

**CONFIDENTIAL**

# **C Programming Introduction**

## **Week 8: Loops**

**Lecturer : Do Quoc Huy**  
**Dept of Computer Science**  
**Hanoi University of**  
**Technology**



# Topic of this week

- Loops
  - Class Lecture Review
    - The While,do Repetition Structure
    - Notes and Observations
    - Continue and break
  - Programming Exercises

# The While,do Repetition Structure

- While Statement

- The expression is evaluated. If it is *true*, statement is executed and expression is reevaluated.
- This cycle continues until expression becomes *false*.

```
while (expression) {  
    Statement1;  
    Statement2;  
    ...  
}
```

# The While,do Repetition Structure

- Example of **While**

```
#include <stdio.h>
#define PERIOD '.'
main() {
    char C;
    while ((C = getchar()) != PERIOD)
        putchar(C);
    printf("Good Bye.\n");
}
```

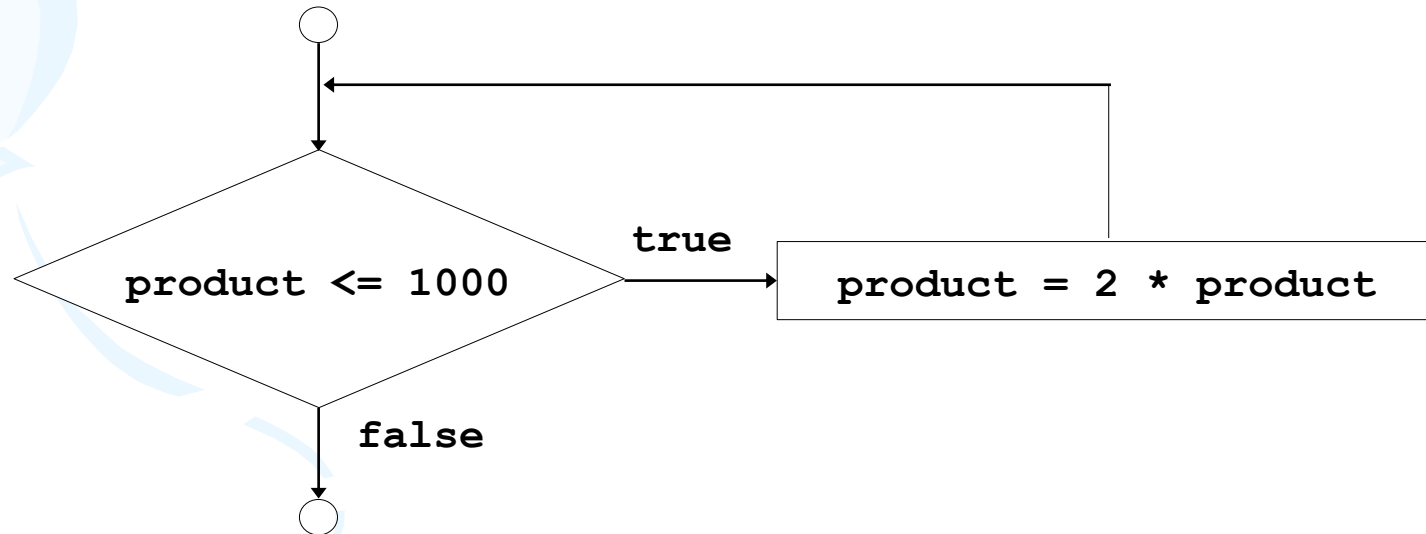


**Result?**

# The While,do Repetition Structure

- Example:

```
int product = 2;  
while ( product <= 1000 )  
    product = 2 * product;
```





# The While,do Repetition Structure

- Do-While Statement

- The **do-while**, tests at the bottom after making each pass through the loop body; the body is always executed at least once.

```
do {  
    statement1;  
    statement2;  
    ...  
} while (expression);
```

# The While,do Repetition Structure

- Example of **Do-While**

```
int i = 1, sum = 0;  
do {  
    sum += i;  
    i++;  
} while (i <= 50);  
printf("The sum of 1 to 50 is %d\n", sum);
```



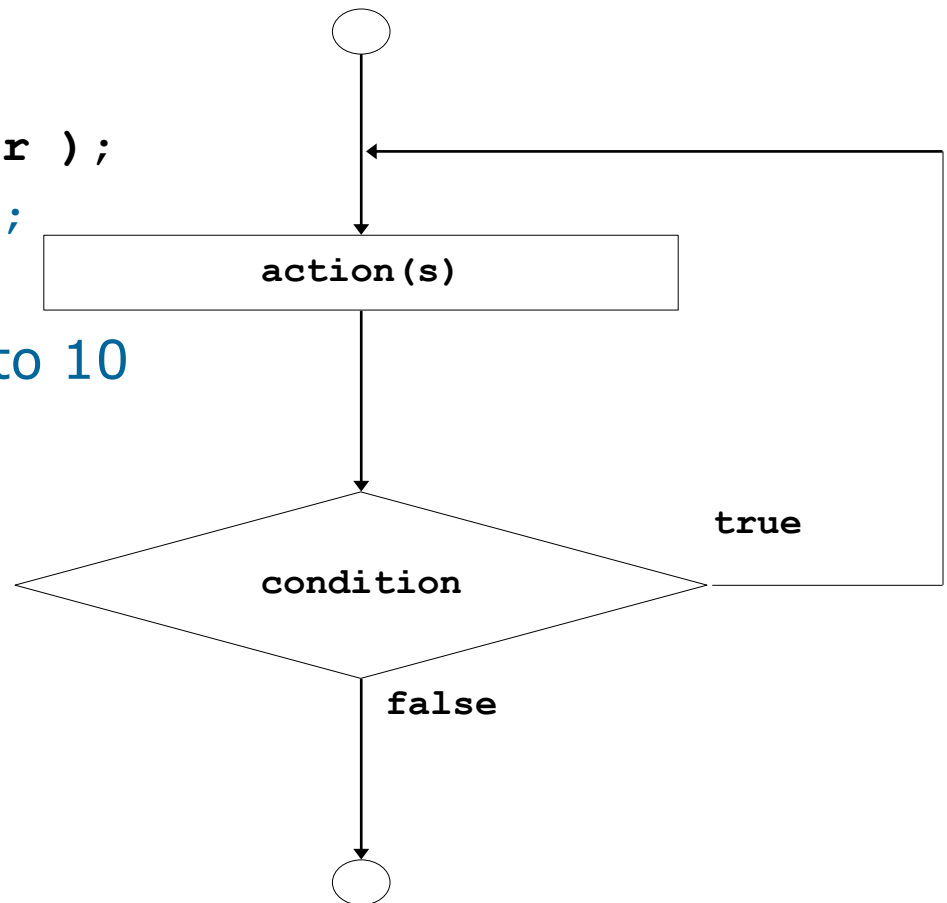
**Result?**

# The While,do Repetition Structure

- Example (letting `counter = 1`)

```
do {  
    printf( "%d  ", counter );  
} while ( ++counter <= 10 );
```

Prints the integers from 1 to 10







# Continue and Break

- Break and Continue Statement

- The **break** statement provides an early exit from for, while, and do.

```
break;
```

- The **continue** statement is related to break, but less often used; it causes the **next iteration** of the enclosing for, while, or do loop to begin.

```
continue;
```



# Continue and Break

- Example of Break and Continue

```
int c;
while ((c = getchar()) != -1) {
    if (C == '.')
        break;
    else if (c >= '0' && c <= '9')
        continue;
    else putchar(c);
}
printf("*** Good Bye ***\n");
```



# Exercise 8.1

- Write a program that copies content inputted from the keyboard to the screen, but replace the sequence of blank characters by only one blank character.
- You can use `getchar()` and `putchar()` method to carry out this program.



## Exercise 8.2

- Write a program that replaces characters such as: `tab, \t, \b` by `\\` character in the input string and print out.
- You can use `getchar()` method to carry out this program.
- You can use *if* structure or *switch* structure.



## Exercise 8.3

- Given number  $a$  from user input, calculate square cube ( $\sqrt{a}$ ) by using newton method.



## Exercise 8.4

- How to compute the payroll for a company?
- Write and compile the program below to see how you can use **while** statement to do this task.



# exercise8\_4.c

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

```
int
```

```
main(void)
```

```
{
```

```
    double total_pay;    /* company payroll    */
```

```
    int    count_emp;    /* current employee    */
```

```
    int    number_emp;    /* number of employees */
```

```
    double hours;        /* hours worked        */
```

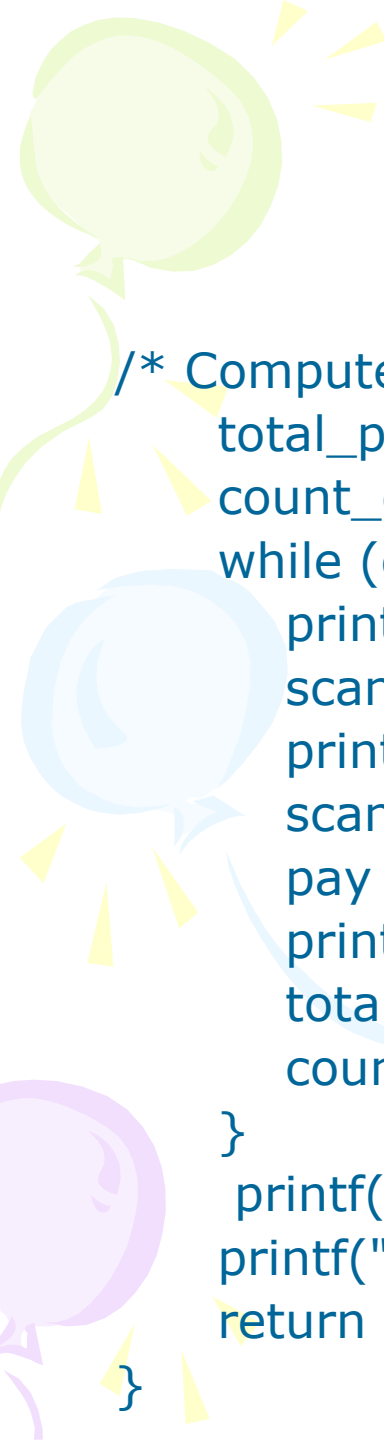
```
    double rate;          /* hourly rate          */
```

```
    double pay;           /* pay for this period */
```

```
    /* Get number of employees. */
```

```
    printf("Enter number of employees> ");
```

```
    scanf("%d", &number_emp);
```



```
/* Compute each employee's pay and add it to the payroll. */
total_pay = 0.0;
count_emp = 0;
while (count_emp < number_emp) {
    printf("Hours> ");
    scanf("%lf", &hours);
    printf("Rate > $");
    scanf("%lf", &rate);
    pay = hours * rate;
    printf("Pay is $%6.2f\n\n", pay);
    total_pay = total_pay + pay;
    count_emp = count_emp + 1;
}
printf("All employees processed\n");
printf("Total payroll is $%8.2f\n", total_pay);
return (0);
```





## Exercise 8.5

- Write a program that use *while* structure to analysis of examination results: how many passed students and failed students.
- You can simply ask user to show that a student is passed or failed by entering a presented number: 1 is passed and 2 is failed.



## Exercise 8.6

- Use `do...while` statement to print out integers that is smaller than a preceded number.
- Note that the `do...while` statement always performs one time at least.



## Exercise 8.7

- We would like a program to average a set of grades.
- Algorithm notes:
  - We need a running sum of grades, and a running count of how many grades have been read so far.
  - We need to read until we get a sentinel value | let's use a negative grade to indicate we are done.
  - Need to be sure we print prompts.



# Exercise 8.8

- Write a program that compute  $n!$  using a loop.
- You can use:
  - Counter" variable,  $i$ , ranging from 1 to  $n$ .
  - Running product  $f$ , tracking  $i!$ .