注意春寒

- 乍暖还寒时候，最难将息。三杯两盏淡酒，怎敌他晚来风急？

几种说法

- Compile
  - 编译并生成可运行程序
- Build
  - 生成解决方案
  - 将项目中所有代码进行整合编译
    - 大规模程序可能需要编译几小时到数天
    - 因此引入了Make工具
  - 并生成可运行程序
- Run
  - 运行程序
- Build and Run
  - 编译并执行
- 断点
  - Break point
  - 调试时至该位置会停下来

释疑

- 关于类型的选择
  - 什么是用整型
  - 什么时候用浮点
- 关于类型的运算
  - 哪些地方需要注意
  - DEMO
- 关于控制台
  - 命令行
  - Shell
  - 常见的命令
    - cd
    - cc

取模与取余

- 取模运算（“Modulo Operation”）
- 取余运算（“Remainder Operation”）、
  - 两个概念有重叠的部分但又不完全一致
  - 主要的区别在于对负整数进行除法运算时操作不同
    - 1.求整数商： c = a/b;
    - 2.计算模或者余数： r = a - c*b.
    - 求模运算和求余运算在第一步不同:
      - 取余运算在取c的值时，向0 方向舍入
      - 取模运算在计算c的值时，向负无穷方向舍入
      - C/C++ Java求余，python求模
    - DEMO

SI：Memory address

- In C it permits you to see where and how the value is stored
  - The sizeof() operator (也可以写成sizeof 表达式，但不建议，优先级困扰)
  - Address: & operator
  - DEMO

School of Economics ang Management, Beihang University

Introduction to C Programming

jichang@buaa.edu.cn

Processing and Interactive Input

Objectives

- Assignment
- Mathematical Library Functions
- Interactive Input
- Formatted Output
- Symbolic Constants
- Case Study: Interactive Input
- ***Common Programming and Compiler Errors***
  - ***P117页 3.7***

Assignment

- The general syntax for an assignment statement is

```
variable = operand;
```

  - The operand to the right of the assignment operator (=) can be a constant, a variable, or an expression
- The equal sign in C does not have the same meaning as an equal sign in algebra
  - `length=25;` is read “length is assigned the value 25”
- Subsequent assignment statements can be used to change the value assigned to a variable

```
length = 3.7;
length = 6.28;
```

Assignment

- The operand to the right of the equal sign in an assignment statement can be a variable or any valid C expression

```
sum = 3 + 7;
product = .05 * 14.6;
```

  - The value of the expression to the right of = is computed first and then the calculated value is stored in the variable to the left of =
  - lvalue and rvalue
- Variables used in the expression to the right of the = must be initialized if the result is to make sense
  - 否则产生不可预期的错误！
- `amount + 1892 = 1000 + 10 * 5` is invalid!
- 赋值运算的“副作用”（side effect)
  - Demo

Assignment

- = has the lowest precedence of all the binary and unary arithmetic operators
- Multiple assignments are possible in the same statement

```
a = b = c = 25;
```

  - All = operators have the same precedence
  - Operator has right-to-left associativity

```
c = 25;
b = c;
a = b;
```

Implicit Type Conversions

- Data type conversions take place across assignment operators

```
double result;
result = 4; //integer 4 is converted to 4.0
```

- The automatic conversion across an assignment operator is called an **implicit type conversion**

```
int answer;
answer = 2.764; //2.764 is converted to 2
```

  - Here the implicit conversion is from a higher precision to a lower precision data type; the compiler will issue a warning
    - 务必注意此类警告
  - **However, this does not hold when you try to print an integer with the printf function.**

补充：隐式类型转换发生的时机

- 算术表达式或逻辑表达式中操作数的类型不相同
  - 提升
  - 注意无符号与有符号，会转成无符号
- 赋值运算符右值与左值的类型不相同
  - 转左值
- 函数调用时实参与形参的类型不相同
  - 转形参
- return语句中表达式的类型与函数返回类型不相同
  - 转返回类型

Explicit Type Conversion

- 显式类型转换
  - *(datatype) expression*
- 强制类型转换
  - int a；
  - float b=10.0
  - b=(float)a;
  - a=(int)b;

Assignment Variations

- Assignment expressions like `sum = sum + 25` can be written using the following operators:
  - `+=` `-=` `*=` `/=` `%=`
- `sum = sum + 10` can be written as `sum += 10`
- `price *= rate` is equivalent to `price = price * rate`
- `price *= rate + 1` is equivalent to `price = price * (rate + 1)`
  - 不是好的风格！！！
- **This requires initial value!**

Counting

- A **counting statement** is very similar to the accumulating statement

  ```
  variable = variable + fixedNumber;
  ```

- Examples: `i = i + 1;` and `m = m + 2;`
- **Increment operator** (++): `variable = variable + 1` can be replaced by `variable++` or `++variable`
- `This is also allowed by` **`float`** `and` **`double`**

Counting

- When the ++ operator appears before a variable, it is called a **prefix increment operator**; when it appears after a variable, it is called **postfix increment operator**
  - `k = ++n;` is equivalent to
    - `n = n + 1; // increment n first`
    - `k = n; // assign n's value to k`
  - `k = n++;` is equivalent to
    - `k = n; // assign n's value to k`
    - `n = n + 1; // and then increment n`
  - `count1 = 1+ ++count;`（不建议如是写）
  - `count1 = 1+count++;`（不建议如是写）
  - **DEMO**

补充：++i和i++的比较

- ++i 操作除 i 之外不涉及新的（隐含的）操作数，而 i++ 则在 i 之外还涉及另一个新的（隐含的）操作数
  - 1、优先使用 ++i：
    - (a)单独使用时：++i
    - (b)作为循环控制变量使用时：for(int i=0; i!=100; ++i)
  - 2、杜绝两个以上的 i++ 或者 ++i 进行合成
  - 3、++i 比 i++要快?
    - DEMO
    - 如何计时？
    - 对于现代编译器来讲，不会使编译后的程序更小更快
    - 编译器的优化（简介）

Counting

- **Prefix decrement operator**
  - the expression `k = --n` first decrements the value of `n` by 1 before assigning the value of `n` to `k`
- **Postfix decrement operator**
  - the expression `k = n--` first assigns the current value of `n` to `n` and then reduces the value of `n` by 1

Interactive Input

- `scanf()` is used to enter data into a program while it is executing; the value is stored in a variable
  - It requires a control string as the first argument inside the function name parentheses
  - Why a control string is necessary?
- The control string passed to `scanf()` typically consists of conversion control sequences only
  - `scanf()` requires that a list of variable addresses follow the control string
  - `scanf("%d", &num1);`

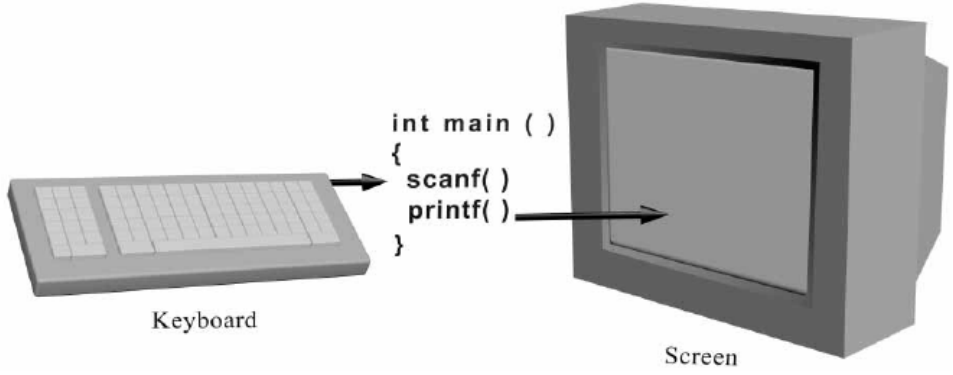

Figure 3.5 `scanf()` used to enter data; `printf()` used to display data

Interactive Input (continued)

- `scanf()` can be used to enter many values
  - `scanf("%f %f",&num1,&num2); //"%f%f"` is the same
  - `scanf("%d%d%f%f",&num1,&num2,&num3,&num4);//1-20.3-4e3`
  - 在数字之间插入多个空格对scanf没有影响
  - 在数字之间插入多个回车、制表符对scanf没有影响
- **A space can affect what the value being entered is when `scanf()` is expecting a character data type**
  - `scanf("%c%c%c",&ch1,&ch2,&ch3);` stores the next three characters typed in the variables `ch1`, `ch2`, and `ch3`; if you type `x y z`, then `x` is stored in `ch1`, a blank is stored in `ch2`, and `y` is stored in `ch3`
  - `scanf("%c %c %c",&ch1,&ch2,&ch3);` causes `scanf()` to look for three characters, each character separated by exactly one space
  - DEMO

Interactive Input (continued)

- In printing a double-precision number using `printf()`, the conversion control sequence for a single-precision variable, `%f`, can be used
- When using `scanf()`, if a double-precision number is to be entered, you must use the `%lf` conversion control sequence
- `scanf()` does not test the data type of the values being entered
- In `scanf("%d %f", &num1, &num2)`, if user enters `22.87`, `22` is stored in `num1` and `.87` in `num2`

补充：“幻影”字符

- CPU-寄存器-cache-内存-硬盘
  - 目前的瓶颈仍在硬盘
  - 食堂的例子
  - 海底捞的瓜子、棋盘和美甲

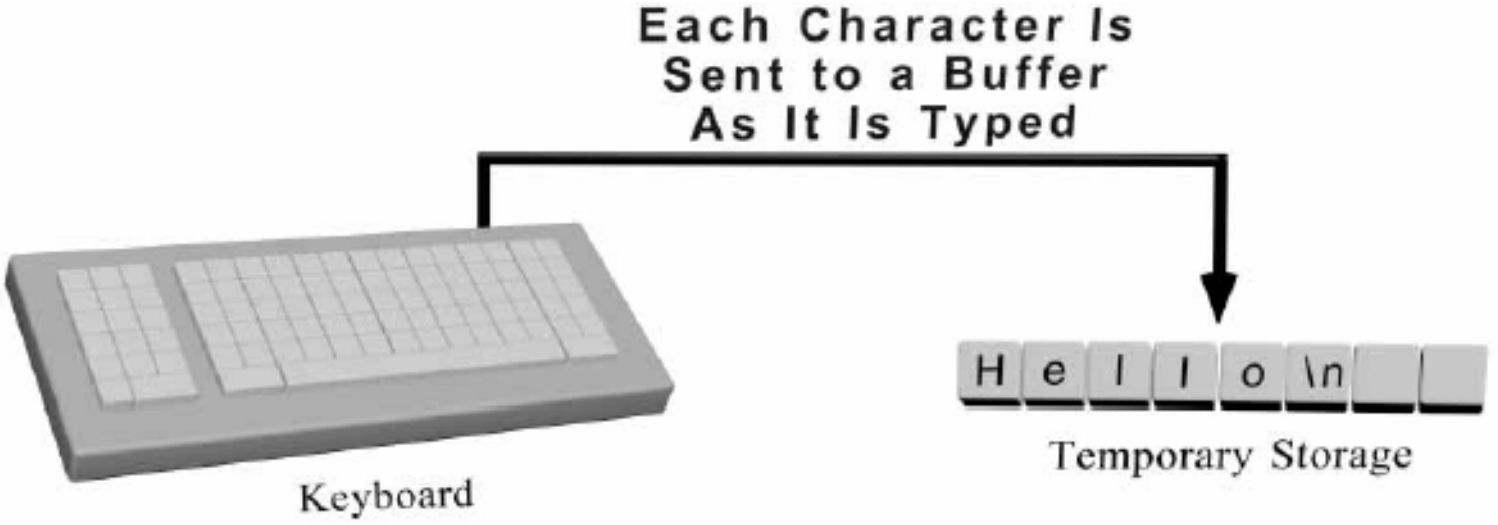

Figure 3.6 Typed keyboard characters are first stored in a buffer

关于“幻影”字符

- scanf()在连续接收字符时
  - 回车键会被作为后续scanf的输入
  - 解决方法
    - 在scanf前写一个打印函数，输出提示信息
    - page 95
    - page 97
    - fflush(stdin) (真的管用么）
    - rewind(stdin)
    - 阅读《现代方法》3.2

Formatted Output

Program 3.13

```
 1  #include <stdio.h>
 2  int main()
 3  {
 4    printf("\n%d", 6);
 5    printf("\n%d", 18);
 6    printf("\n%d", 124);
 7    printf("\n---");
 8    printf("\n%d\n", 6+18+124);
 9
10    return 0;
11  }
```

```
6
18
124
---
148
```

Formatted Output

Program 3.14

```
 1  #include <stdio.h>
 2  int main()
 3  {
 4    printf("\n%3d", 6);
 5    printf("\n%3d", 18);
 6    printf("\n%3d", 124);
 7    printf("\n---");
 8    printf("\n%3d\n", 6+18+124);
 9
10    return 0;
11  }
```

Field width specifier

```
  6
 18
124
---
148
```

Formatted Output

Table 3.6 Effect of Field Width Specifiers

Specifier	Number	Display	Comments
`%2d`	3	`⅄3`	Number fits in field
`%2d`	43	`43`	Number fits in field
`%2d`	143	`143`	Field width ignored
`%2d`	2.3	Compiler dependent	Floating-point number in an integer field
`%5.2f`	2.366	`⅄2.37`	Field of 5 with 2 decimal digits
`%5.2f`	42.3	`42.30`	Number fits in field
`%5.2f`	142.364	`142.36`	Field width ignored but fractional specifier is used
`%5.2f`	142	Compiler dependent	Integer in a floating-point field

Format Modifiers

- **Left justification:** `printf("%-10d",59);` produces the display `59^^^^^^^^`
- **Explicit sign display:** `printf("%+10d",59);` produces the display `^^^^^^^+59`
- Format modifiers may be combined
  - `%-+10d` would cause an integer number to both display its sign and be left-justified in a field width of 10 spaces
    - The order of the format modifiers is not critical

      `%+-10d` is the same

Symbolic Constants

- **Literal data** refers to any data within a program that explicitly identifies itself
- Literal values that appear many times in the same program are called **magic numbers**
- C allows you to define the value once by equating the number to a **symbolic name**
  - `#define SALESTAX 0.05`
  - `#define PI 3.1416`
  - Also called **symbolic constants** and **named constants**
  - const限定符
    - const int MONTHS=12;
    - 限定一个变量为只读

Case Study: Guess the number

- Randomly generate an integer
- Input an integer
- Output the result
  - Greater
  - Smaller
  - Win (equal)

Homework

- 1. P87, 3.1简答题 3 （编写程序进行验证）
- 2. P115, 编程题 4
- 3. P116, 编程题 5
- 4. P116, 编程题 6
- 5. P116, 编程题 7
- 6. P116, 编程题 8
- 7. P116, 编程题 9
- 8. 编写一个程序，要求用户输入一个3位数，然后按数位的逆序打印出这个数
  - 想想scanf能不能做到