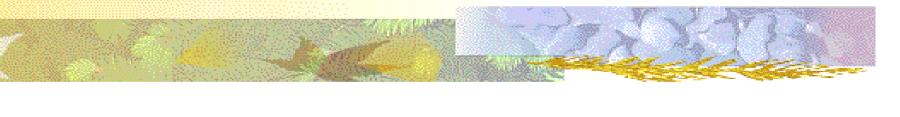


规格严格 功夫到家



第1章 C数据类型

——常量



哈尔滨工业大学 苏小红 sxh@hit.edu.cn

本讲要讨论的主要问题

- 为什么不建议在程序中直接使用常数?
- const常量和宏常量相比,其优势主要体现在哪里?



常量(Constant) ■ 在程序中不 八进制 长整型 无符号整型 十六进制 十进制 ■ 包括: * 整型(如 67, 单精度实 长双精度实型 十进制小 指数形式 • 默认为int * 实型(如 3.14, 1.2e-5, 2.73F, 2.73L) • 默认为double *字符型(如 'z', '3', '\$') * 字符串(如 "UKM", "3", "5a") * 枚举型

计算圆的面积和周长

```
#include <stdio.h>
main()
{
    printf("area = %f\n", 3.14159*5.3*5.3);
    printf("circumference = %f\n", 2*3.14159*5.3);
}
```



计算圆的面积和周长

```
#include <stdio.h>
main()
{
    printf("area = %f\n", 3.14159*5.3*5.3);
    printf("circumference = %f\n", 2*3.14159*5.3);
}
```

- 在程序中直接使用的常数,称为幻数(Magic Number)
- 问题: 使用幻数存在什么问题?
 - * 程序的可读性变差
 - * 容易发生书写错误,产生不一致性
 - * 当常数需要改变时,要修改所有引用它的代码,繁琐,还可能有遗漏



问题: 如何避免在程序中使用幻数?

- 良好的程序设计风格建议把幻数定义为
 - 宏常量
 - const常量
- 优点
 - 减少重复书写常数的工作量
 - 提高程序的可读性和可维护性



宏常量

- 宏常量 (Macro Constant)
 - * 用一个标识符号来表示的常量
- 宏定义

#define 标识符 字符串

编译预处理命令:在源程序编译之前,先对程序中的编译预处理命令进行处理然后将处理的结果和源程序一起进行编译,以得到目标代码

宏常量

■ 宏常量(Macro Constant)

■ 宏定义

#define

#define PI

标识符 字符串

3.14159

不区分数据类型

符号常量(Symbolic Constant)

宏名(Macro Name),一般全大写

计算圆的周长和面积

```
#include <stdio.h>
#define PI 3.14159
#define R 5.3
main()
{
    printf("area = %f\n", PI * R * R);
    printf("circumference = %f\n", 2 * PI * R);
}
```

```
宏替换以后,相当于执行
#include <stdio.h>
main()
{
    printf("area = %f\n", 3.14159 * 5.3 * 5.3 );
    printf("circumference = %f\n", 2 * 3.14159 * 5.3 );
}
```

计算圆的周长和面积

```
#include <stdio.h>
#define PI 3.14159;
#define R 5.3;

main()

{
    printf("area = %f\n", PI * R * R);
    printf("circumference = %f\n", 2 * PI * R);
}
```



```
相当于执行
#include <stdio.h>
main()
{
    printf("area = %f\n", 3.14159; * 5.3; * 5.3;);
    printf("circumference = %f\n", 2 * 3.14159; * 5.3;);
}
```

宏常量

- 问题: 宏常量存在的问题是什么?
 - * 没有数据类型,编译器在宏替换时不进行类型检查
 - * 只进行简单的字符串替换, 极易产生意想不到的错误
- 问题: 能否声明具有某种数据类型的常量呢?

const常量

```
#include <stdio.h>
main()
{

const double pi = 3.14159;
const double r = 5.3;
printf("area = %f\n", pi * r * r);
printf("circumference = %f\n", 2 * pi * r);
}
```

- ■问题: const常量与宏常量相比的优点是什么?
 - -const常量有数据类型,编译器能对其进行类型检查
 - 某些集成化调试工具可以对const常量进行调试

const常量

```
#include <stdio.h>
main()
{
    const float pi = 3.14159;
    const float r = 5.3;
    printf("area = %f\n", pi * r * r);
    printf("circumference = %f\n", 2 * pi * r);
}
```

'initializing': truncation from 'const double' to 'const float'

■ 常量3.14159隐含按double型处理

讨论

```
#include <stdio.h>
main()
{
    const float pi = 3.14159;
    const float r = 5.3;
    printf("area = %f\n", pi * r * r);
    printf("circumference = %f\n", 2 * pi * r);
}
```

'initializing': truncation from 'const double' to 'const float'

■ 如何修改这个程序,可以消除这个警告?



讨论

```
#include <stdio.h>
#define PI 3.14159
#define R 5.3
main()
{
    printf("PI * R * R = %f\n", PI * R * R);
    printf("2 * PI * R = %f\n", 2 * PI * R);
}
```

格式控制字符串中出现的 宏名是否进行宏替换?





