第6章 函数

——函数原型

本节要讨论的主要问题

- 函数原型与函数定义有何区别？
- 函数原型的主要作用是什么？

如何合并成一个完整的程序？

- **问题**：在一个函数中调用另一个函数，需要具备哪些条件呢?
- 若函数的定义出现在函数调用之**前**

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

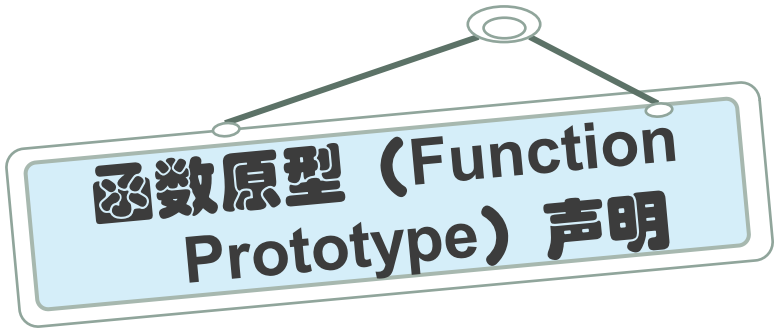
```
int main()
{
    int  a,  b, ave;
    scanf("%d,%d", &a, &b);
    ave = Average(a, b);
    printf("%d\n", ave);
    return 0;
}
```

```
int  Average(int x, int y)
{
    int result;
    result = (x + y) / 2;
    return result;
}
```

如何合并成一个完整的程序？

- 若函数的定义出现在函数调用之后

```
#include <stdio.h>
int  Average(int x, int y);
```




```
int main()
{
    int  a,  b, ave;
    scanf("%d,%d", &a, &b);
    ave = Average(a, b);
    printf("%d\n", ave);
    return 0;
}
```

```
int  Average(int x, int y)
{
    int result;
    result = (x + y) / 2;
    return result;
}
```

函数原型与函数定义的区别

函数定义	函数原型
指函数功能的确立	对函数名、返回值类型、形参类型进行声明
有函数体	不包括函数体
是完整独立的单位	是一条语句，以分号结束，只起声明作用
编译器做实事，分配内存，把函数装入内存	编译器对声明的态度是“我知道了” 不分配内存，只保留一个引用，执行程序链接时，将函数的内存地址链接到那个引用上

如何合并成一个完整的程序？

- 当函数定义出现在函数调用之前
  - C89允许不明确地给出函数原型，编译器自动创建隐含的函数声明
  - 但C99不支持隐含的函数声明

```
#include <stdio.h>
int  Average(int x, int y)
{
     int result;
     result = (x + y) / 2;
     return result;
}
int main()
{
    int  a,  b, ave;
    scanf("%d,%d", &a, &b);
    ave = Average(a, b);
    printf("%d\n", ave);
    return 0;
}
```

如何合并成一个完整的程序？

- **问题：当函数定义出现在函数调用之后时，是否也支持隐含的函数声明呢？**

```
warning: implicit declaration of function 'Average'
```

```c
#include <stdio.h>
int main()
{
    int  a,  b, ave;
    scanf("%d,%d", &a, &b);
    ave = Average(a, b);
    printf("%d\n", ave);
    return 0;
}
int  Average(int x, int y)
{
     int result;
     result = (x + y) / 2;
     return result;
}
```

如何合并成一个完整的程序？

- **问题：**当函数定义出现在函数调用之**后**时，是否也支持隐含的函数声明呢？

```
#include <stdio.h>
int main()
{
    long a,  b, ave;
    scanf("%ld,%ld", &a, &b);
    ave = Average(a, b);
    printf("%ld\n", ave);
    return 0;
}
long Average(long x, long y)
{
    long result;
    result = (x + y) / 2;
    return result;
}
```

```
warning: implicit declaration of function 'Average'
error: conflicting types for 'Average'
```

函数原型的作用

- 告诉编译器被调函数需要接收几个何种类型的参数，并让其进行参数匹配检查
  - 函数原型中的形参及其类型可省略不写
  - 但写上有助于参数类型匹配检查

```
#include <stdio.h>
int  Average(int x, int y);
int main()
{
    int a,  b, ave;
    scanf("%d,%d", &a, &b);
    ave = Average(a, b);
    printf("%d\n", ave);
    return 0;
}
int Average(int x, int y)
{
     int result;
     result = (x + y) / 2;
     return result;
}
```

Software Engineering Observation

The function prototype, function header and function calls should all agree in the number, type, and order of arguments and parameters, and in the type of return value.

如何合并成一个完整的程序？

- **问题：在函数调用时，若实参与形参不匹配，结果会怎样？**
  - 某些编译器会保持沉默，仅当函数原型与函数定义中的形参类型不一致时才给出编译错误
  - 某些编译器可以捕获实参与形参类型不匹配的错误，并发出警告

Software Engineering Observation

The function prototype, function header and function calls should all agree in the number, type, and order of arguments and parameters, and in the type of return value.

```
#include <stdio.h>
int  Average(int x, int y);
int main()
{
    int a,  b, ave;
    scanf("%d,%d", &a, &b);
    ave = Average(a, b);
    printf("%d\n", ave);
    return 0;
}
int Average(int x, int y)
{
     int result;
     result = (x + y) / 2;
     return result;
}
```

如何合并成一个完整的程序？

- **问题：** 把所有函数的定义都放在main函数的前面，是否可以不用函数原型了呢？
  - 其他函数之间也会相互调用
- 良好的编程习惯
  - 在程序开头给出所有的函数原型

```
#include <stdio.h>
int  Fun(int x, int y)
{
     return Average(x, y);
}
int  Average(int x, int y)
{
     return (x + y) / 2;
}
int main()
{
    int  a,  b, ave;
    scanf("%d,%d", &a, &b);
    ave = Average(a, b);
    printf("%d\n", ave);
    return 0;
}
```