

# Programming with C I

Fangtian Zhong  
CSCI 112

Gianforte School of Computing  
Norm Asbjornson College of Engineering  
E-mail: [fangtian.zhong@montana.edu](mailto:fangtian.zhong@montana.edu)

# while Statement Syntax

```
while (loop repetition condition)  
    statement;
```

```
/* display N asterisks. */  
count_star = 0;  
while (count_star < N) {  
    printf("*");  
    count_star = count_star + 1;  
}
```

# Increment and Decrement Operators

- `counter = counter + 1`  
`count += 1`  
`counter++`  
`++counter`
- `counter = counter - 1`  
`count -= 1`  
`counter--`  
`--counter`

# while Statement Syntax

```
while (loop repetition condition)
    statement;
```

```
/* display N asterisks. */
count_star = 0;
while (count_star < N) {
    printf("*");
    count_star = count_star + 1;
}
```

# while Statement Syntax

```
while (loop repetition condition)  
    statement;
```

```
/* display N asterisks. */  
count_star = 0;  
while (count_star < N) {  
    printf("*");  
    count_star += 1;  
}
```

# Compound assignment

Operator	Definition
+	addition
-	subtraction
*	multiplication
/	division
%	remainder

➤ Can do these too:

`+=`

`-=`

`*=`

`/=`

`%=`

# Increment and Decrement Operators

## side effect

- – a change in the value of a variable as a result of carrying out an operation

# Increment and Decrement Operators

**Before..**



**Increments...**

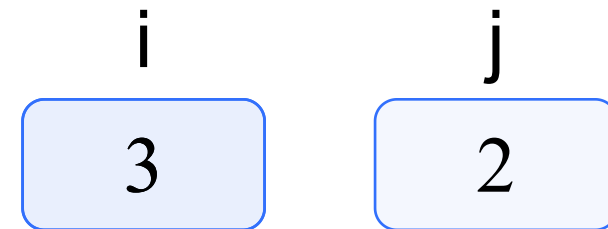
**$j = ++i;$**

prefix:  
Increment i and then use it.

**$j = i++;$**

postfix:  
Use i and then increment it..

**After...**





# The **for** Statement Syntax

```
for (initialization expression;  
     loop repetition condition;  
     update expression)  
statement;
```

```
/* Display N asterisks. */  
for (count_star = 0;  
     count_star < N;  
     count_star += 1)  
printf("*");
```

# do-while Statement



For conditions where we know that a loop must execute **at least one time.**

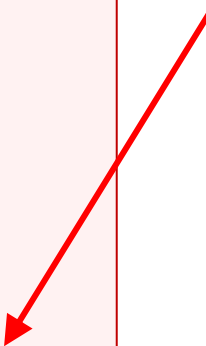
1. Get a *data value*
2. If *data value* isn't in the acceptable range, go back to step 1.

# do-while Syntax

```
do
    statement;
while (loop repetition condition);

/* Find first even number input */
do
    status = scanf("%d", &num);
while (status > 0 && (num % 2) !=
0);
```

We will talk more about the output of scanf next time.



# Computing a Sum or Product in a Loop

## ➤ accumulator

- a variable used to store a value being computed in increments during the execution of a loop

# Computing Factorial

## ➤ logical complement (negation)

- loop body executes for decreasing value of **i** from **n** through 2
- each value of **i** is incorporated in the accumulating product
- loop exit occurs when **i** is 1

# Nested Loops

- Loops may be nested just like other control structures
- Nested loops consist of an outer loop with one or more inner loops
- Each time the outer loop is repeated, the inner loops are reentered, their loop control expressions are evaluated, and all required iterations are performed

# Table Compound Assignment Operators

## Statement with Simple Assignment Operator

```
count_emp = count_emp + 1;  
time = time - 1;  
total_time = total_time +  
    times;  
product = product * item;  
n = n * (x + 1);
```

## Equivalent Statement with Compound Assignment Operator

```
count_emp += 1;  
time -= 1;  
total_time += times;  
product *= item;  
n *= (x + 1);
```

# Loop Control Components

- 🏆 initialization of the loop control variable
  - 🏆 test of the loop repetition condition
  - 🏆 change (update) of the loop control variable
- 🏆 the **for** loop supplies a designated place for each of these three components



# Figure Function to Compute Factorial

```
/*
 * Computes n!
 * Pre: n is greater than or equal to zero
 */
int
factorial(int n)
{
    int i,          /* local variables */
        product;   /* accumulator for product computation */

    product = 1;
    /* Computes the product n x (n-1) x (n-2) x ... x 2 x 1 */
    for (i = n; i > 1; --i) {
        product = product * i;
    }

    /* Returns function result */
    return (product);
}
```

# Endfile-Controlled Loop Design

- 🛡️ Get the first *data value* and save *input status*
- 🛡️ while *input status* does not indicate that end of file has been reached
- 🛡️ Process data value
- 🛡️ Get next data value and save *input status*

# Figure Batch Version of Sum of Exam Scores Program

```
/*
 * Compute the sum of the list of exam scores stored in the file scores. txt
 */
#include <stdio.h>

int
main(void)
{
    int sum = 0,          /* sum of scores input so far */
        score,          /* current score */
        input_status;   /* status value returned by scanf */

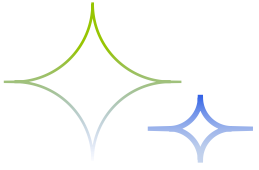
    printf("Scores\n");

    input_status = scanf("%d", &score);
    while (input_status != EOF) {
        printf("%5d\n", score);
        sum += score;
        input_status = scanf("%d", &score);
    }

    printf("\nSum of exam scores is %d\n", sum);

    return (0);
}

Scores
  55
  33
  77
sum of exam scores is 165
```



# THE END

Fangtian Zhong  
CSCI 112

Gianforte School of Computing  
Norm Asbjornson College of Engineering  
E-mail: [fangtian.zhong@montana.edu](mailto:fangtian.zhong@montana.edu)