第6章 函数

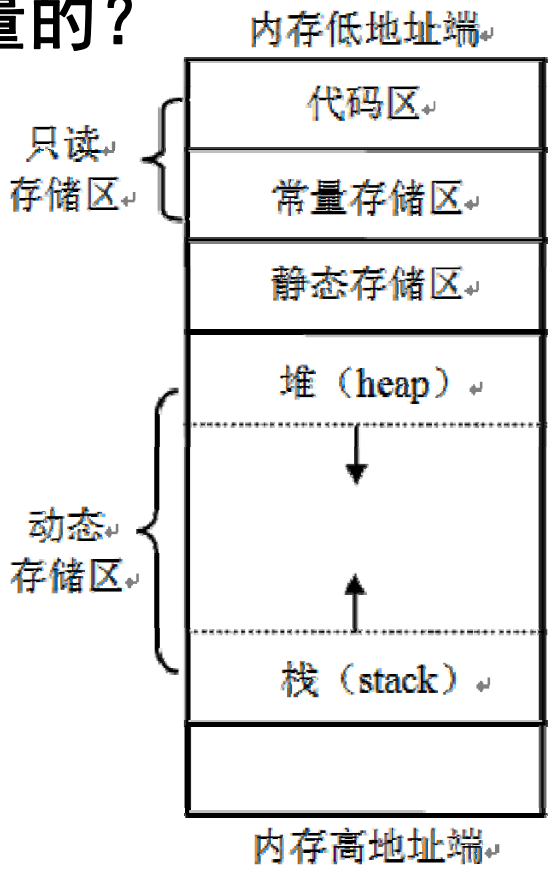
——变量的存储类型

本节要讨论的主要问题

- 编译器是如何给变量分配内存的？
- 变量的存储类型有哪几种？
- 变量的存储类型决定了什么？
- 自动变量和静态局部变量有什么不同？

C程序的内存映像

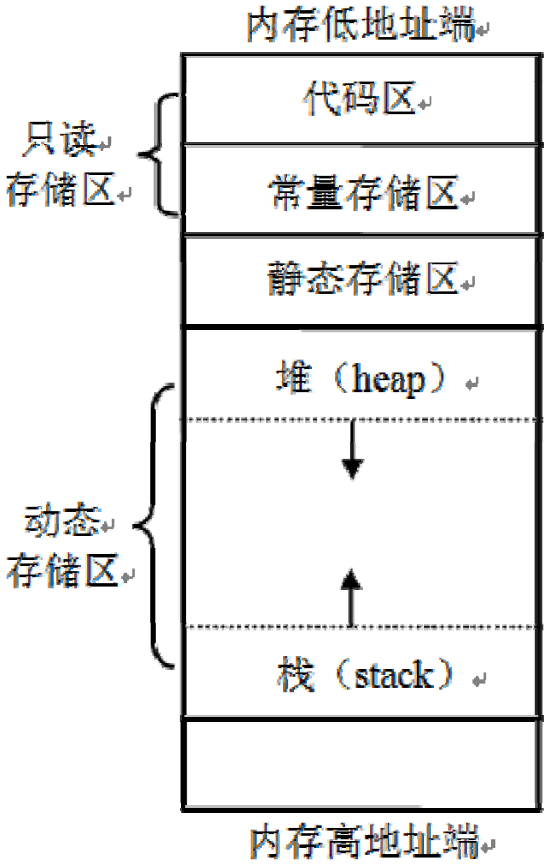
- 问题：编译器是如何区分不同作用域中的同名变量的？
- 只读存储区
  - 存放机器代码和常量等只读数据
- 静态存储区
  - 存放程序中的全局变量和静态变量等
  - 静态——发生在程序编译或链接时
- 动态存储区
  - 包括堆和栈。其中，栈用于保存函数调用时的返回地址、函数的形参、局部变量等信息
  - 动态——发生在程序载入和运行时



何为变量的存储类型

- 变量的存储类型
  - 编译器为变量分配内存的方式
  - 决定了变量的生存期（Lifetime ）
- 在静态存储区中分配内存的变量
  - 生存期是整个程序，全程占据内存
- 在动态存储区中分配内存的变量
  - 生存期是定义它的语句块

静态存储区中的变量：与程序 “共存亡”
动态存储区中的变量：与语句块 “共存亡”



如何声明变量的存储类型

- 声明变量的存储类型

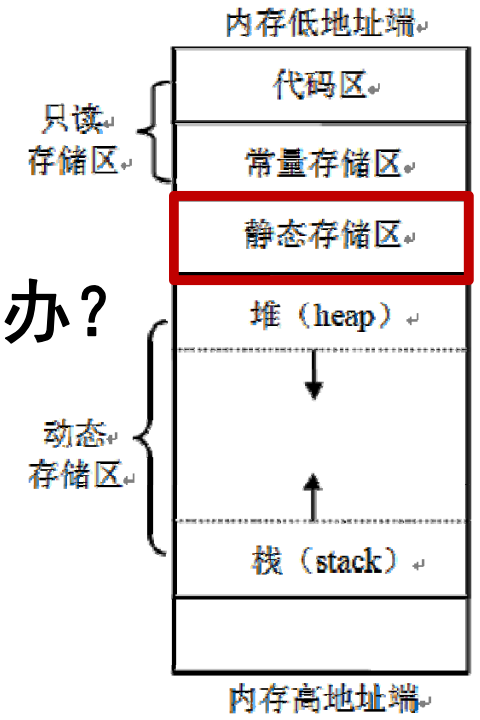
  存储类型　　数据类型　　变量名;

- C存储类型关键字
  - auto（自动变量）
  - static（静态变量）
  - extern（外部变量）
  - register（寄存器变量）

外部变量

- 全局变量
  - 在静态存储区内分配内存
  - 其生存期是整个程序的运行期
  - 没有显式初始化的外部变量由编译程序自动初始化为0
- 问题：若在定义点之前或在其他文件中访问，怎么办？
- 声明变量的存储类型

  extern 数据类型 变量名；
  - 编译器并不对其分配内存，只是表明“我知道了”



自动变量和静态局部变量

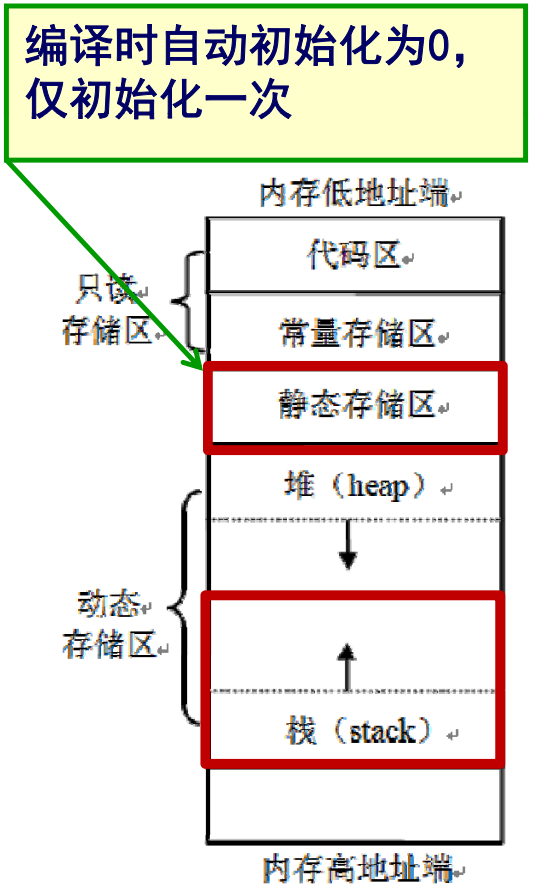
- 自动变量——动态局部变量（缺省类型）

  `auto` 数据类型 变量名;

  - 进入语句块时自动申请内存，退出时自动释放内存
    - 离开函数，值就消失
- 静态变量

  `static` 数据类型 变量名;

  - 从程序运行起占据内存，程序退出时释放内存
    - 离开函数，值仍保留
  - 包括：静态局部变量，静态全局变量
  - 生存期相同，作用域不同（语句块内，文件内）



自动变量和静态局部变量

```
#include <stdio.h>
long Func(int n);
int main()
{
   int i, n;
   printf("Input n:");
   scanf("%d", &n);
   for (i=1; i<=n; i++)
   {
      printf("%d! = %ld\n", i, Func(i));
   }
   return 0;
}
long Func(int n)
{
   static long p = 1;
   p = p * n;
   return p;
}
```

利用静态变量计算n!

```
Input n:10↙
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
```

自动变量和静态局部变量

```
#include <stdio.h>
long Func(int n);
int main()
{
   int i, n;
   printf("Input n:");
   scanf("%d", &n);
   for (i=1; i<=n; i++)
   {
      printf("%d! = %ld\n", i, Func(i));
   }
   return 0;
}
long Func(int n)
{
   auto long p = 1;
   p = p * n;
   return p;
}
```

若静态变量改成自动变量...

```
Input n:10↙
1! = 1
2! = 2
3! = 3
4! = 4
5! = 5
6! = 6
7! = 7
8! = 8
9! = 9
10! = 10
```

寄存器变量

- 寄存器变量的生存期与程序“共存亡”

  `register` 类型名 变量名;

- 适用于使用频率较高的变量，可使程序更小、执行速度更快
  - 现代编译器有能力自动把普通变量优化为寄存器变量，且可以忽略用户的指定
  - 一般无需特别声明变量为`register`

变量的作用域和存储类型

- 全局变量
  - 静态全局变量 （只限本文件使用）
  - 外部变量 （非静态外部变量允许其他文件引用）
- 局部变量
  - 静态局部变量（离开函数，值仍保留）
  - 动态局部变量（自动变量）（离开函数，值就消失）

定义点之前使用，需用extern声明

讨论

- 自动变量和静态局部变量有什么相同之处和不同之处？
- 静态变量和全局变量有什么相同之处和不同之处？
- 静态全局变量和静态局部变量有什么相同之处和不同之处？