Jiangxi University of Science and Technology

Chapter 3 Processing and Interactive Input

Leture0301 Assignment

TASK

THE C PROGRAMMING LANGUAGE

```
#include<stdio.h>
int main()
{
    printf("displaying cat\n");
    printf("Extra task \n");
    printf("                          '    '   \n");
    printf("                           ____     \n");
    printf("                          ( 00 )   \n");
    printf("                    +=========\/     \n");
    printf("                   / || $$$ ||      \n");
    printf("                  *  || --- ||      \n");
    printf("                     **     **      \n");
    return 0;
}
```

```
1   #include<stdio.h>
2   int main()
3   {
4       printf("displaying cat\n");
5       printf("Extra task \n");
6       printf("                          '    '   \n");
7       printf("                           ____     \n");
8       printf("                          ( 00 )   \n");
9       printf("                    +=========\/     \n");
10      printf("                   / || $$$ ||      \n");
11      printf("                  *  || --- ||      \n");
12      printf("                     **     **      \n");
13      return 0;
14  }
15
```

```
G:\salam1\bin\Debug\salam1.exe
displaying cat
Extra task
                          '    '
                           ____
                          ( 00 )
                    +=========/
                   / || $$$ ||
                  *  || --- ||
                     **     **

Process returned 0 (0x0)   execution time : 0.103 s
Press any key to continue.
```

3.1 Assignment (赋值)

THE C PROGRAMMING LANGUAGE

- The general syntax for an **assignment statement** is
  - ***variable = operand***;
  - The ***operand*** to the right of the ***assignment operator*** (赋值运算符) = 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***
  - ***sum = 3 + 7;***
  - ***product = .05 * 14.6;***

3.1 Assignment

- Program 3.1

```
1.  #include <stdio.h>
2.  int main(){
3.  float length, width, area;
4.  length=27.2;  // assignment statement
5.  width=13.8; // assignment statement
6.  area=length*width;
7.  printf("the length of the rectangle is %f\n", length);
8.  printf("the width of the rectangle is %f\n", width);
9.  printf("the area of the rectangle is %f\n", area);
10. return 0;
11. }
```

3.1 Assignment

- Precedence and associativity
  - = has the **lowest precedence** of all the binary and unary arithmetic operators.
  - All = operators have the same precedence, Operator has **right-to-left** associativity
  - a = b = c = 25;

3.1 Assignment

THE C PROGRAMMING LANGUAGE

➢ Implicit Type Conversions (隐式类型转换)

- The automatic conversion across an assignment operator is called an implicit type conversion.
- **double** result=4; **//integer 4 is converted to 4.0**
- int 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***

3.1 Assignment

THE C PROGRAMMING LANGUAGE

- **Explicit Type Conversions** (显式类型转换)
  - The operator used to **force** the conversion of a value to another type is the ***cast operator***
  - **(*dataType*) expression**
- Example:
  - **int** n=2;
  - **double** m=1/(**double**)n;

3.1 Assignment

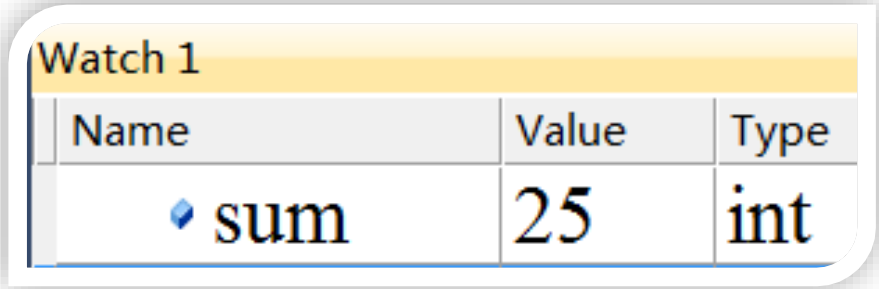
- Compound Assignment(复合赋值)
  - += -= *= /= %=
  - sum += 10 **//sum = sum + 10**
  - price *= rate **//price = price * rate**
  - price*=rate+1 **//price=price*(rate+1)**

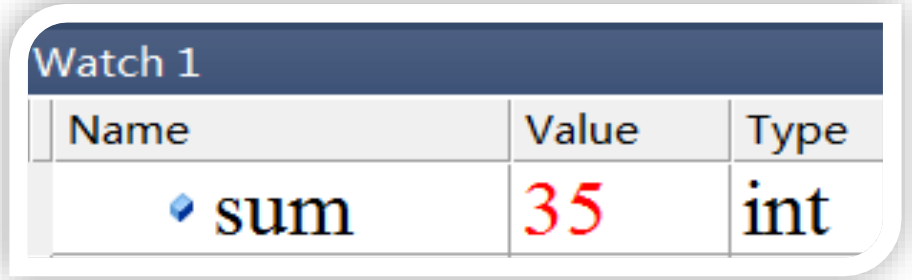
3.1 Assignment

THE C PROGRAMMING LANGUAGE

- Compound Assignment

```
1. #include <stdio.h>
2. int main(){
3. int sum=25;
4. printf("the number stored in sum is %d\n",sum);
5.  sum+=10; //sum=sum+10;
6. printf("the number now stored in sum is %d\n",sum);
7. return 0;
8. }
```





3.1 Assignment

THE C PROGRAMMING LANGUAGE

- Program 3.3 Accumulating (累加)

```
1.  #include <stdio.h>
2.  int main(){
3.  int sum=0;
4.  printf("the sum is initialy set to %d\n",sum);
5.  sum+=96;
6.  printf("sum is now  %d\n", sum);
7.  sum+=70;
8.  printf("sum is now  %d\n", sum);
9.  sum+=85;
10. printf("sum is now  %d\n", sum);
11. sum+=60;
12. printf("sum is now  %d\n", sum);
13. return 0;
14. }
```

3.1 Assignment

THE C PROGRAMMING LANGUAGE

- Accumulating
  - The first statement initializes sum to 0
  - This removes any previously stored value in sum that would **invalidate** the final total
  - A previously stored number, if it has not been initialized to a specific and known value, is frequently called a **garbage value**

3.1 Assignment

- Counting (计数)
  - A counting statement is very similar to the accumulating statement
  - **variable = variable + fixedNumber;**
  - Examples: i = i + 1; and m = m + 2;
- Increment operator (++) (递增运算符):
  - **variable=variable+1** can be replaced by **variable++** or **++variable**
  - i=i+1; i++; or ++i;
  - n=n+1; n++; or ++n;
  - count=count+1; count++; or ++count;

3.1 Assignment

- Program 3.4 Counting

```
1.  #include <stdio.h>
2.  int main(){
3.  int count=0;
4.          printf("the count is initialy set to %d\n",count);
5.          count++;
6.          printf("count is now  %d\n", count);
7.          count++;
8.          printf("count is now  %d\n", count);
9.          count++;
10.         printf("count is now  %d\n", count);
11.         count++;
12.         printf("count is now  %d\n", count);
13.         return 0;
14. }
```

3.1 Assignment

- **Prefix increment operator (前置递增)**
  - **k = ++n;**
  - is equivalent to
  - **n = n + 1; // increment n first**
  - **k = n; // assign n's value to k**
  - **在一个表达式中，先递增， 后使用**
  - **i=3, j=++i, j=?, i=?**

3.1 Assignment

- **Postfix increment operator (后置递增)**
  - **k = n++;**
  - is equivalent to
  - **k = n; // assign n's value to k**
  - **n = n + 1; // and then increment n**
  - **在一个表达式中，先使用， 后递增**
  - **i=3, j=i++ , j=?，i=?**

3.1 Assignment

THE C PROGRAMMING LANGUAGE

- Prefix decrement operator: **k = --n**
  - decrements the value of n by 1
  - assigns the value of n to k
  - **i=3 , j= - - i, j=?, i=?**
- Postfix decrement operator: **k = n- -**
  - assigns the current value of n to k
  - reduces the value of n by 1
  - **i=3, j=i - - , j=?, i=?**

Reference

- BOOK
- Some part of this PPT given by Prof 区 **Chengtian Ouyang)**
- with special thank
- https://www.codingunit.com/c-tutorial hello-world