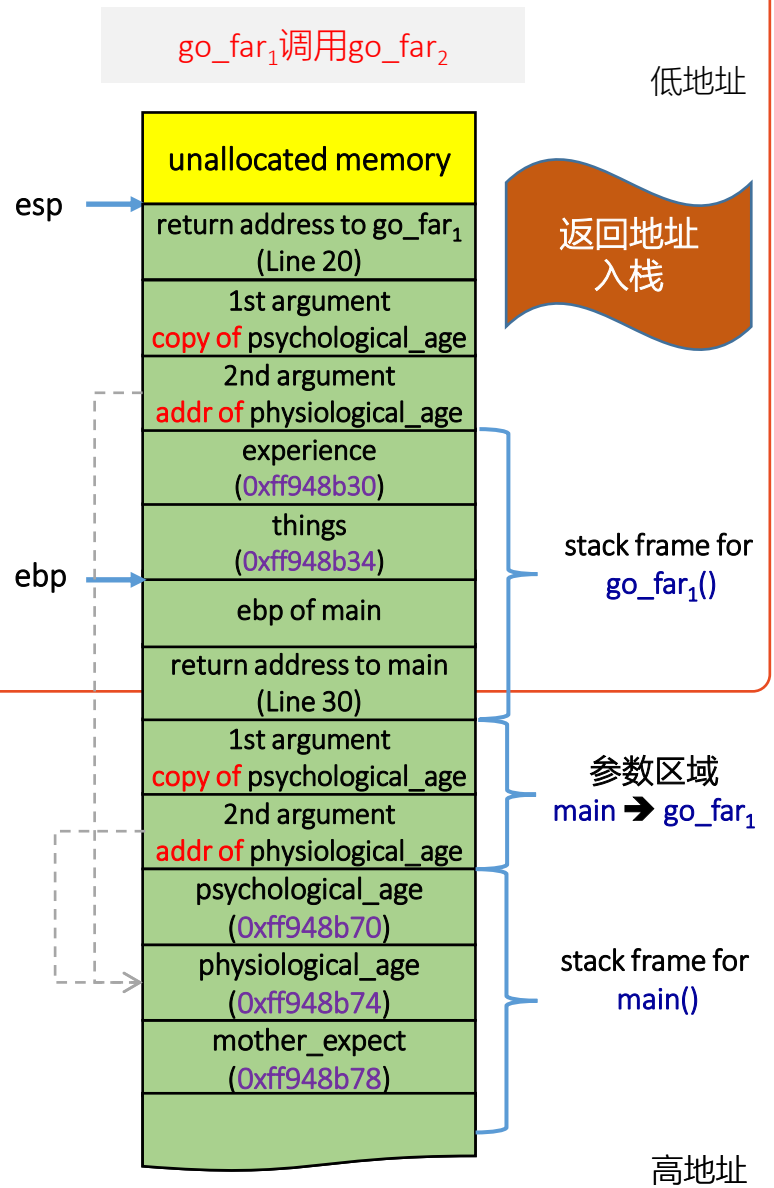
以32位x86体系
UNIX平台为例



```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <time.h>

4.  int people;

5.  int go_far(int psychological_age,
6.              int *physiological_age) {
7.    static int year = 0;
8.    int experience, things = 0;

9.    psychological_age++; (*physiological_age)++;
10.   year++, things++;
11.   experience = (rand()%200 - 100);
12.   people += experience * year;
13.   printf("year(%p): %d\n", &year, year);
14.   printf("people(%p): %d\n", &people, people);
15.   printf("things(%p): %d\n", &things, things);
16.   printf("psychological_age: %p\n", psychological_age);
17.   printf("physiological_age: %p\n", physiological_age);
18.   if (year < 50) experience +=
19.     go_far(psychological_age, physiological_age);
20.   printf("\n");
21.   return experience;
22. }
```

```
year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.....
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101
```

以32位x86体系
UNIX平台为例



```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <time.h>

4.  int people;

5.  int go_far(int psychological_age,
6.              int *physiological_age) {
7.    static int year = 0;
8.    int experience, things = 0;

9.    psychological_age++; (*physiological_age)++;
10.   year++, things++;
11.   experience = (rand()%200 - 100);
12.   people += experience * year;
13.   printf("year(%p): %d\n", &year, year);
14.   printf("people(%p): %d\n", &people, people);
15.   printf("things(%p): %d\n", &things, things);
16.   printf("psychological_age: %p\n", psychological_age);
17.   printf("physiological_age: %p\n", physiological_age);
18.   if (year < 50) experience +=
19.     go_far(psychological_age, physiological_age);
20.   printf("\n");
21.   return experience;
22. }
```

year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

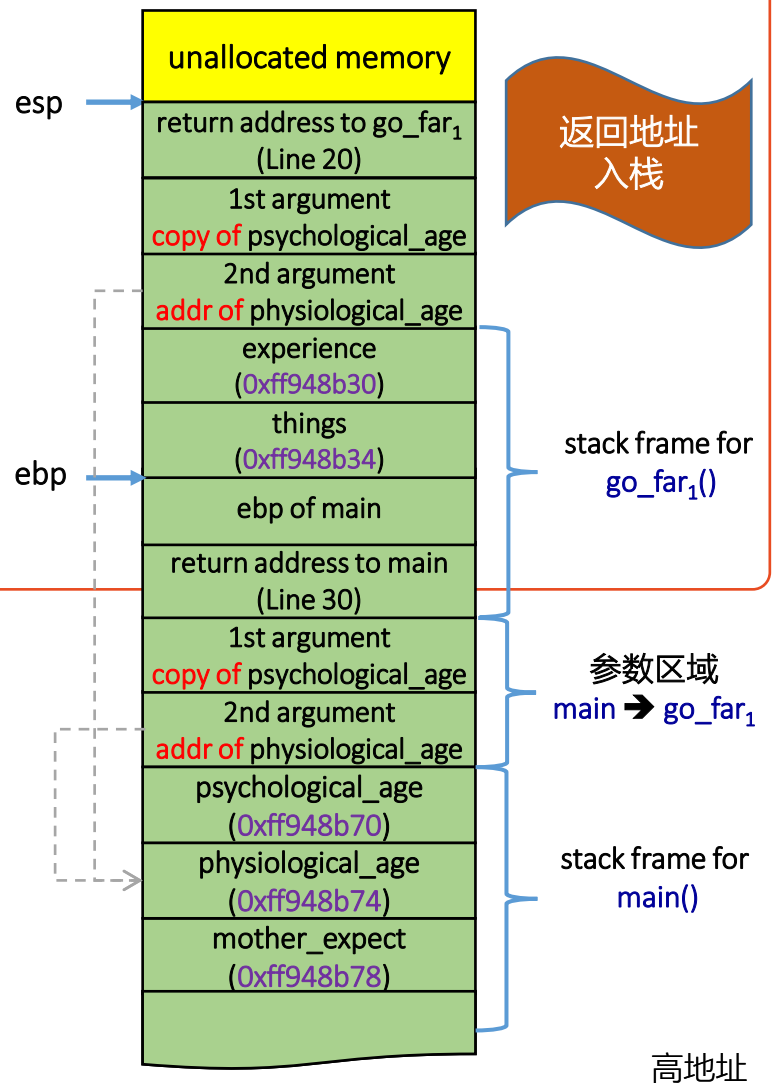
year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.....
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101

以32位x86体系
UNIX平台为例

进入go_far$_2$

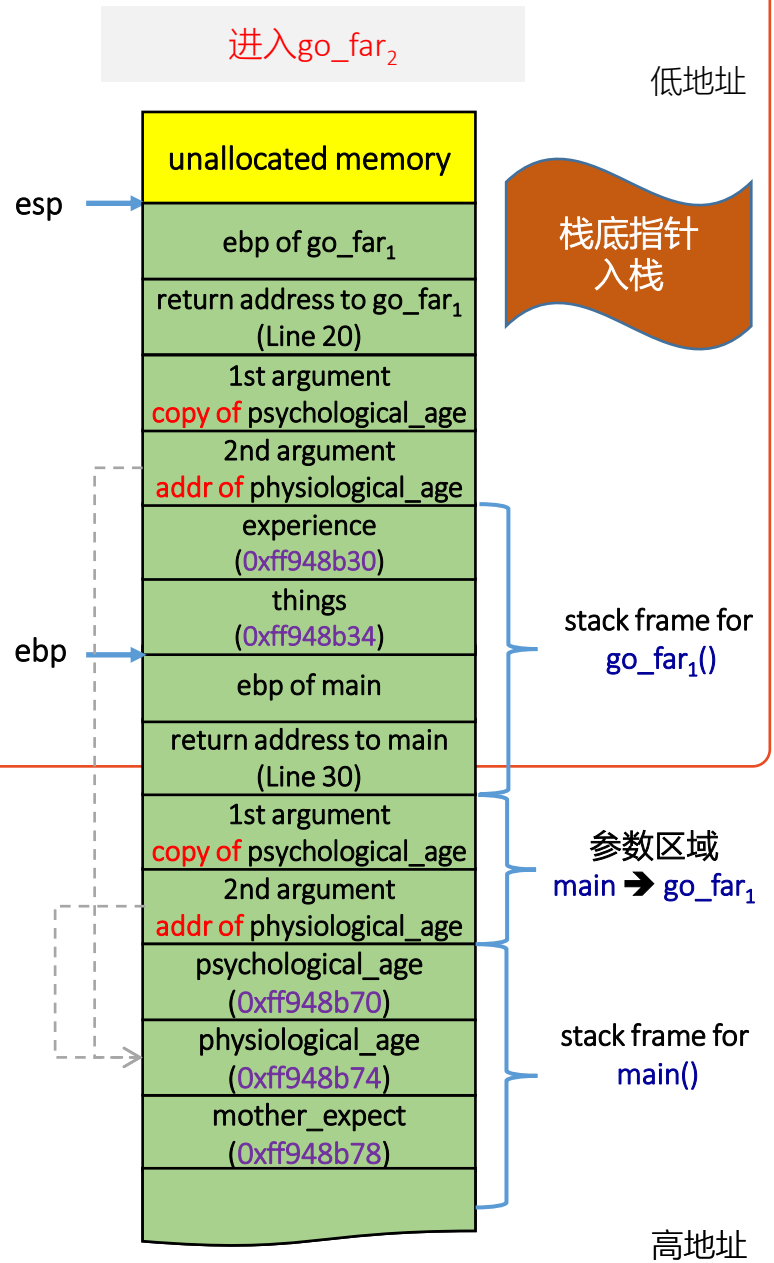
低地址



高地址

```
1.   #include <stdio.h>
2.   #include <stdlib.h>
3.   #include <time.h>

4.   int people;

5.   int go_far(int psychological_age,
6.              int *physiological_age) {
7.     static int year = 0;
8.     int experience, things = 0;

9.     psychological_age++; (*physiological_age)++;
10.    year++, things++;
11.    experience = (rand()%200 - 100);
12.    people += experience * year;
13.    printf("year(%p): %d\n", &year, year);
14.    printf("people(%p): %d\n", &people, people);
15.    printf("things(%p): %d\n", &things, things);
16.    printf("psychological_age: %p\n", psychological_age);
17.    printf("physiological_age: %p\n", physiological_age);
18.    if (year < 50) experience +=
19.      go_far(psychological_age, physiological_age);
20.    printf("\n");
21.    return experience;
22.  }
```

```
year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.....
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101
```



以32位x86体系
UNIX平台为例

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <time.h>

4.  int people;

5.  int go_far(int psychological_age,
6.              int *physiological_age) {
7.    static int year = 0;
8.    int experience, things = 0;

9.    psychological_age++; (*physiological_age)++;
10.   year++, things++;
11.   experience = (rand()%200 - 100);
12.   people += experience * year;
13.   printf("year(%p): %d\n", &year, year);
14.   printf("people(%p): %d\n", &people, people);
15.   printf("things(%p): %d\n", &things, things);
16.   printf("psychological_age: %p\n", psychological_age);
17.   printf("physiological_age: %p\n", physiological_age);
18.   if (year < 50) experience +=
19.     go_far(psychological_age, physiological_age);
20.   printf("\n");
21.   return experience;
22. }
```

year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.....
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

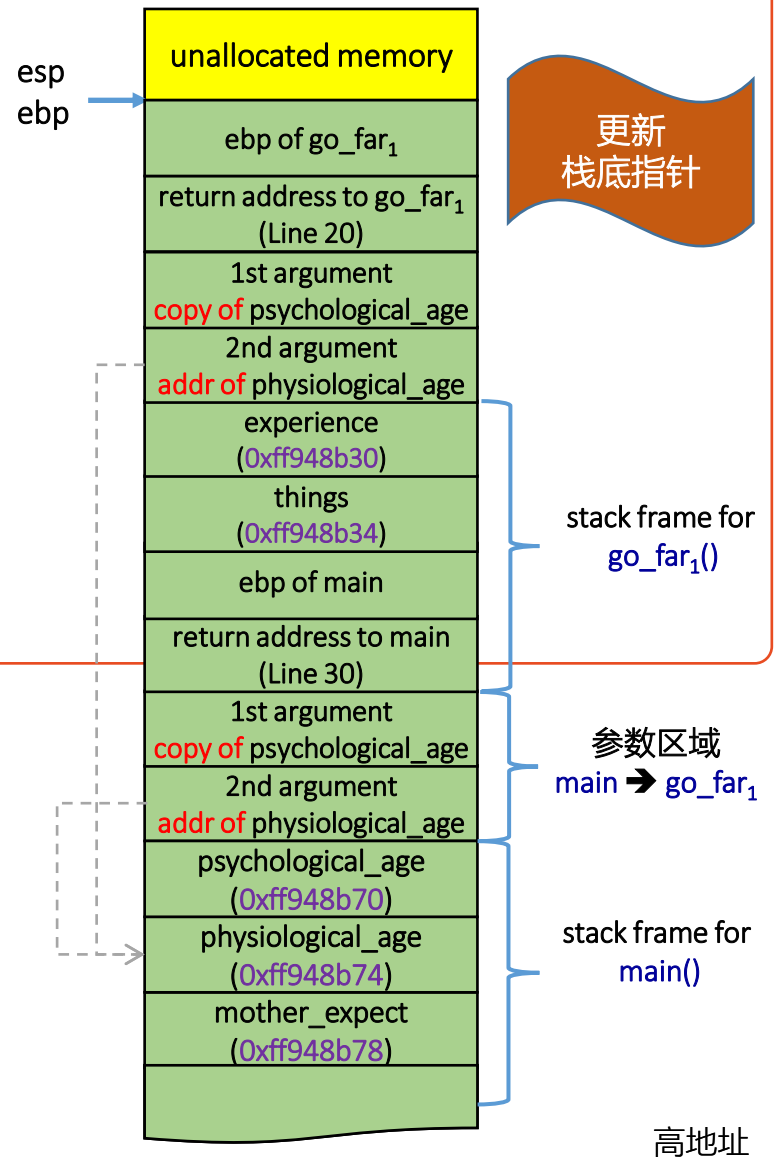
psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101

以32位x86体系
UNIX平台为例

进入go_far$_2$

低地址

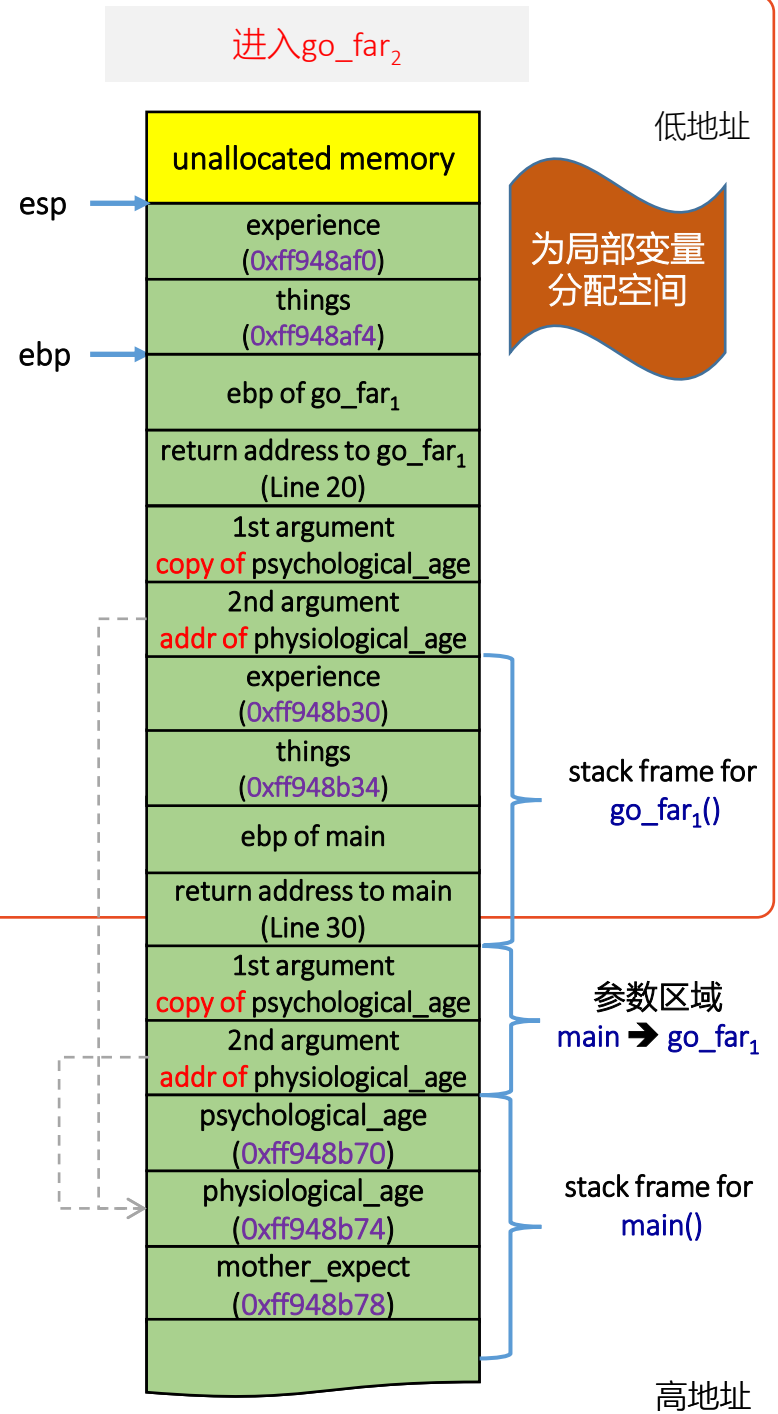
esp
ebp



高地址

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int people;

int go_far(int psychological_age,
           int *physiological_age) {
  static int year = 0;
  int experience, things = 0;

  psychological_age++; (*physiological_age)++;
  year++, things++;
  experience = (rand()%200 - 100);
  people += experience * year;
  printf("year(%p): %d\n", &year, year);
  printf("people(%p): %d\n", &people, people);
  printf("things(%p): %d\n", &things, things);
  printf("psychological_age: %p\n", psychological_age);
  printf("physiological_age: %p\n", physiological_age);
  if (year < 50) experience +=
    go_far(psychological_age, physiological_age);
  printf("\n");
  return experience;
}
```

```
year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.....
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101
```



以32位x86体系
UNIX平台为例

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <time.h>

4.  int people;

5.  int go_far(int psychological_age,
6.              int *physiological_age) {
7.    static int year = 0;
8.    int experience, things = 0;

9.    psychological_age++; (*physiological_age)++;
10.   year++, things++;
11.   experience = (rand()%200 - 100);
12.   people += experience * year;
13.   printf("year(%p): %d\n", &year, year);
14.   printf("people(%p): %d\n", &people, people);
15.   printf("things(%p): %d\n", &things, things);
16.   printf("psychological_age: %p\n", psychological_age);
17.   printf("physiological_age: %p\n", physiological_age);
18.   if (year < 50) experience +=
19.     go_far(psychological_age, physiological_age);
20.   printf("\n");
21.   return experience;
22. }
```

```
year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.....
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101
```

进入go_far$_2$

低地址

esp

ebp

unallocated memory
experience (0xff948af0)
things (0xff948af4)
ebp of go_far$_1$
return address to go_far$_1$ (Line 20)
1st argument copy of psychological_age
2nd argument addr of physiological_age
experience (0xff948b30)
things (0xff948b34)
ebp of main
return address to main (Line 30)
1st argument copy of psychological_age
2nd argument addr of physiological_age
psychological_age (0xff948b70)
physiological_age (0xff948b74)
mother_expect (0xff948b78)

stack frame for go_far$_2$()

参数区域 go_far$_1$ ➔ go_far$_2$

stack frame for go_far$_1$()

参数区域 main ➔ go_far$_1$

stack frame for main()

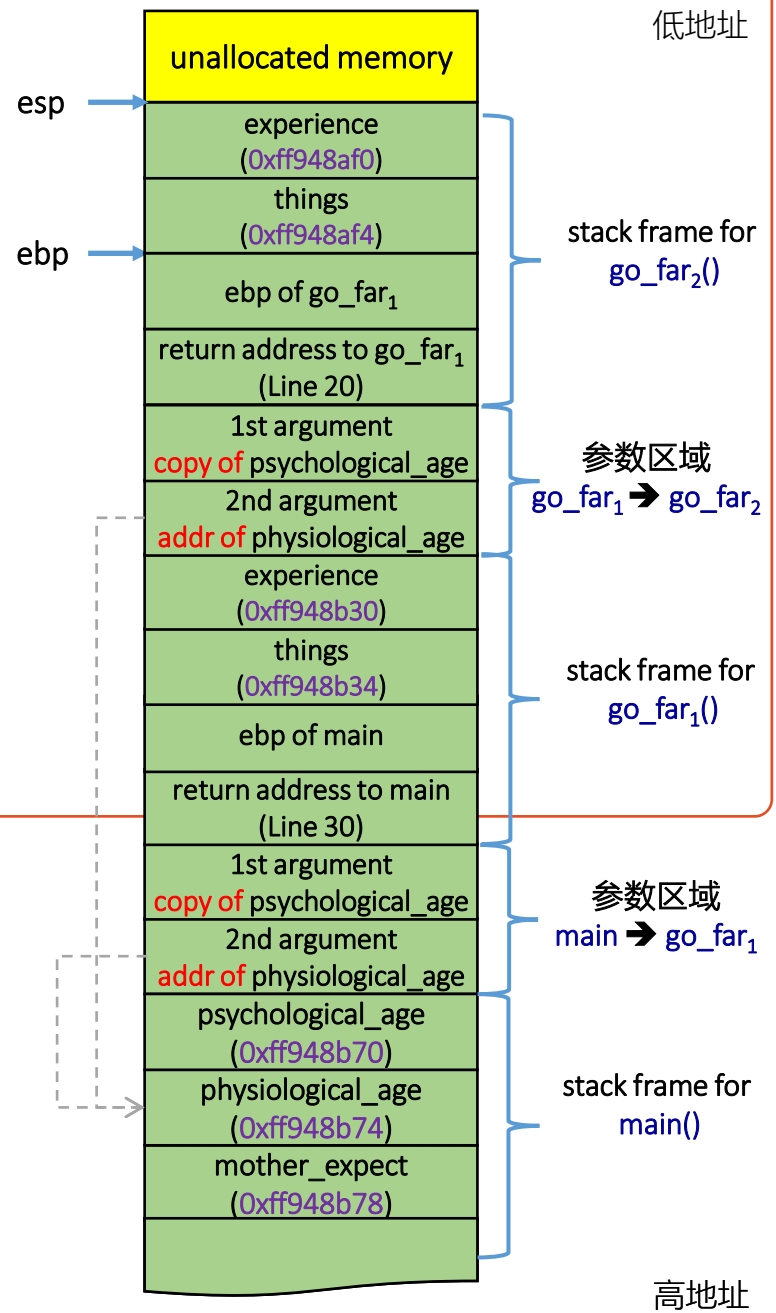
高地址

以32位x86体系
UNIX平台为例

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <time.h>

4.  int people;

5.  int go_far(int psychological_age,
6.              int *physiological_age) {
7.    static int year = 0;
8.    int experience, things = 0;

9.    psychological_age++; (*physiological_age)++;
10.   year++, things++;
11.   experience = (rand()%200 - 100);
12.   people += experience * year;
13.   printf("year(%p): %d\n", &year, year);
14.   printf("people(%p): %d\n", &people, people);
15.   printf("things(%p): %d\n", &things, things);
16.   printf("psychological_age: %p\n", psychological_age);
17.   printf("physiological_age: %p\n", physiological_age);
18.   if (year < 50) experience +=
19.     go_far(psychological_age, physiological_age);
20.   printf("\n");
21.   return experience;
22. }
```

```
year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.....
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101
```

从go_far$_2$返回go_far$_1$



以32位x86体系
UNIX平台为例

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int people;

int go_far(int psychological_age,
           int *physiological_age) {
  static int year = 0;
  int experience, things = 0;

  psychological_age++; (*physiological_age)++;
  year++, things++;
  experience = (rand()%200 - 100);
  people += experience * year;
  printf("year(%p): %d\n", &year, year);
  printf("people(%p): %d\n", &people, people);
  printf("things(%p): %d\n", &things, things);
  printf("psychological_age: %p\n", psychological_age);
  printf("physiological_age: %p\n", physiological_age);
  if (year < 50) experience +=
    go_far(psychological_age, physiological_age);
  printf("\n");
  return experience;
}
```

year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

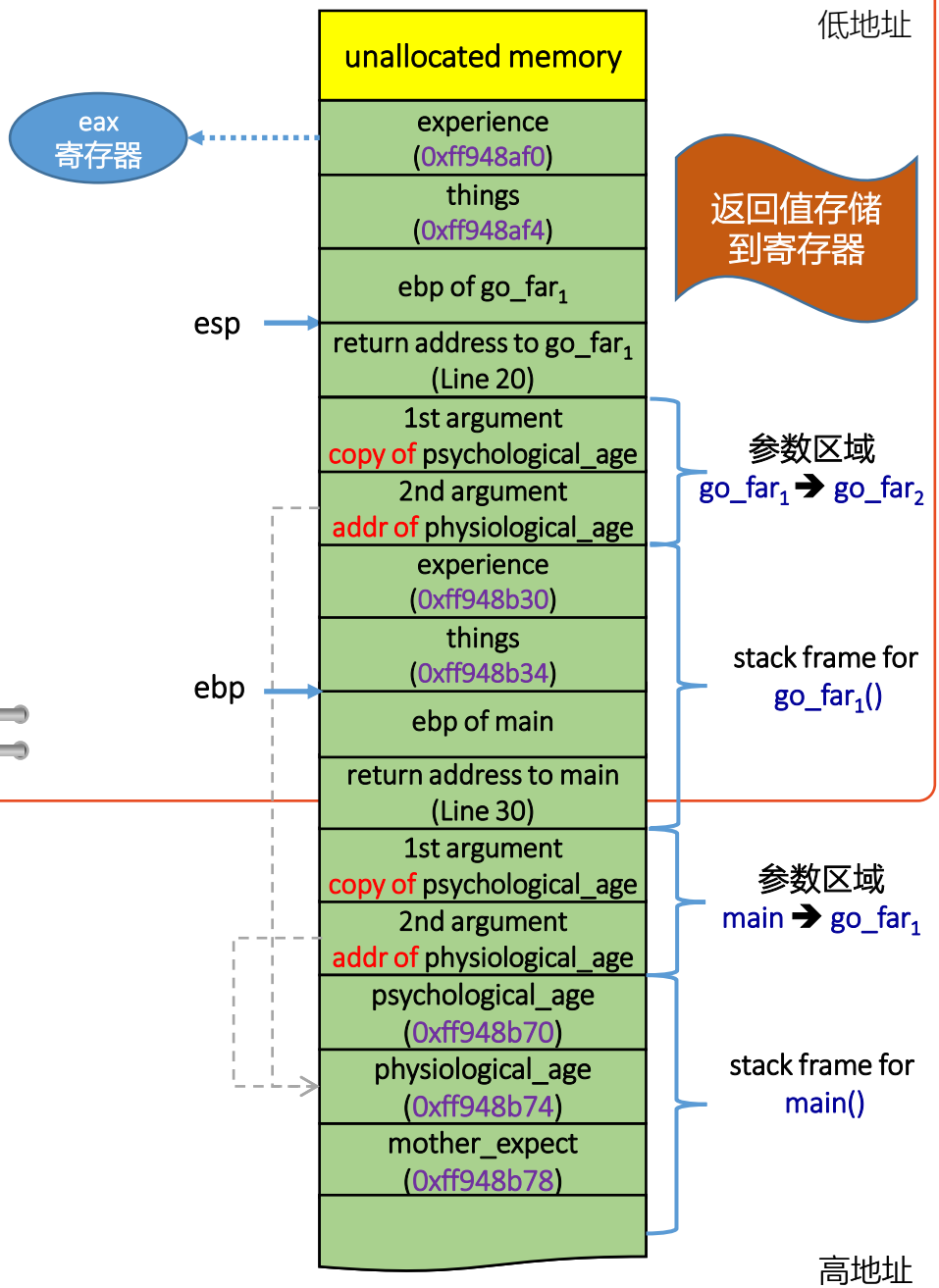
year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.....
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101

以32位x86体系
UNIX平台为例

从go_far$_2$返回go_far$_1$

低地址



高地址

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <time.h>

4.  int people;

5.  int go_far(int psychological_age,
6.              int *physiological_age) {
7.    static int year = 0;
8.    int experience, things = 0;

9.    psychological_age++; (*physiological_age)++;
10.   year++, things++;
11.   experience = (rand()%200 - 100);
12.   people += experience * year;
13.   printf("year(%p): %d\n", &year, year);
14.   printf("people(%p): %d\n", &people, people);
15.   printf("things(%p): %d\n", &things, things);
16.   printf("psychological_age: %p\n", psychological_age);
17.   printf("physiological_age: %p\n", physiological_age);
18.   if (year < 50) experience +=
19.     go_far(psychological_age, physiological_age);
20.   printf("\n");
21.   return experience;
22. }
```

```
year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.....
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101
```

从go_far$_2$返回go_far$_1$

低地址

eax
寄存器

esp
ebp

unallocated memory
experience (0xff948af0)
things (0xff948af4)
ebp of go_far$_1$
return address to go_far$_1$ (Line 20)
1st argument copy of psychological_age
2nd argument addr of physiological_age
experience (0xff948b30)
things (0xff948b34)
ebp of main
return address to main (Line 30)
1st argument copy of psychological_age
2nd argument addr of physiological_age
psychological_age (0xff948b70)
physiological_age (0xff948b74)
mother_expect (0xff948b78)

回收为局部变量分配的空间

参数区域
go_far$_1$ ➔ go_far$_2$

stack frame for go_far$_1$()

参数区域
main ➔ go_far$_1$

stack frame for main()

高地址

以32位x86体系
UNIX平台为例

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int people;

int go_far(int psychological_age,
           int *physiological_age) {
  static int year = 0;
  int experience, things = 0;

  psychological_age++; (*physiological_age)++;
  year++, things++;
  experience = (rand()%200 - 100);
  people += experience * year;
  printf("year(%p): %d\n", &year, year);
  printf("people(%p): %d\n", &people, people);
  printf("things(%p): %d\n", &things, things);
  printf("psychological_age: %p\n", psychological_age);
  printf("physiological_age: %p\n", physiological_age);
  if (year < 50) experience +=
    go_far(psychological_age, physiological_age);
  printf("\n");
  return experience;
}
```

year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

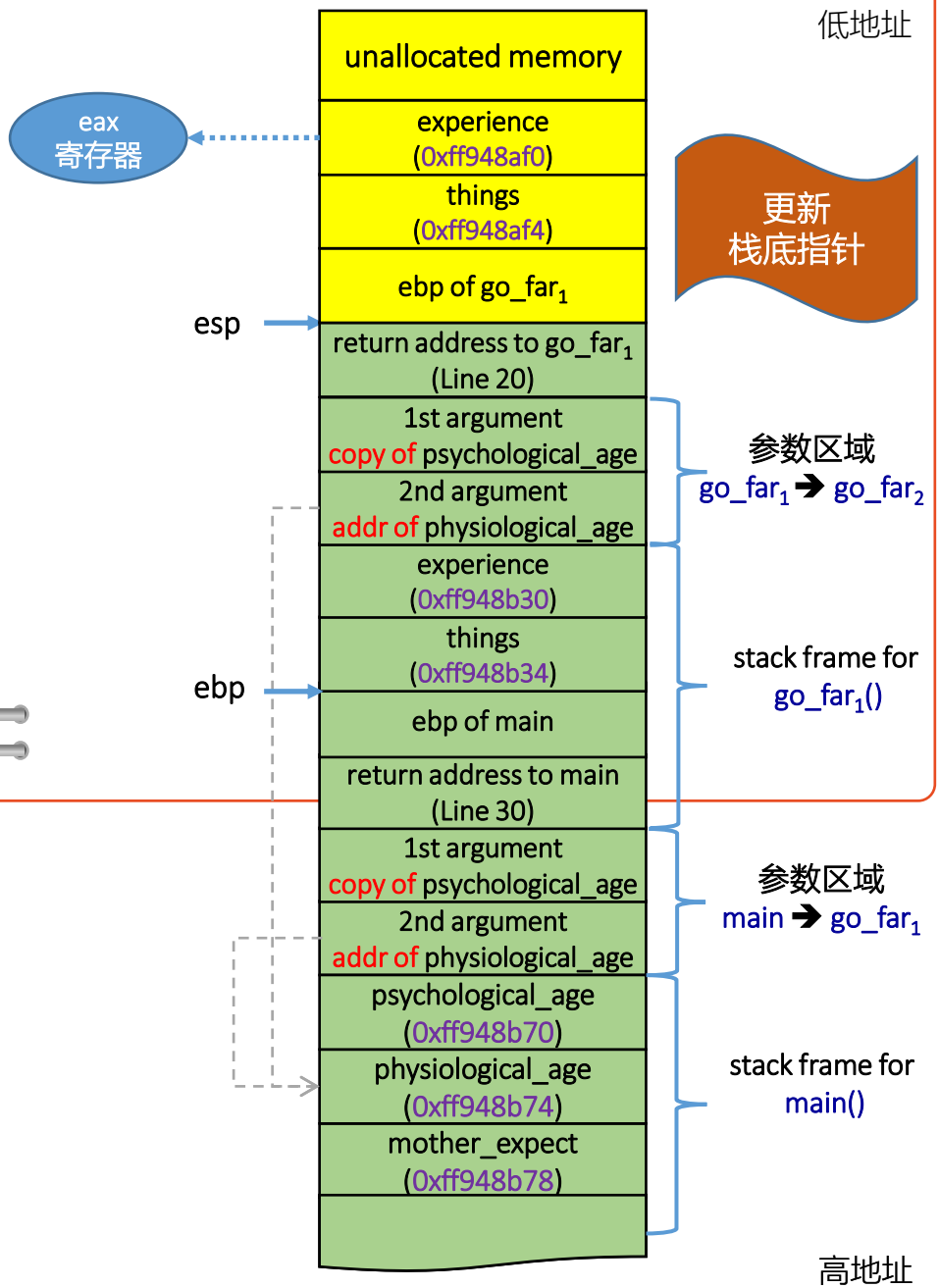
year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.....
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101

以32位x86体系
UNIX平台为例

从go_far$_2$返回go_far$_1$

低地址



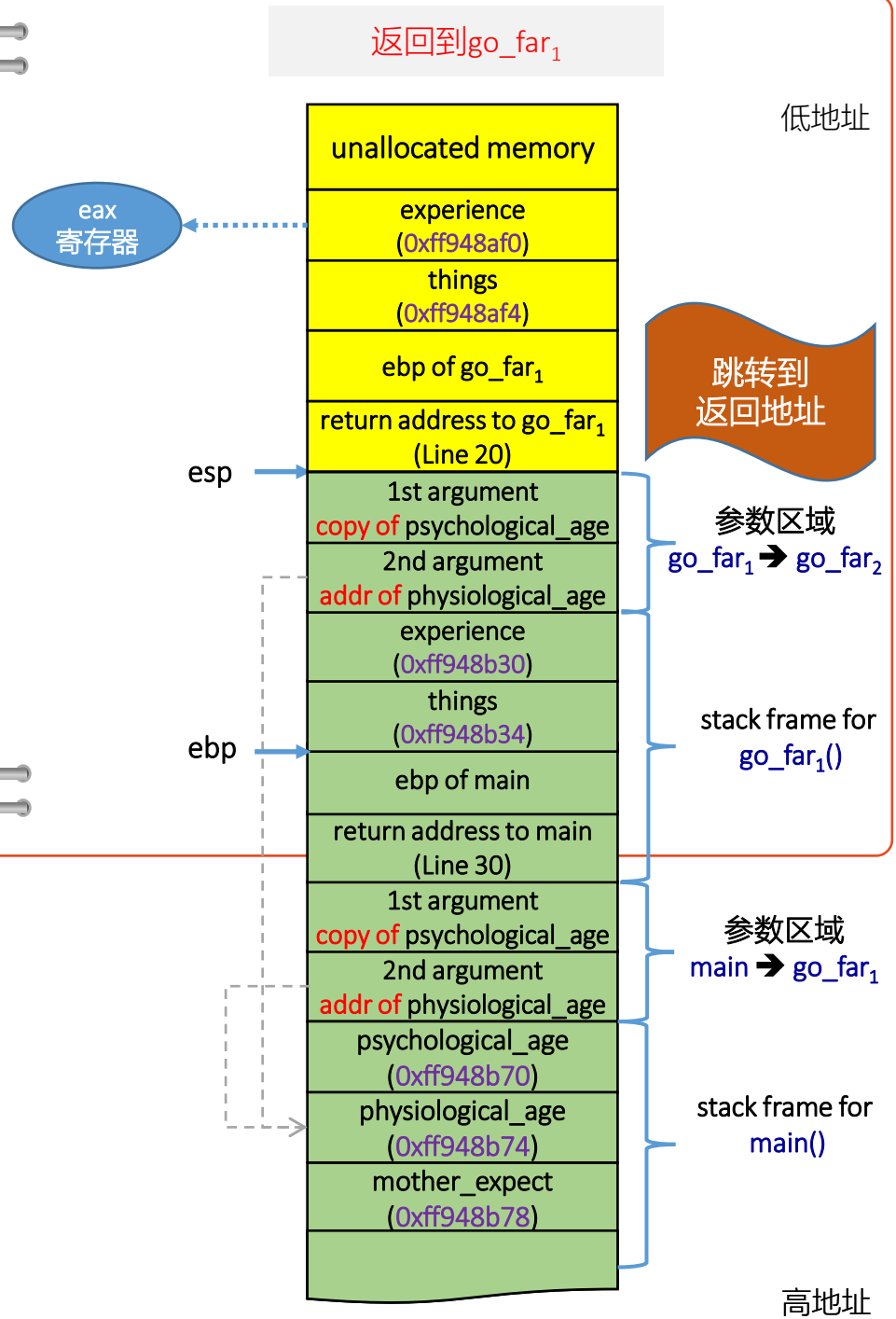
高地址

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <time.h>

4.  int people;

5.  int go_far(int psychological_age,
6.              int *physiological_age) {
7.    static int year = 0;
8.    int experience, things = 0;

9.    psychological_age++; (*physiological_age)++;
10.   year++, things++;
11.   experience = (rand()%200 - 100);
12.   people += experience * year;
13.   printf("year(%p): %d\n", &year, year);
14.   printf("people(%p): %d\n", &people, people);
15.   printf("things(%p): %d\n", &things, things);
16.   printf("psychological_age: %p\n", psychological_age);
17.   printf("physiological_age: %p\n", physiological_age);
18.   if (year < 50) experience +=
19.     go_far(psychological_age, physiological_age);
20.   printf("\n");
21.   return experience;
22. }
```

year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.....
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

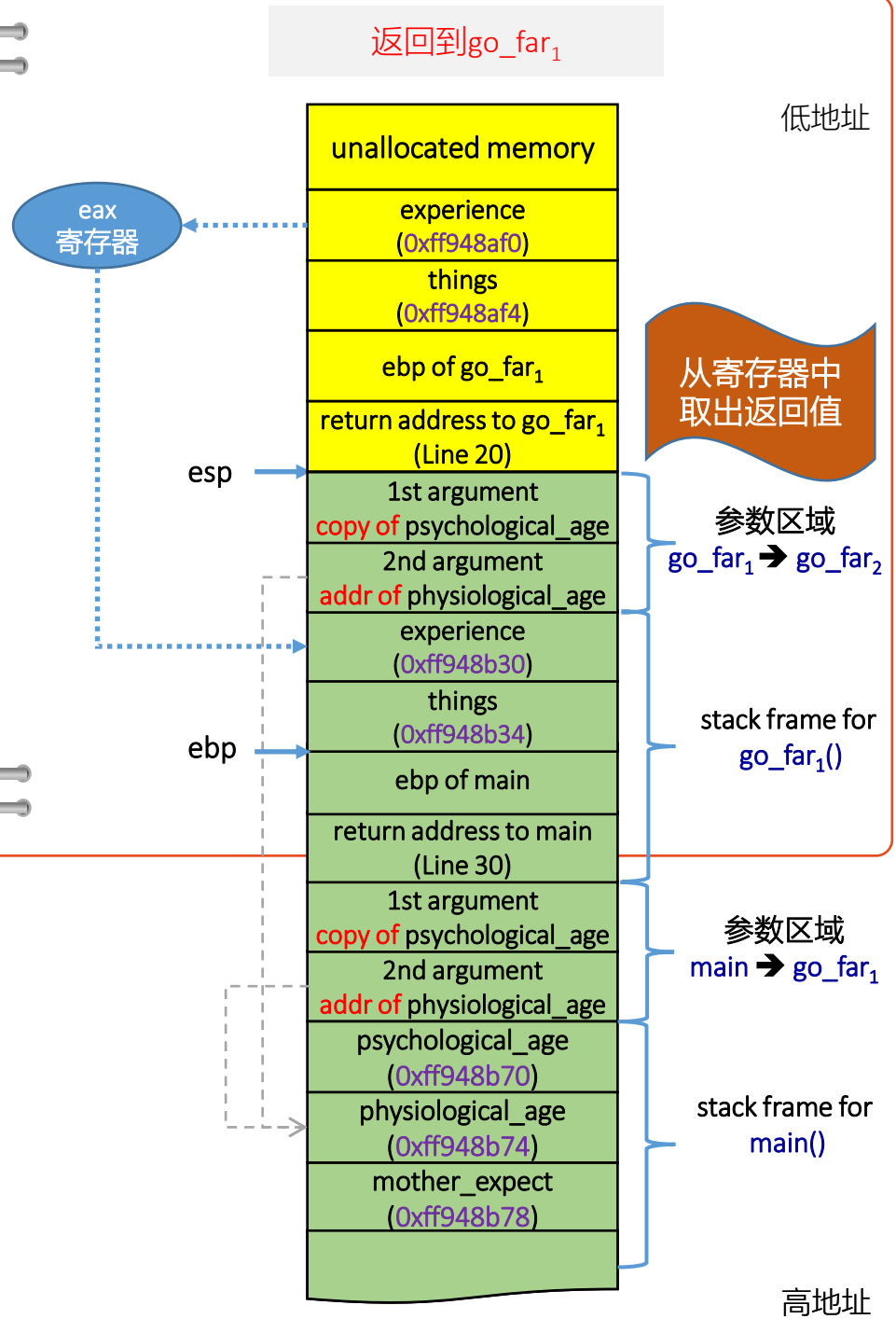
psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101



以32位x86体系
UNIX平台为例

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <time.h>

4.  int people;

5.  int go_far(int psychological_age,
6.              int *physiological_age) {
7.    static int year = 0;
8.    int experience, things = 0;

9.    psychological_age++; (*physiological_age)++;
10.   year++, things++;
11.   experience = (rand()%200 - 100);
12.   people += experience * year;
13.   printf("year(%p): %d\n", &year, year);
14.   printf("people(%p): %d\n", &people, people);
15.   printf("things(%p): %d\n", &things, things);
16.   printf("psychological_age: %p\n", psychological_age);
17.   printf("physiological_age: %p\n", physiological_age);
18.   if (year < 50) experience +=
19.     go_far(psychological_age, physiological_age);
20.   printf("\n");
21.   return experience;
22. }
```

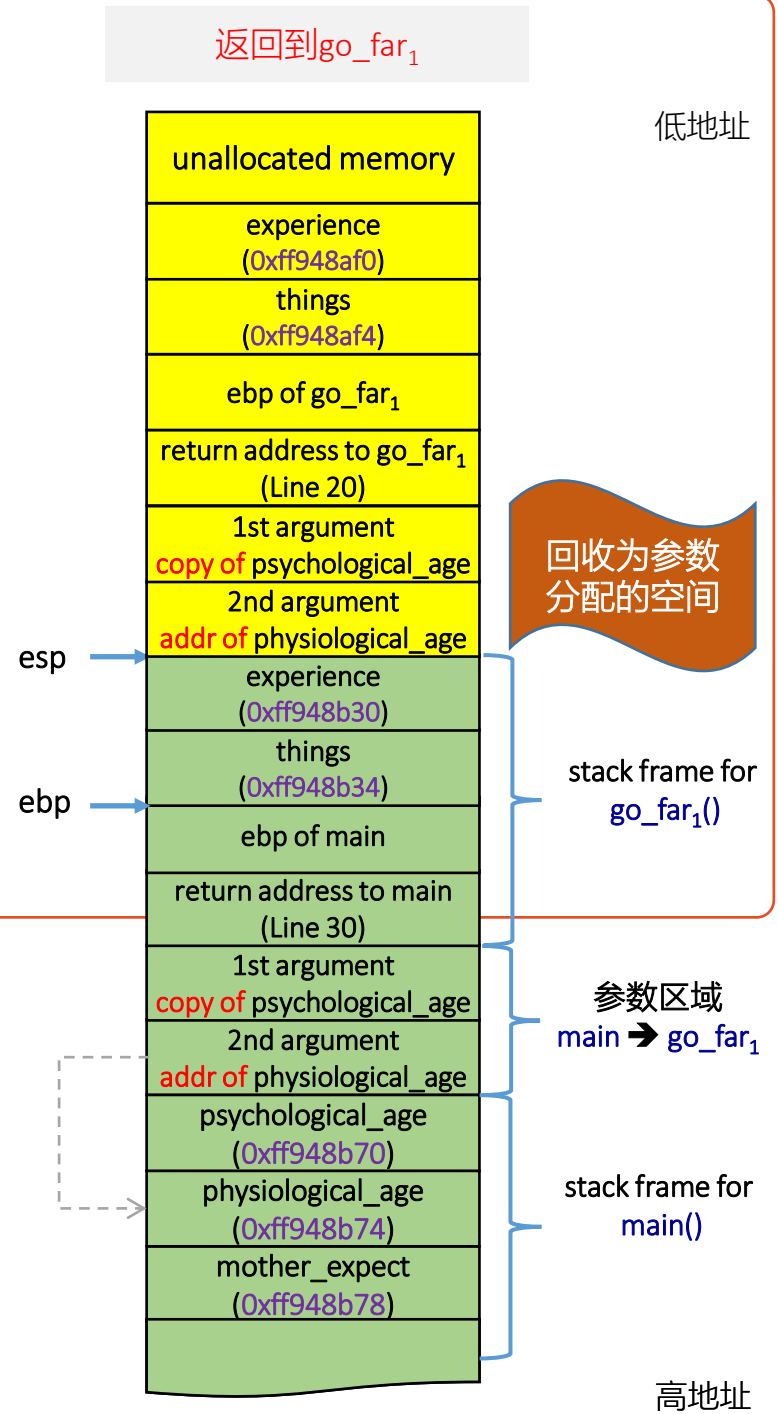
```
year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.....
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101
```



以32位x86体系
UNIX平台为例

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <time.h>

4.  int people;

5.  int go_far(int psychological_age,
6.              int *physiological_age) {
7.    static int year = 0;
8.    int experience, things = 0;

9.    psychological_age++; (*physiological_age)++;
10.   year++, things++;
11.   experience = (rand()%200 - 100);
12.   people += experience * year;
13.   printf("year(%p): %d\n", &year, year);
14.   printf("people(%p): %d\n", &people, people);
15.   printf("things(%p): %d\n", &things, things);
16.   printf("psychological_age: %p\n", psychological_age);
17.   printf("physiological_age: %p\n", physiological_age);
18.   if (year < 50) experience +=
19.     go_far(psychological_age, physiological_age);
20.   printf("\n");
21.   return experience;
22. }
```

```
year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.....
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101
```



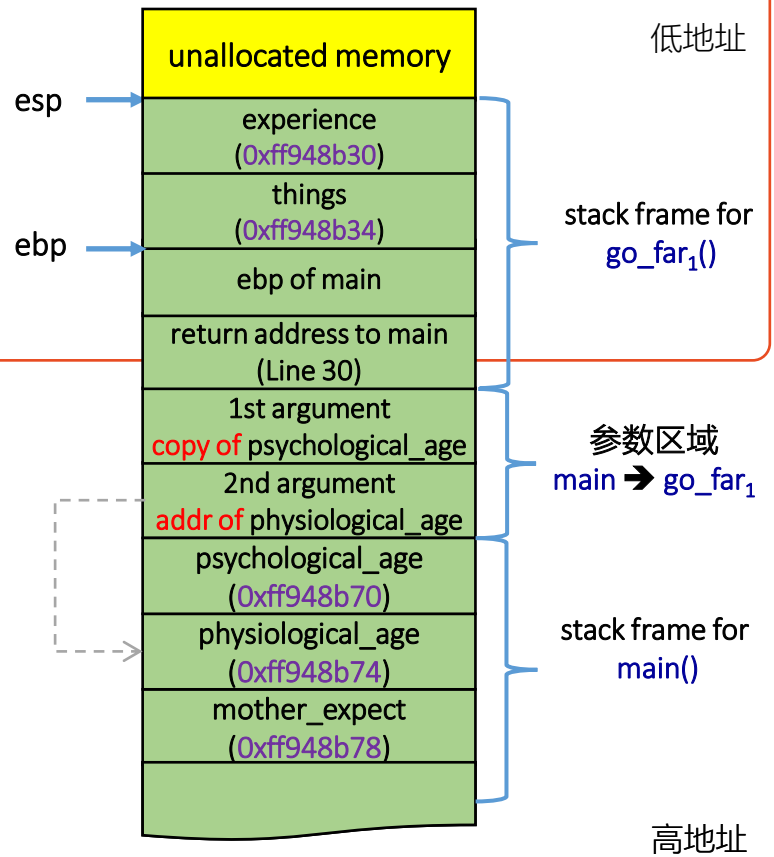
以32位x86体系
UNIX平台为例

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int people;

int go_far(int psychological_age,
           int *physiological_age) {
  static int year = 0;
  int experience, things = 0;

  psychological_age++; (*physiological_age)++;
  year++, things++;
  experience = (rand()%200 - 100);
  people += experience * year;
  printf("year(%p): %d\n", &year, year);
  printf("people(%p): %d\n", &people, people);
  printf("things(%p): %d\n", &things, things);
  printf("psychological_age: %p\n", psychological_age);
  printf("physiological_age: %p\n", physiological_age);
  if (year < 50) experience +=
    go_far(psychological_age, physiological_age);
  printf("\n");
  return experience;
}
```

```c
int main(int argc, char **argv) {
  int psychological_age = 18;
  int physiological_age = 18;
  int mother_expect;
  srand(time(NULL));
  mother_expect = go_far(psychological_age,
                         &physiological_age);
  printf("psychological_age(%p): %d\n",
         &psychological_age, psychological_age);
  printf("physiological_age(%p): %d\n",
         &physiological_age, physiological_age);
  printf("mother_exprect(%p): %d\n",
         &mother_expect, mother_expect);
}
```

```
year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.....
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101
```

从go_far$_1$返回main



以32位x86体系
UNIX平台为例

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <time.h>

4.  int people;

5.  int go_far(int psychological_age,
6.              int *physiological_age) {
7.    static int year = 0;
8.    int experience, things = 0;

9.    psychological_age++; (*physiological_age)++;
10.   year++, things++;
11.   experience = (rand()%200 - 100);
12.   people += experience * year;
13.   printf("year(%p): %d\n", &year, year);
14.   printf("people(%p): %d\n", &people, people);
15.   printf("things(%p): %d\n", &things, things);
16.   printf("psychological_age: %p\n", psychological_age);
17.   printf("physiological_age: %p\n", physiological_age);
18.   if (year < 50) experience +=
19.     go_far(psychological_age, physiological_age);
20.   printf("\n");
21.   return experience;
22. }
```

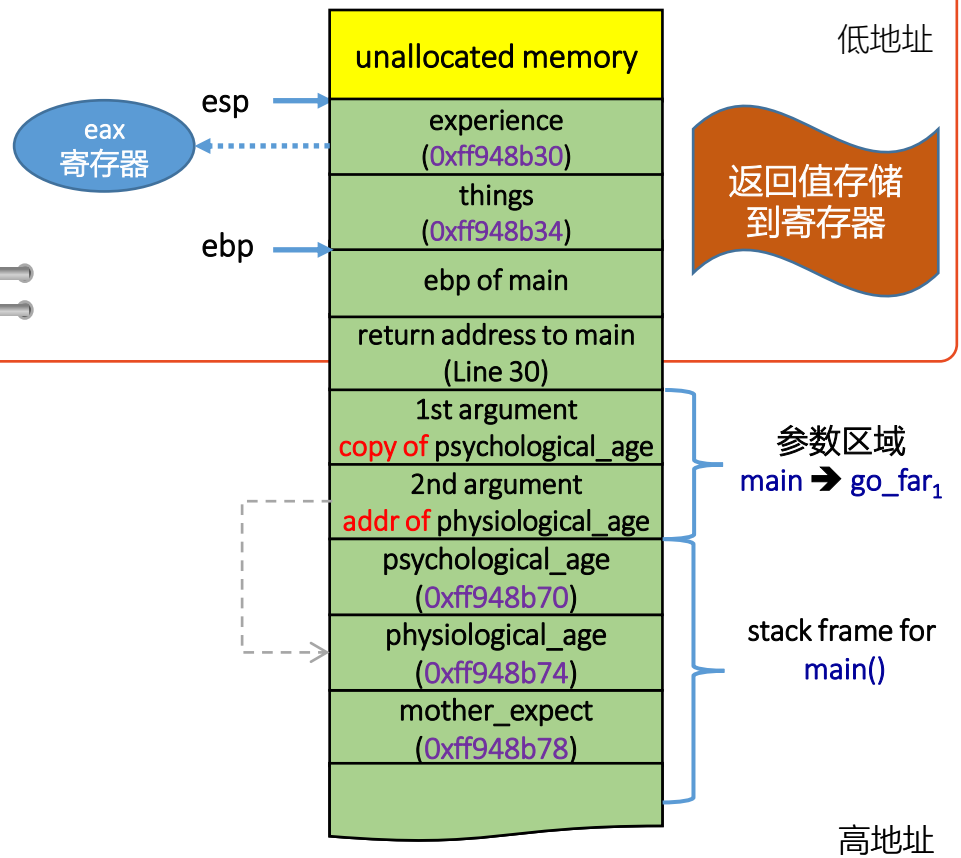
```
23. int main(int argc, char **argv) {
24.   int psychological_age = 18;
25.   int physiological_age = 18;
26.   int mother_expect;
27.   srand(time(NULL));
28.   mother_expect = go_far(psychological_age,
29.                          &physiological_age);
30.   printf("psychological_age(%p): %d\n",
31.          &psychological_age, psychological_age);
32.   printf("physiological_age(%p): %d\n",
33.          &physiological_age, physiological_age);
34.   printf("mother_exprect(%p): %d\n",
35.          &mother_expect, mother_expect);
36. }
```

year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.....
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101

从go_far$_1$返回main



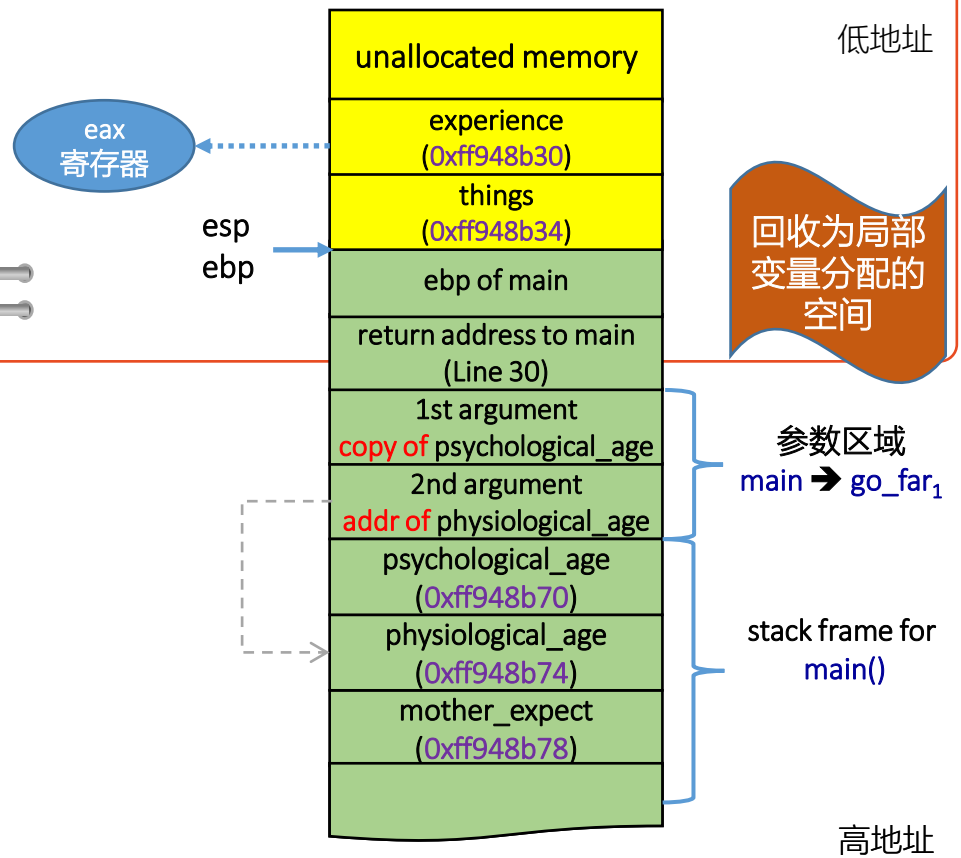
以32位x86体系
UNIX平台为例

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int people;

int go_far(int psychological_age,
           int *physiological_age) {
  static int year = 0;
  int experience, things = 0;

  psychological_age++; (*physiological_age)++;
  year++, things++;
  experience = (rand()%200 - 100);
  people += experience * year;
  printf("year(%p): %d\n", &year, year);
  printf("people(%p): %d\n", &people, people);
  printf("things(%p): %d\n", &things, things);
  printf("psychological_age: %p\n", psychological_age);
  printf("physiological_age: %p\n", physiological_age);
  if (year < 50) experience +=
    go_far(psychological_age, physiological_age);
  printf("\n");
  return experience;
}
```

```c
int main(int argc, char **argv) {
  int psychological_age = 18;
  int physiological_age = 18;
  int mother_expect;
  srand(time(NULL));
  mother_expect = go_far(psychological_age,
                         &physiological_age);
  printf("psychological_age(%p): %d\n",
         &psychological_age, psychological_age);
  printf("physiological_age(%p): %d\n",
         &physiological_age, physiological_age);
  printf("mother_exprect(%p): %d\n",
         &mother_expect, mother_expect);
}
```

```
year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.....
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101
```

从go_far$_1$返回main



以32位x86体系
UNIX平台为例

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <time.h>

4.  int people;

5.  int go_far(int psychological_age,
6.              int *physiological_age) {
7.    static int year = 0;
8.    int experience, things = 0;

9.    psychological_age++; (*physiological_age)++;
10.   year++, things++;
11.   experience = (rand()%200 - 100);
12.   people += experience * year;
13.   printf("year(%p): %d\n", &year, year);
14.   printf("people(%p): %d\n", &people, people);
15.   printf("things(%p): %d\n", &things, things);
16.   printf("psychological_age: %p\n", psychological_age);
17.   printf("physiological_age: %p\n", physiological_age);
18.   if (year < 50) experience +=
19.     go_far(psychological_age, physiological_age);
20.   printf("\n");
21.   return experience;
22. }
```

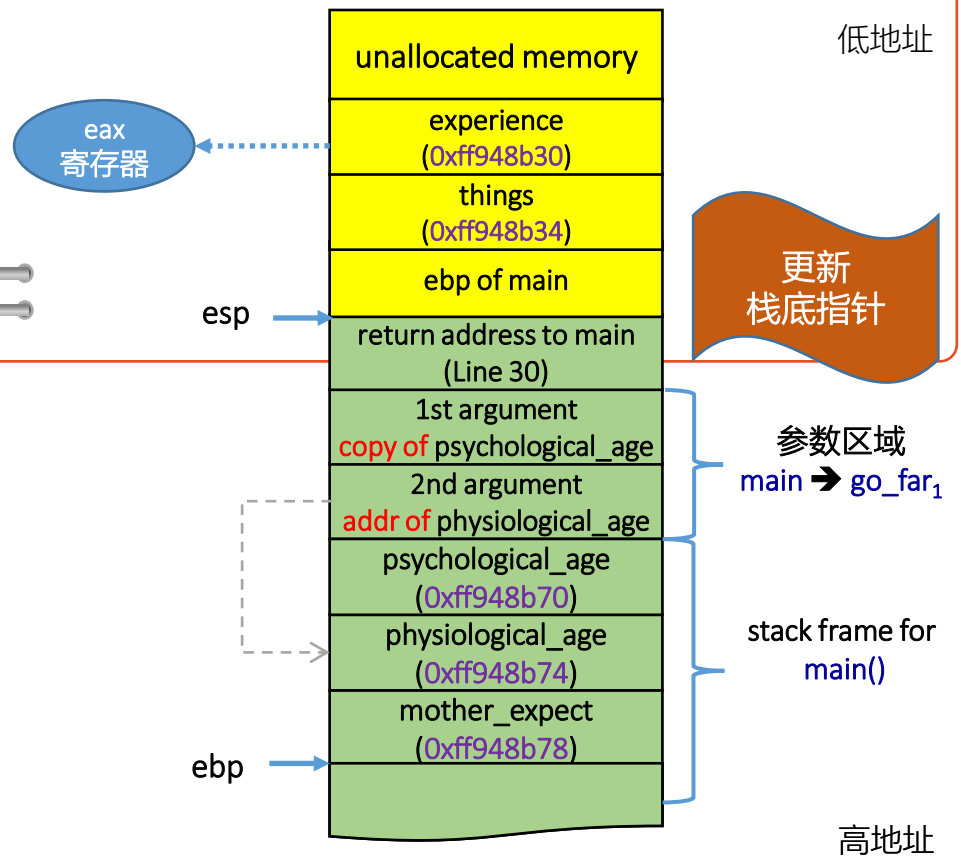
```
23. int main(int argc, char **argv) {
24.   int psychological_age = 18;
25.   int physiological_age = 18;
26.   int mother_expect;
27.   srand(time(NULL));
28.   mother_expect = go_far(psychological_age,
29.                          &physiological_age);
30.   printf("psychological_age(%p): %d\n",
31.          &psychological_age, psychological_age);
32.   printf("physiological_age(%p): %d\n",
33.          &physiological_age, physiological_age);
34.   printf("mother_exprect(%p): %d\n",
35.          &mother_expect, mother_expect);
36. }
```

year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.....
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101

从go_far$_1$返回main



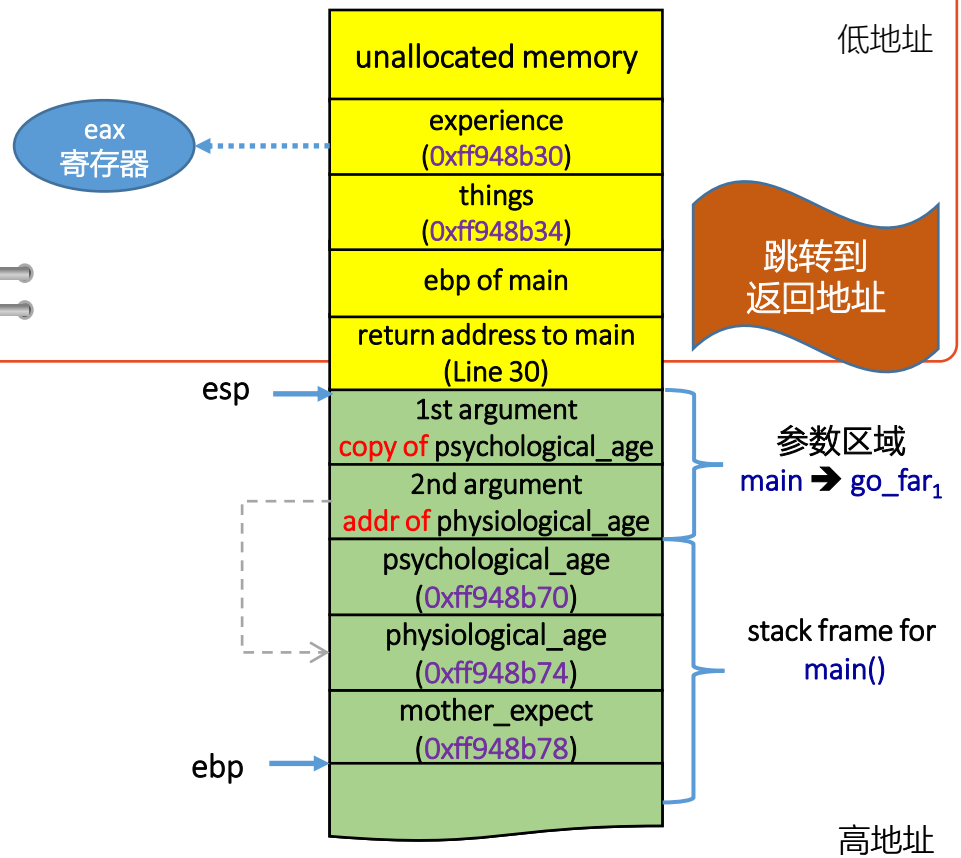
以32位x86体系
UNIX平台为例

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int people;

int go_far(int psychological_age,
           int *physiological_age) {
  static int year = 0;
  int experience, things = 0;

  psychological_age++; (*physiological_age)++;
  year++, things++;
  experience = (rand()%200 - 100);
  people += experience * year;
  printf("year(%p): %d\n", &year, year);
  printf("people(%p): %d\n", &people, people);
  printf("things(%p): %d\n", &things, things);
  printf("psychological_age: %p\n", psychological_age);
  printf("physiological_age: %p\n", physiological_age);
  if (year < 50) experience +=
    go_far(psychological_age, physiological_age);
  printf("\n");
  return experience;
}
```

```c
int main(int argc, char **argv) {
  int psychological_age = 18;
  int physiological_age = 18;
  int mother_expect;
  srand(time(NULL));
  mother_expect = go_far(psychological_age,
                         &physiological_age);
  printf("psychological_age(%p): %d\n",
         &psychological_age, psychological_age);
  printf("physiological_age(%p): %d\n",
         &physiological_age, physiological_age);
  printf("mother_exprect(%p): %d\n",
         &mother_expect, mother_expect);
}
```

返回到main

```
year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.....
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101
```



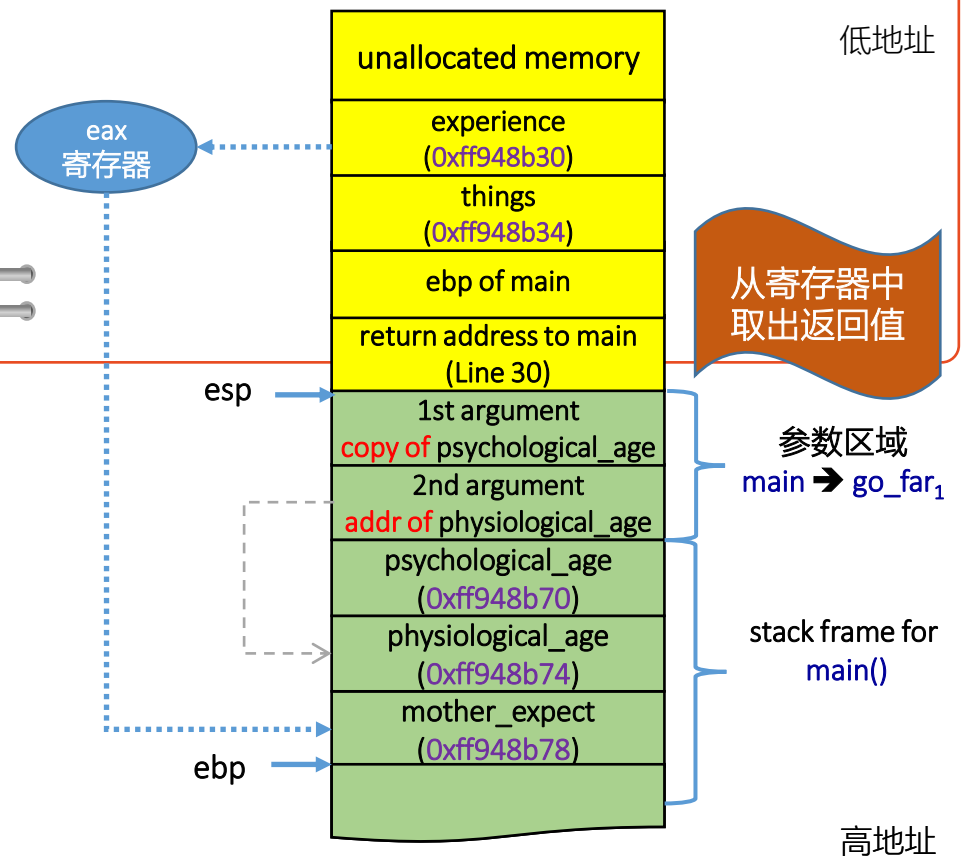
以32位x86体系
UNIX平台为例

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <time.h>

4.  int people;

5.  int go_far(int psychological_age,
6.              int *physiological_age) {
7.    static int year = 0;
8.    int experience, things = 0;

9.    psychological_age++; (*physiological_age)++;
10.   year++, things++;
11.   experience = (rand()%200 - 100);
12.   people += experience * year;
13.   printf("year(%p): %d\n", &year, year);
14.   printf("people(%p): %d\n", &people, people);
15.   printf("things(%p): %d\n", &things, things);
16.   printf("psychological_age: %p\n", psychological_age);
17.   printf("physiological_age: %p\n", physiological_age);
18.   if (year < 50) experience +=
19.     go_far(psychological_age, physiological_age);
20.   printf("\n");
21.   return experience;
22. }
```

```
23. int main(int argc, char **argv) {
24.   int psychological_age = 18;
25.   int physiological_age = 18;
26.   int mother_expect;
27.   srand(time(NULL));
28.   mother_expect = go_far(psychological_age,
29.                          &physiological_age);
30.   printf("psychological_age(%p): %d\n",
31.          &psychological_age, psychological_age);
32.   printf("physiological_age(%p): %d\n",
33.          &physiological_age, physiological_age);
34.   printf("mother_exprect(%p): %d\n",
35.          &mother_expect, mother_expect);
36. }
```

```
year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.....
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101
```

返回到main



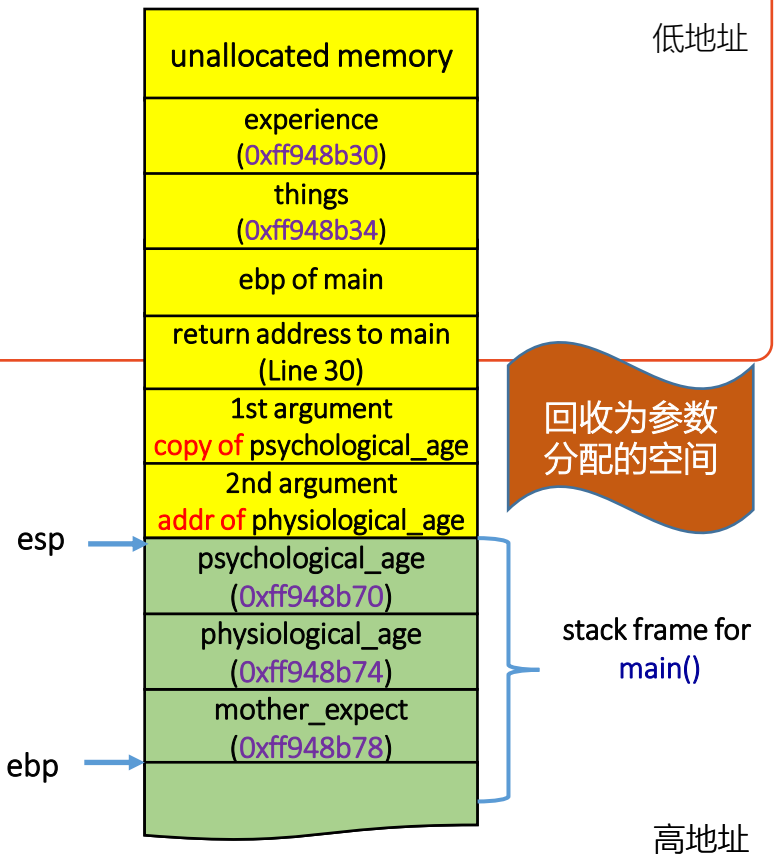
以32位x86体系
UNIX平台为例

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int people;

int go_far(int psychological_age,
           int *physiological_age) {
  static int year = 0;
  int experience, things = 0;

  psychological_age++; (*physiological_age)++;
  year++, things++;
  experience = (rand()%200 - 100);
  people += experience * year;
  printf("year(%p): %d\n", &year, year);
  printf("people(%p): %d\n", &people, people);
  printf("things(%p): %d\n", &things, things);
  printf("psychological_age: %p\n", psychological_age);
  printf("physiological_age: %p\n", physiological_age);
  if (year < 50) experience +=
    go_far(psychological_age, physiological_age);
  printf("\n");
  return experience;
}
```

```c
int main(int argc, char **argv) {
  int psychological_age = 18;
  int physiological_age = 18;
  int mother_expect;
  srand(time(NULL));
  mother_expect = go_far(psychological_age,
                         &physiological_age);
  printf("psychological_age(%p): %d\n",
         &psychological_age, psychological_age);
  printf("physiological_age(%p): %d\n",
         &physiological_age, physiological_age);
  printf("mother_exprect(%p): %d\n",
         &mother_expect, mother_expect);
}
```

```
year(0x5659800c): 1
people(0x56598010): 80
things(0xff948b34): 1
psychological_age: 0xff948b50
physiological_age: 0xff948b74

year(0x5659800c): 2
people(0x56598010): 182
things(0xff948af4): 1
psychological_age: 0xff948b10
physiological_age: 0xff948b74
.....
year(0x5659800c): 50
people(0x56598010): -1462
things(0xff947ef4): 1
psychological_age: 0xff947f10
physiological_age: 0xff948b74

psychological_age(0xff948b70): 18
physiological_age(0xff948b74): 68
mother_expect(0xff948b78): -101
```

返回到main



以32位x86体系
UNIX平台为例

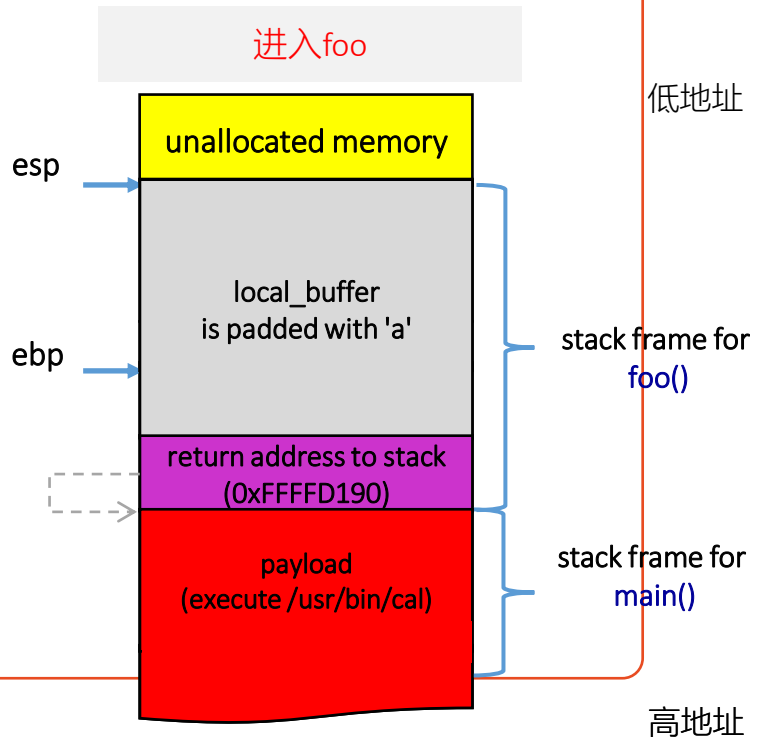
思考：这段程序还有哪里有潜在的安全漏洞？

含有漏洞的待测程序——简易文本编辑器

```
1. #include <stdlib.h>
2. #include <stdio.h>
3. #include <string.h>

4. char global_buffer[64];

5. void foo() {
6.   char local_buffer[16];
7.   strcpy(local_buffer, global_buffer);   //缓冲区溢出漏洞，当global_buffer中'\0'的位置>=16时候触发
8. }

9. int main() {
10.  char filepath[16];

11.  printf("Please input the file path: ");
12.  scanf("%s", filepath);

13.  FILE *fp = fopen(filepath, "r");
14.  fgets(global_buffer, 64, fp);
15.  fclose(fp);

16.  foo();

17.  return 0;
18.}
```

```
          0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000000h: 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 ; aaaaaaaaaaaaaaaa
00000010h: 61 61 61 61 61 61 61 61 61 61 61 61 90 D1 FF FF ; aaaaaaaaaaaa
00000020h: 31 C0 50 68 2F 63 61 6C 68 2F 62 69 6E 68 2F 75 ; 1ÀPh/calh/binh/u
00000030h: 73 72 89 E3 50 53 89 E1 99 B0 0B CD 80 00       ; srãPSá°.Í.
```

特意构造的输入文件shellcode.txt



回顾：冲击波病毒利用微软RPC漏洞

```
1.  error_status_t _RemoteActivation(..., WCHAR *pwszObjectName, ... ) {
2.    *phr = GetServerPath(pwszObjectName, &pwszObjectName);
3.     ...
4.  }

5.  HRESULT GetServerPath(WCHAR *pwszPath, WCHAR **pwszServerPath ){
6.    WCHAR *pwszFinalPath = pwszPath;
7.    WCHAR wszMachineName[MAX_COMPUTERNAME_LENGTH_FQDN+1];
8.    hr = GetMachineName(pwszPath, wszMachineName);
9.    *pwszServerPath = pwszFinalPath;
10.}

11. HRESULT GetMachineName(WCHAR *pwszPath, WCHAR wszMachineName[MAX_COMPUTERNAME_LENGTH_FQDN+1])
12. {
13.   pwszServerName = wszMachineName;
14.   LPWSTR pwszTemp = pwszPath + 2;
15.   while ( *pwszTemp != L'\\' ) *pwszServerName++ = *pwszTemp++;
16.   ...
17. }
```

源数据来自远程网络包

字符串复制操作
（可能会导致缓冲区溢出）