Lec 07 Functions in C

函数的定义与作用

C语言中的函数类似于LC-3汇编语言中的子程序，用于执行特定任务。

函数的主要优势包括：

- **隐藏底层细节**：将复杂的实现细节封装在函数中，使用者只需关注函数的功能。
- **提供程序结构**：使代码更清晰，易于阅读和理解。
- **支持独立开发**：在大型项目中，不同开发者可以分别开发不同的函数。
- **代码重用**：函数可以多次调用，减少重复代码。

C函数的结构

```c
#include <stdio.h>
int Fact(int n) {
    int i, result = 1;
    for (i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
int main() {
    int number, answer;
    printf("Enter a number: ");
    scanf("%d", &number);
    answer = Fact(number); // 调用Fact函数，传递参数number，并接收返回值
    printf("factorial of %d is %d\n", number, answer);
    return 0;
}
```

函数的组成部分：

- **函数名**：标识函数的名称，例如`Fact`
- **返回类型**：指定函数返回值的类型，例如`int`或`float`
- **参数类型**：定义函数接受的参数及其类型，例如`int n`

函数原型：函数原型的作用是提前告知编译器函数的属性（返回类型和参数列表），以便在调用时进行类型检查。例如`int Fact(int n)`

参数和返回值

参数：函数可以有**多个参数**或**无参数**。

- 示例：
    - `int func(int a, int b)`：接受两个整数参数，返回它们的和。

- `int func(void)`：无参数，返回固定值0。

返回值：函数最多返回**一个值**，或者不返回任何值（使用void）。

- 示例：
  - `return a + b;`：返回计算结果。
  - 无`return`语句或`return;`：表示无返回值。

局部变量：函数内的变量（如`Fact`中的`i`和`result`）是局部的，仅在函数内部有效，函数调用结束时销毁。

函数的调用

调用方式:

- 函数通过名称和参数列表调用，例如`Fact(number)`
- 参数以**值传递**方式传入，函数内部的操作不会影响外部变量。

作用域与生命周期：局部变量的作用域仅限于函数内部，生命周期从函数调用开始到返回结束。

函数的分离定义

多文件组织：函数可以在多个文件中定义和使用，通过头文件声明。

- 示例：
  - `main.c`：包含主函数，调用其他文件中定义的函数。
  - `reimann_func.c`：定义积分计算相关的函数。
  - `myHeader.h`：声明函数原型。

```
// myHeader.h
float fun(float x);
float reimann_int(int n, float a, float b);

// reimann_func.c
float fun(float x) {
    return x * x + 2 * x + 3;
}
float reimann_int(int n, float a, float b) {
    float dx = (b - a) / n, s = 0.0, x, y;
    for (int i = 0; i < n; i++) {
        x = a + dx * i;
        y = fun(x);
        s += y * dx;
    }
    return s;
}

// main.c
#include <stdio.h>
#include "myHeader.h"
int main() {
```

```c
    float s = reimann_int(100, -1.0, 1.0);
    printf("The integral of f(x) is %f\n", s);
    return 0;
}
```

编译命令：`gcc main.c reman_func.c`

库函数

常用标准库：

- **stdio.h**：提供输入输出函数，如`printf`、`scanf`
- **math.h**：提供数学函数，如`cos`、`sqrt`，编译时需加`-lm`
- **stdlib.h**：提供通用函数，如`rand`
- **time.h**：提供时间函数，如`time(0)`