Iterations

Kornchawal Chaipah, PhD

Computer Engineering Department,
Khon Kaen University

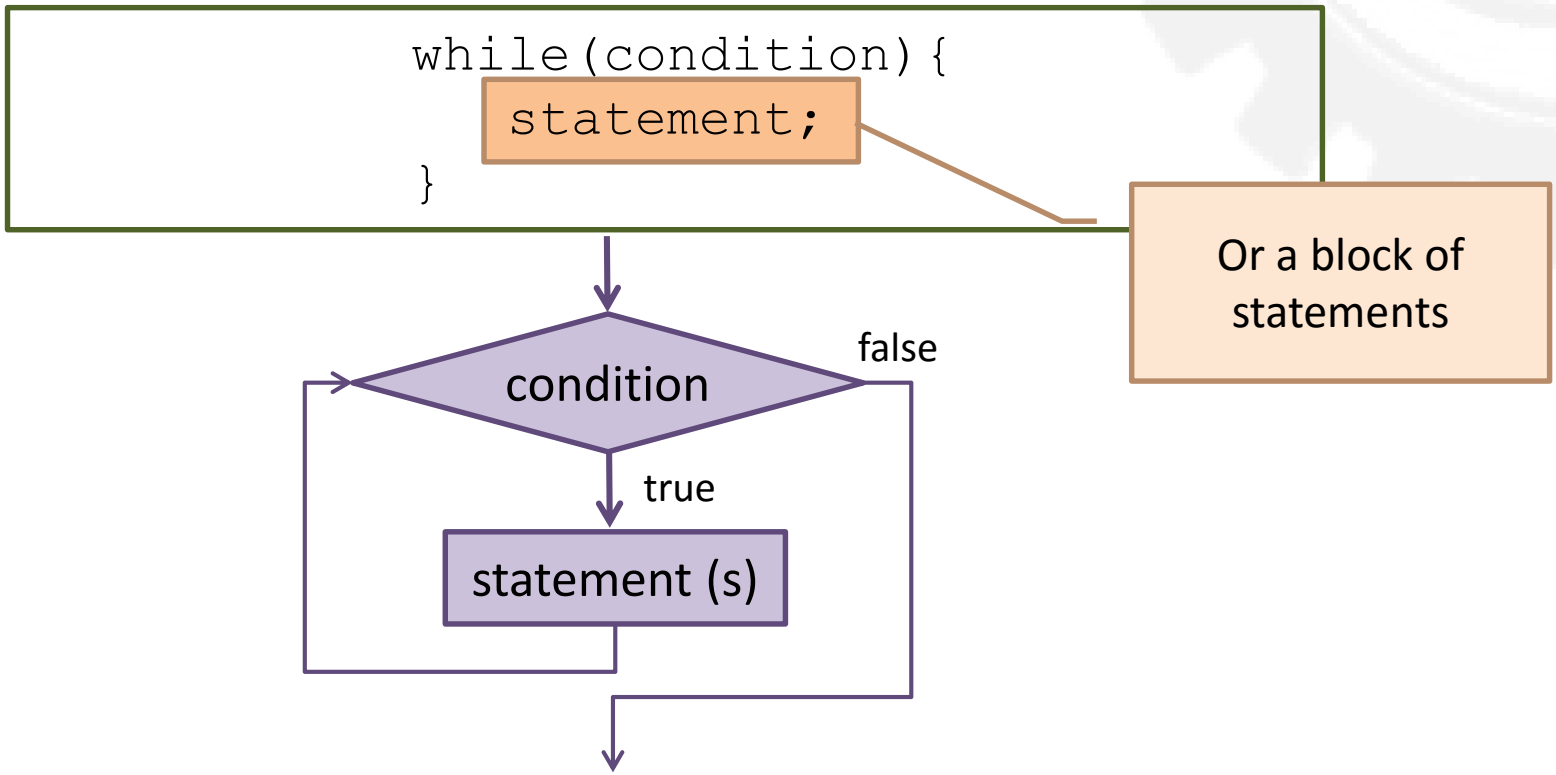
Agenda

- Iterations
  - while
  - do..while
  - for
- Nested loop
- Lots and lots of loop examples!

Iteration (Loop)

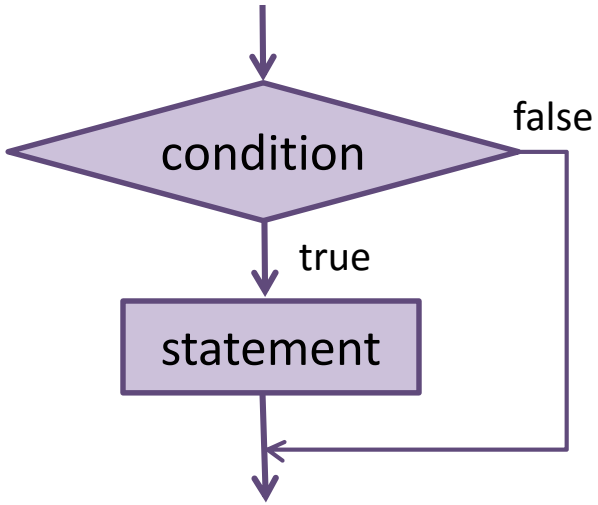
- The repetition of a statement (or a block of statements) in a program
- Used to many programs
  - Repeatedly output values
  - Repeatedly read values
  - Array processing
- In C/C++
  - while
  - do..while
  - for

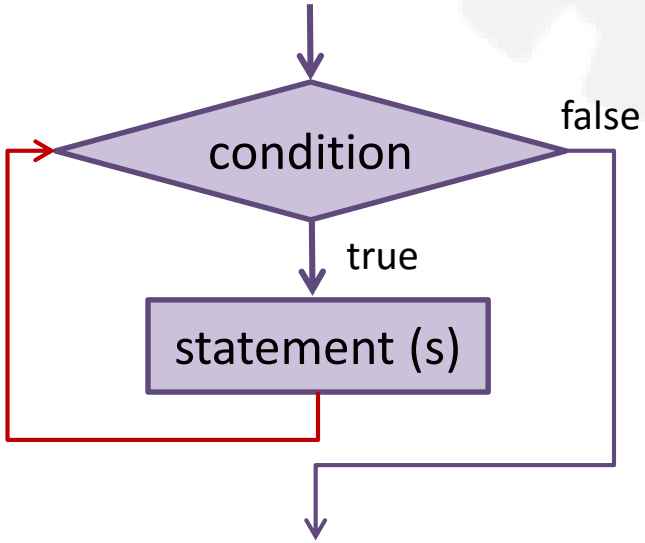
The while Statement



- If condition is true (non-zero), then execute the statement,
  - Then check the condition and execute the statement again until the condition is false (zero)
- If not, just do nothing and skip

if VS. while

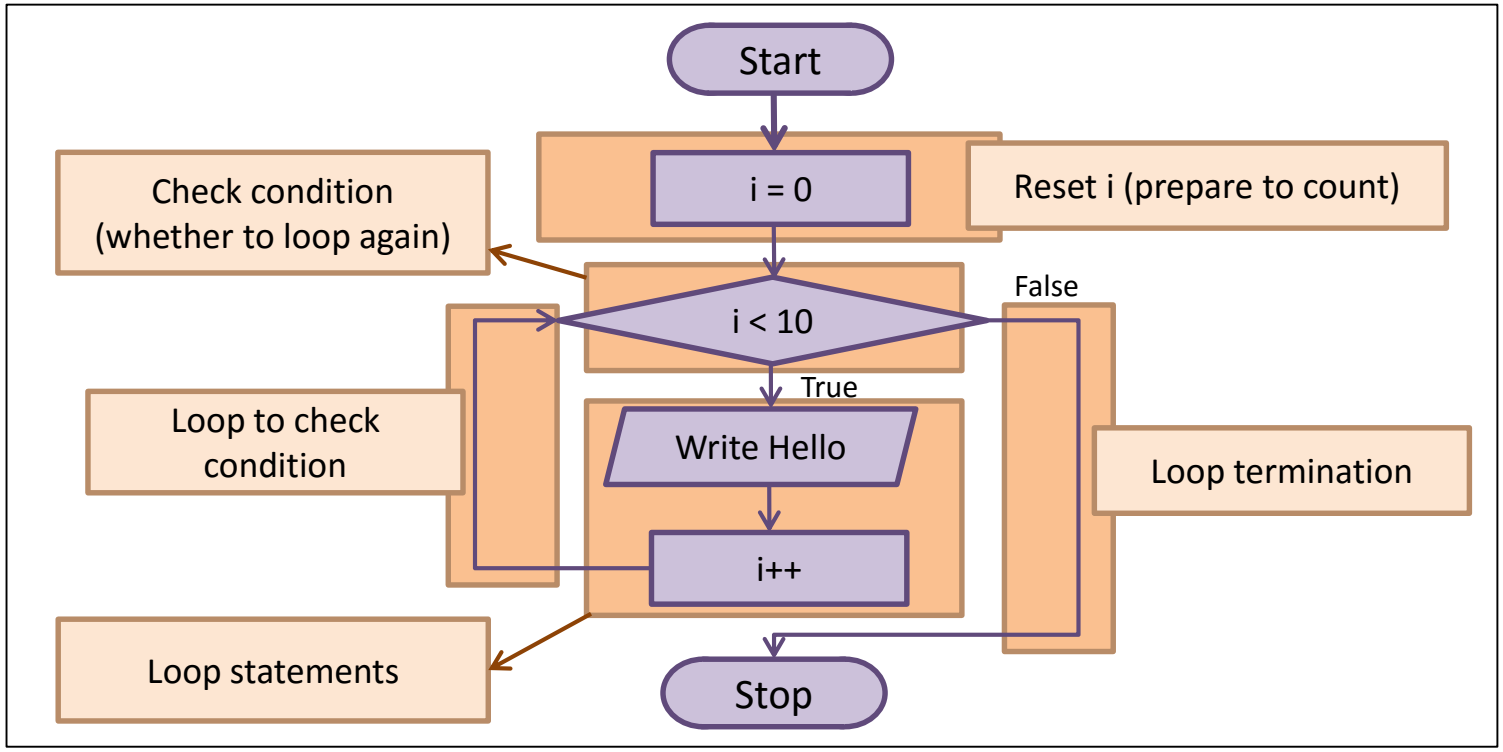




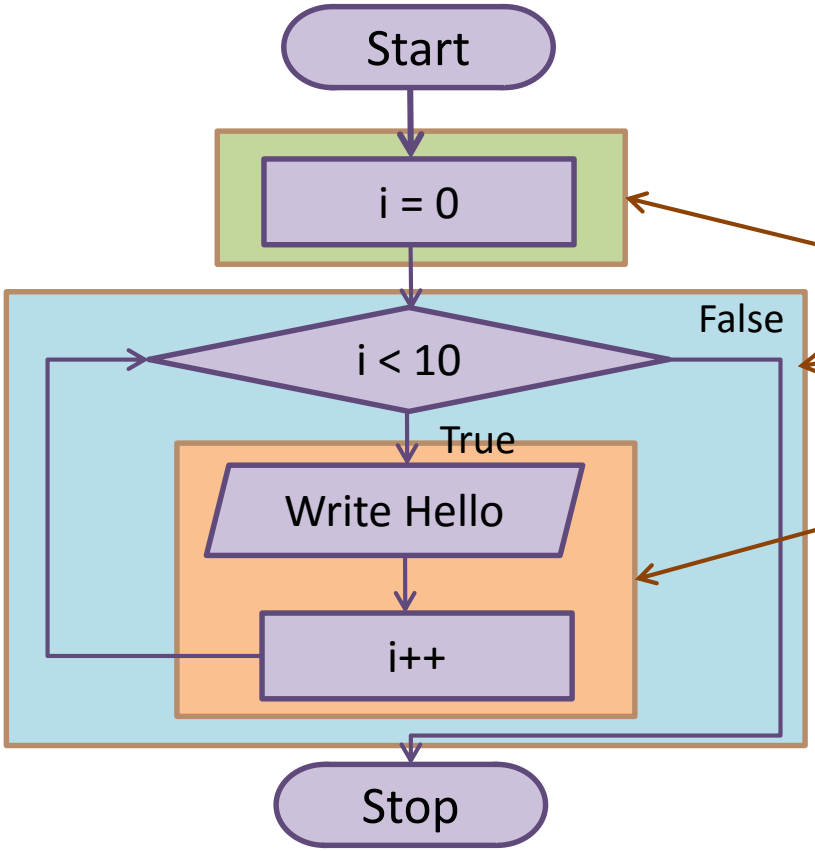
- Both need a condition
- while executes statements repeatedly, but if does it once (no loop)

Hello! Hello! Hello!

- Problem: display "hello" 10 times
  - Do this several times → use loop!



Flowchart VS. Code




```
int main(){
    int i = 0;
    while(i < 10){
        cout << "Hello" << endl;
        i++;
    }
}
```

Hello! Hello! Hello!

- Problem: display "hello" 10 times
  - Do this several times → use loop!

Code:

```
int main(){
   int i = 0;
   while(i < 10){
        cout << "Hello" << endl;
        i++;
   }
}
```

What does i do in this code?

Output:

```
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
```

Hello! Hello! Hello!

- Problem: display "hello" 10 times
  - Do this several times → use loop!

Code:

```
int main(){
   int i = 0;
   while(i < 10){
        cout << "Hello" << endl;
        i++;
   }
}
```

i is basically a **"counter"**
It tells us:
1) How many times we already execute a statement(s)
2) When to terminate the loop

Output:

```
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
```

Hello! Hello! Hello!

- Problem: display "hello" 100 times
  - Do this several times → use loop!

Code:

```
int main(){
   int i = 0;
   while(          ){
        cout << "Hello" << endl;
        i++;
   }
}
```

Output:

```
Hello
Hello
Hello
Hello
Hello
Hello
Hello
…
Hello
Hello
```

Hello! Hello! Hello!

- Problem: display "hello" n times
  - Do this several times → use loop!

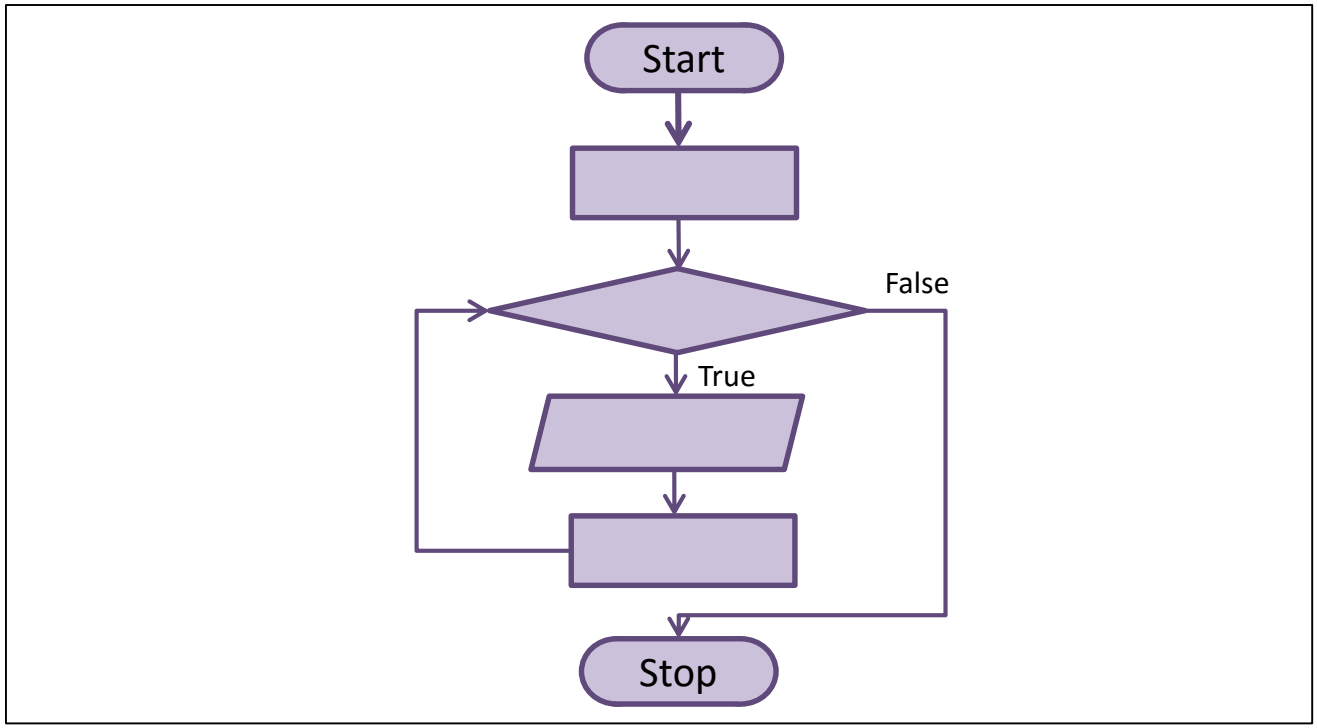
Code:

```
int main(){




}
```

Output:

Count 1-10

- Problem: display integers from 1-10
  - Yes, use loop!



Count 1-10

- Problem: display integers from 1-10
  - Yes, use loop!

Code:

```
int main(){
   int i = 0;
   while(          ){
        cout <<          << endl;
        i++;
   }
}
```

Output:

```
1
2
3
4
5
6
7
8
9
10
```

Count 1 to 100 … to n

- Problem: display integers from 1-100 and from 1-n
  - Yes, use loop!

Code (1-100):

```
int main(){




}
```

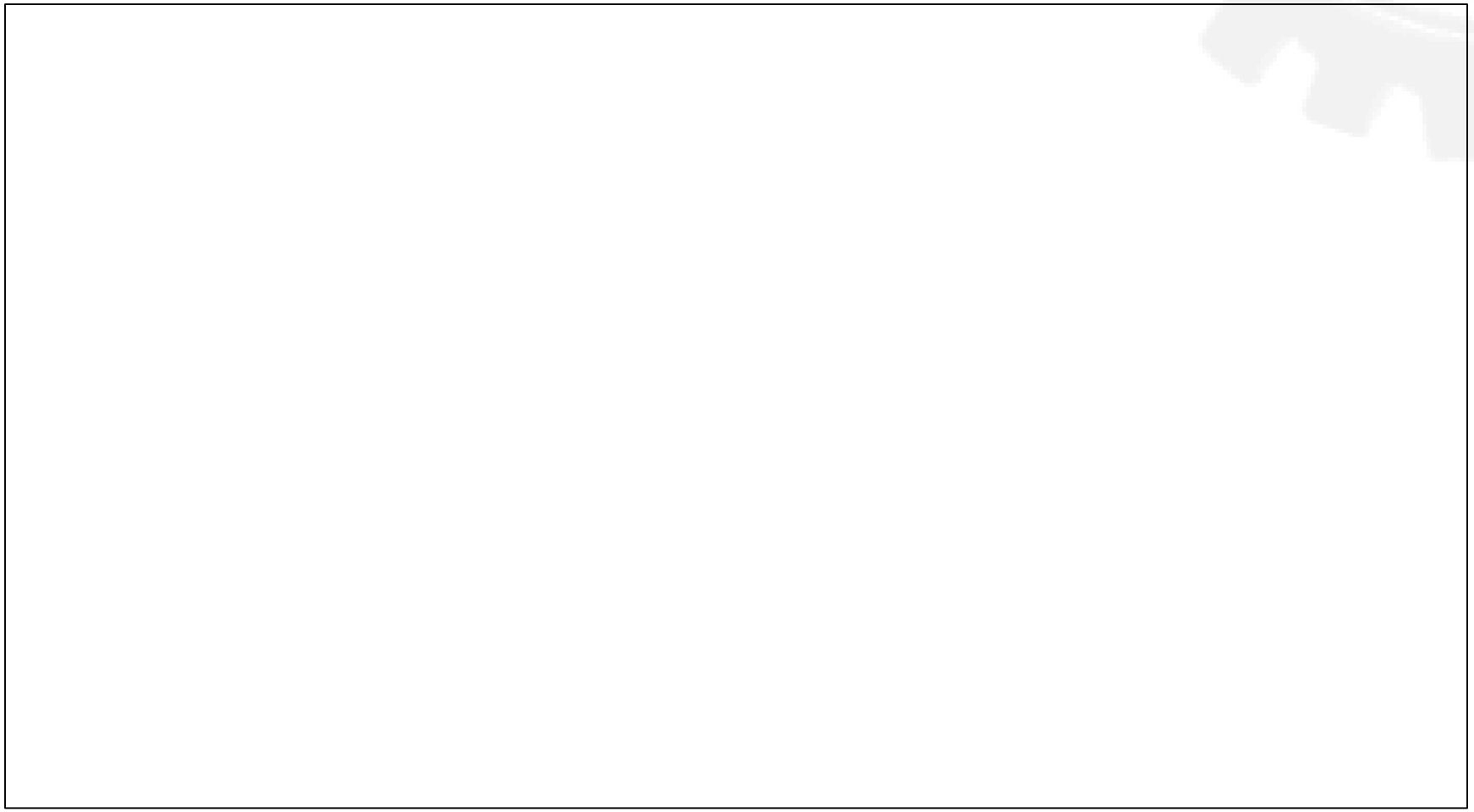
Code (1-n):

```
int main(){




}
```

Sum of Consecutive Integers

- Problem: sum integers from 1 to n
- What you know
  - If n = 5, sum = 1+2+3+4+5 = 15
- What are inputs, outputs, processes, and decision point(s), loop statement(s)?
- Let's write a flowchart first!

Sum of Consecutive Integers Flowchart



Sum of Consecutive Integers Code

Code:

```
int main(){
   int n, i = 1; // ok, we need a counter
   int sum = 0; // and we need a sum
   // what else do we need?









}
```

Output:

Sum of Consecutive Integers Code

Code:

```
int main(){
   int n, i = 1; // ok, we need a counter
   int sum = 0; // and we need a sum
   cout << "Enter a positive integer: ";
   cin >> n;

   while (i <= n)
        sum = sum + i++;    //one statement

    cout << "The sum of the first " << n
    << " integers is " << sum << endl;

    return 0;
}
```

Output:

Sum of Consecutive Integers Code

Code:

```
int main(){
   int n, i = 1; // ok, we need a counter
   int sum = 0; // and we need a sum
    cout << "Enter a positive integer: ";
    cin >> n;
    while (i <= n) {    //statement block
        sum = sum + i;
        cout << "sum to " << i << "is"
        << sum << endl;
        i++;
    }
    cout << "The sum of the first " << n
    << " integers is " << sum << endl;
    return 0;
}
```

Output:

Sum of Reciprocals

- Reciprocals: 3 and 1/3 are reciprocals (multiplication result = 1)
- Problem: get a bound, find the sum of the i reciprocals (1/i) within a bound
- Confused? Let's see the code

Sum of Reciprocals Code

Code:

```
int main(){
   int bound;
   cout << "Enter a positive integer: ";
   cin >> bound;
   double sum = 0;
   int i = 0;
   while(sum < bound){
        sum += 1.0/++i;
   }
   cout << "The sum of the first " << i
        << " reciprocals is " << sum
        << endl;
   return 0;
}
```

Output:

More loop example: Repeating a computation

Code:

```
#include <cmath>
int main(){
   double x;
   cout << "Enter a positive number: ";
   cin >> x;
   while(x > 0){
        cout << "Square root (" << x <<
        ") = " << sqrt(x) << endl;
        cout << "Enter another positive "
        << "number (or 0 to quit): ";
        cin >> x;
   }
   return 0;
}
```

Output:

N Fibonacci Numbers

- Problem: print the first n Fibonacci numbers
- What you know
  - Fibonacci numbers =>
- What are inputs, outputs, processes, and decision point(s), loop statement(s)?
- What happens in each round of our loop?
- How many "buffers" do we need?

$$f_0 = 0$$
$$f_1 = 1$$
$$f_2 = f_0 + f_1 = 1$$
$$f_3 = f_1 + f_2 = 2$$
$$f_4 = f_2 + f_3 = 3$$
$$f_5 = f_3 + f_4 = 5$$
$$\ldots$$
$$f_n = f_{n-2} + f_{n-1}$$

Computing Fibonacci Numbers

$f_0 = 0$

$f_1 = 1$

$f_2 = f_0 + f_1 = 0 + 1 = 1$

$f_3 = f_1 + f_2 = 1 + 1 = 2$

$f_4 = f_2 + f_3 = 1 + 2 = 3$

$f_5 = f_3 + f_4 = 2 + 3 = 5$

...

$f_n = f_{n-2} + f_{n-1}$

The loop computes new value from old and older value

N Fibonacci Numbers Code

Counter process: keeping the correct number of loops

```
int main() {
  int n;
  cout << "Enter a positive integer: ";
  cin >> n;
  cout << "First " << n << " Fib. numbers: ";
  if (n >= 1) cout << "0";
  if (n >= 2) cout << ", 1";
  long a = 0, b = 1;
  int i = 3;
  while (i <= n) {
    long f = a + b;
    cout << ", " << f;
    a = b; b = f; i++;
  }
}
```

Updating a (f_n2) and b (f_n1) for the next round

$$f_0 = 0$$
$$f_1 = 1$$
$$f_2 = f_0 + f_1 = 1$$
$$f_3 = f_1 + f_2 = 2$$
$$f_4 = f_2 + f_3 = 3$$
$$f_5 = f_3 + f_4 = 5$$
$$\cdots$$
$$f_n = f_{n-2} + f_{n-1}$$

In the Fibonacci loop

Code:

```
while (i <= n) {
    long f = a + b;      // new value of fib
    cout << ", " << f;   // print out
    // a keeps f2 (older), b keeps f1 (old)
    a = b;               // old value (b: f1)
                         // becomes older (a: f2)
    b = f;               // new value (f: fi)
                         // becomes old one (b: f1)
    i++;                 // counting number
}
```

Tracing values in the loop

Code:

```
while (i <= n) {
        long f = a + b;
        cout << ", " << f;
        // a keeps f2 (older)
        // b keeps f1 (old)
        a = b;
        b = f;
        i++;
}
```

f	a (f2)	b (f1)	i
	0	1	3

Terminating a Loop

- Can we put in another kind of condition?

```
while(true){
    statements;
}
```

- How can we terminate this loop to prevent having an "infinite" loop
- Use break; or exit(0); or return;
  - break; terminates only the current loop
  - exit(0); terminates the current function
  - return; terminates the current function

Sum of Consecutive Integers Revisited

Code:

```
int main(){
   int n, i = 1;
   cout << "Enter a positive integer: ";
   cin >> n;
   long sum = 0;
   while(true){
        if (i > n) break;
        sum += i++;
   }
   cout << "The sum of the first " << n
   << " integers is " << sum << endl;
}
```

Output:

Sum of Consecutive Integers Revisited

Code:

```
int main(){
   int n, i = 1;
   cout << "Enter a positive integer: ";
   cin >> n;
   long sum = 0;
   while(true){
        if (i > n) exit(0);
        sum += i++;
   }
   cout << "The sum of the first " << n
   << " integers is " << sum << endl;
}
```

Output:

What if we use exit(0)?

Sum of Consecutive Integers Revisited

Code:

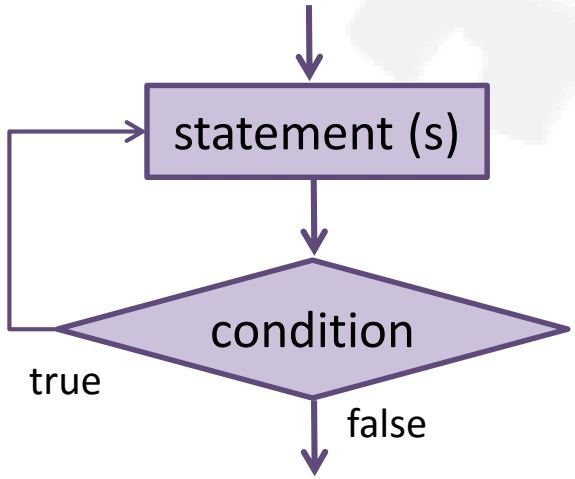
```
int main(){
   int n, i = 1;
   cout << "Enter a positive integer: ";
   cin >> n;
   long sum = 0;
   while(true){
        if (i > n) return 0;
        sum += i++;
   }
   cout << "The sum of the first " << n
   << " integers is " << sum << endl;
}
```

Output:

What if we use return?

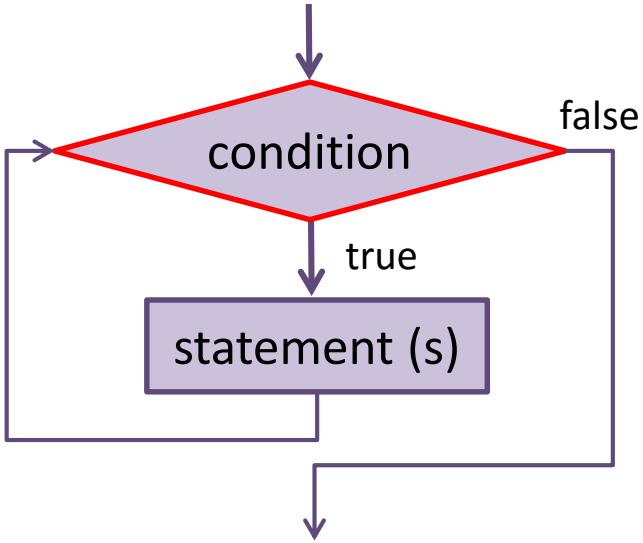
The do..while Statement

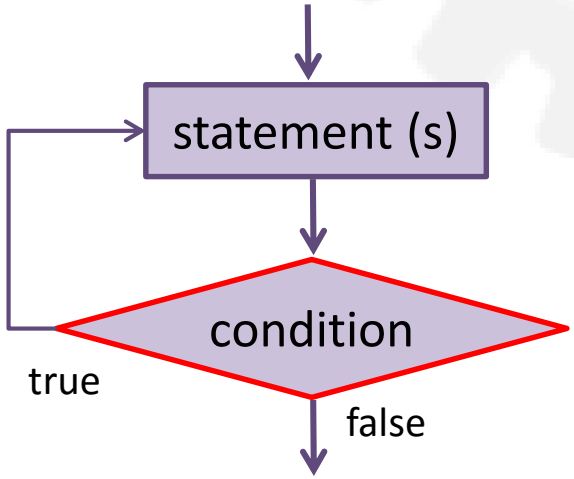
```
do {
    statements;
}while(condition);
```



- First execute the statement, then check the condition
  - If condition is true (non-zero), then execute the statement again until the condition is false (zero)
- Test the condition at the end of the loop

while VS. do..while





- while = check, then execute statements
  - Test the condition at the beginning of the loop
- do..while = execute statements, then check
  - Test the condition at the end of the loop

While vs. Do while #1

While Code:

```
cout << "begin" << endl;

while(false){
  cout << "in loop" << endl;
}

cout << "end" << endl;
```

Do while Code:

```
cout << "begin" << endl;

do{
  cout << "in loop" << endl;
}while(false);

cout << "end" << endl;
```

While Output:

```
begin
end
```

Do while Output:

```
begin
inloop
end
```

While vs. Do while #2

While Code:

```
int i = 0;
cout << "begin" << endl;

while(i < 10){
  cout << i << " " << endl;
  i++;
}

cout << "end " << i << endl;
```

Do while Code:

```
int i = 0;
 cout << "begin" << endl;

 do {
   cout << i << " " << endl;
   i++;
 } while(i < 10);

 cout << "end " << i << endl;
```

While Output:

```
begin
0 1 2 3 4 5 6 7 8 9 end 10
```

Do while Output:

```
begin
0 1 2 3 4 5 6 7 8 9 end 10
```

Sum of Consecutive Integers Revisited#2

Code:

```
int main(){
   int n, i = 1;
   cout << "Enter a positive integer: ";
   cin >> n;
   long sum = 0;
```

What if we use do..while?

```
   cout << "The sum of the first " << n
   << " integers is " << sum << endl;
}
```

Output:

Sum of Consecutive Integers Revisited#2

Code:

```
int main(){
   int n, i = 1;
   cout << "Enter a positive integer: ";
   cin >> n;
   long sum = 0;
   do{
        sum += i;
        i++;
   }while(i <= n);

   cout << "The sum of the first " << n
   << " integers is " << sum << endl;
}
```

Output:

What if we use do..while?

First n Factorial Numbers

Code (do..while):

```
int main(){
   int n;
   cout << "Enter a positive
   integer: ";
   cin >> n;
   cout << "Factorial numbers: ";
   long fac = 1, i = 1;
   do{
```

```
   }while(          );
   cout << endl;
   return 0;
}
```

Code (while):

```
int main(){
   int n;
   cout << "Enter a positive
   integer: ";
   cin >> n;
   cout << "Factorial numbers: ";
   long fac = 1, i = 1;
   while(          ){
```

```
   }
   cout << endl;
   return 0;
}
```

First n Factorial Numbers

Code (do..while):

```
int main(){
   int n;
   cout << "Enter a positive
   integer: ";
   cin >> n;
   cout << "Factorial numbers: ";
   long fac = 1, i = 1;
   do{
        fac *= i;
        i++;
        cout << fac << " ";
   }while( i<=n );
   cout << endl;
   return 0;
}
```

Code (while):

```
int main(){
   int n;
   cout << "Enter a positive
   integer: ";
   cin >> n;
   cout << "Factorial numbers: ";
   long fac = 1, i = 1;
   while( i<=n ){
        fac *= i;
        i++;
        cout << fac << " ";
   }
   cout << endl;
   return 0;
}
```

First n Factorial Numbers within a Bound

Code (do..while):

```
int main(){
   int bound;
   cout << "Enter a positive
   integer: ";
   cin >> bound;
   cout << "Factorial numbers: ";
   long fac = 1, i = 1;
   do{
```

```
   }while(          );
   cout << endl;
   return 0;
}
```

Code (while):

```
int main(){
   int bound;
   cout << "Enter a positive
   integer: ";
   cin >> bound;
   cout << "Factorial numbers: ";
   long fac = 1, i = 1;
   while(          ){
```

```
   }
   cout << endl;
   return 0;
}
```

First n Factorial Numbers within a Bound

Code (do..while):

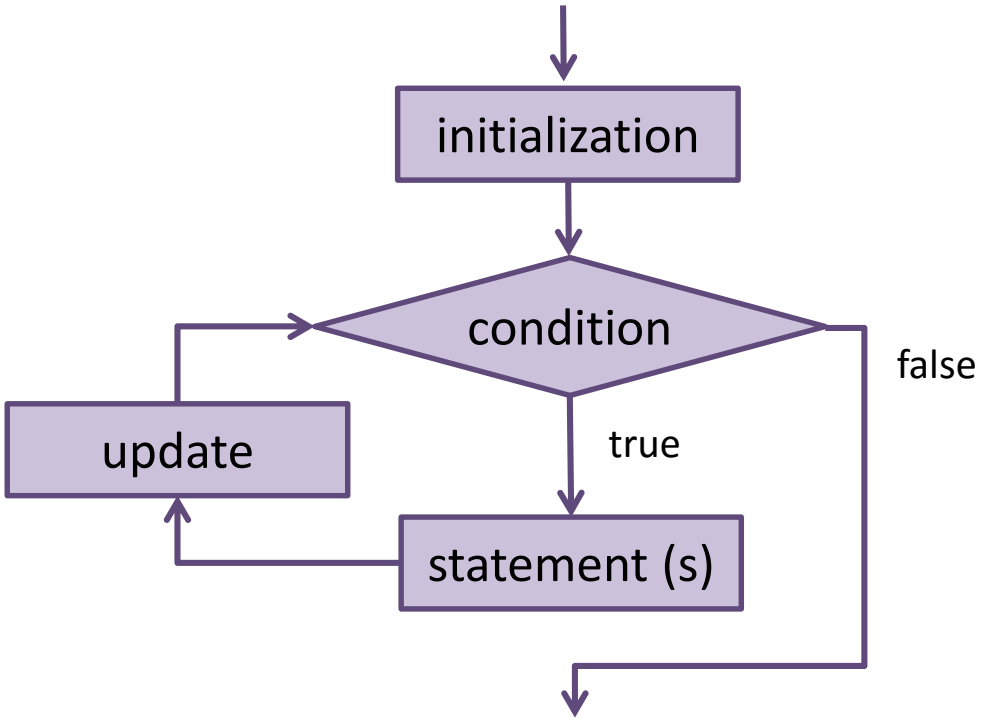
```
int main(){
   int bound;
   cout << "Enter a positive
   integer: ";
   cin >> bound;
   cout << "Factorial numbers: ";
   long fac = 1, i = 1;
   do{
        cout << fac << " ";
        i++;
        fac *= i;
   }while( fac < bound );
   cout << endl;
   return 0;
}
```

Code (while):

```
int main(){
   int bound;
   cout << "Enter a positive
   integer: ";
   cin >> bound;
   cout << "Factorial numbers: ";
   long fac = 1, i = 1;
   while( fac < bound ){
        cout << fac << " ";
        i++;
        fac *= i;
   }
   cout << endl;
   return 0;
}
```

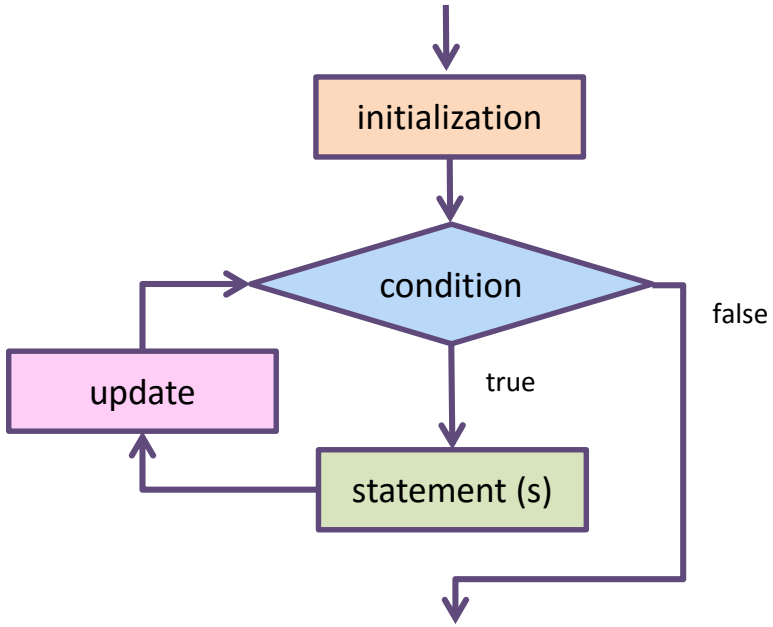
The for Statement

```
for(initialization; condition; update){
    statement;
}
```



What Happens in a for Loop

```
for(initialization; condition; update){
        statement;
    }
```



1. Evaluate initialization
   - Declare/initialize control variable(s) for the loop
2. Evaluate condition
   - Before iteration occurs
3. Execute the statement
4. Execute the update
5. Repeat steps 2-4

for vs. while

Code (for):

```
int main(){
   int n;
   cout << "Enter a positive
   integer: ";
   cin >> n;
   for(int i = 0; i < n; i++){
        cout << (i+1) << " ";
        if( (i+1)%10 == 0)
           cout << endl;
   }
   cout << endl;
   return 0;
}
```

Code (while):

```
int main(){
   int n;
   cout << "Enter a positive
   integer: ";
   cin >> n;
   int i = 0;
   while(i < n){
        cout << (i+1) << " ";
        if( (i+1)%10 == 0)
           cout << endl;
        i++;
   }
   cout << endl;
   return 0;
}
```

Sum of Consecutive Integers Revisited

- Problem: Add consecutive integers from 1 to n
  - 1+2+3+…+n = ?
- You should be able to do this in 2 seconds by now ;)
- Again: what are an input, an output, a decision, and a process?

Sum of Consecutive Integers (for)

Code:

```
int main(){
   int n, sum = 0;
   cout << "Enter a positive integer: ";
   cin >> n;

   cout << "The sum of the first " << n
   << " integers is " << sum << endl;
   return 0;
}
```

Output:

Reusing for Loop Control Variable

Code:

```
int main(){
   int n, sum = 0;
   cout << "Enter a positive integer: ";
   cin >> n;
   long sum = 0;
   for(int i=0; i < n/2; i++){
        sum += i;
   }
   for(int i=n/2; i <= n; i++){
        sum += i;
   }
   cout << "The sum of the first " << n
   << " integers is " << sum << endl;
   return 0;
}
```

Output:

Loop Variables

Code #1:

```
int main() {

    int i = 10;
    cout << "begin " << i <<
    endl;

    for(int i=0; i <= 20; i++){
        cout << i;
    }

    cout << endl;
    cout << "end loop " << i;
    return 0;
}
```

Code #2:

```
int main() {

    int i = 10;
    cout << "begin " << i <<
    endl;

    for (i=0; i <= 20; i++){
        cout << i;
    }

    cout << endl;
    cout << "end loop " << i;
    return 0;
}
```

Factorial Numbers (for)

- Problems:
  - Print the first n factorial numbers
    - 1! 2! 3! … n!
  - Print all factorial numbers within a given bound
    - 1! 2! 3! … (x! < bound)

First n Factorial Numbers

Code:

```
int main(){
   int n;
   cout << "Enter a positive integer: ";
   cin >> n;
   cout << "Factorial numbers: 1 ";
   long fac = 1;
   
   cout << endl;
   return 0;
}
```

Output:

First n Factorial Numbers within a Bound

Code (for):

```
int main(){
   int bound;
   cout << "Enter a positive
   integer: ";
   cin >> bound;
   cout << "Factorial numbers: 1 ";
   long fac = 1;
```

```
   cout << endl;
   return 0;
}
```

Code (while):

```
int main(){
   int bound;
   cout << "Enter a positive
   integer: ";
   cin >> bound;
   cout << "Factorial numbers: 1 ";
   long fac = 1, i = 1;
   while(          ){
```

```
   }
   cout << endl;
   return 0;
}
```

Descending for Loop

- How to print numbers from n to 1?

Code:

```
int main(){
   int n;
   cout << "Enter a positive integer: ";
   cin >> n;
   
   cout << endl;
   return 0;
}
```

Output:

A Prime Number

A natural number greater than 1 that has no positive divisors other than 1 and itself

Write a program to test if a given number is a prime number or not

Is n a prime number?

Code (1/2):

```
int main(){
    int n;
    cout << "Enter a positive integer: ";
    cin >> n;
    if(n < 2){
          cout << n << " is not prime.\n";
    }
    else if(n < 4){
          cout << n << " is prime.\n";
    }
    else if(n%2 == 0){
          cout << n << " = 2 * " << n/2 <<
    endl;
    }
    …
```

Another style of updates

Code (2/2):

```
    else{
      for(int d = 3; d <= n/2; d += 2){
           if(n%d == 0){
              cout << n << " = " << d <<
                   " * " << n/d << endl;
              exit(0);
           }
       }
           cout << n << " is prime.\n";
    }
    return 0;
}
```

Using a Sentinel to Control a for Loop

Code:

```
int main(){
    int n, max;
    cout << "Enter a positive integer
    (0 to quit): ";
    cin >> n;
    for(max = n; n > 0;)
    {
         if(n > max) max = n;
         cin >> n;
    }
    cout << "max = " << max << endl;
    return 0;
}
```

Output:

No need to use just "i" to terminate a for loop (just like while)

Find Min from All Inputs

Code:

```
int main(){
   int n, min;
   cout << "Enter a positive integer (0
   to quit): ";
   cin >> n;
```

```
        cout << "min = " << min << endl;
   return 0;
}
```

Output:

Hint: The last page is to find max

Forever for vs Forever while

Code (for):

```
int main(){
   int n, sum = 0;
   cout << "Enter positive
   integers (0 to quit):"
        << endl;
   for(;;){
      cin >> n;
      if(n <=0) break;
      sum += n;
   }
   cout << "Sum = " << sum
        << endl;
   return 0;
}
```

Code (while):

```
int main(){
   int n, sum = 0;
   cout << "Enter positive
   integers (0 to quit):"
        << endl;
   while(true){
      cin >> n;
      if(n <=0) break;
      sum += n;
   }
   cout << "Sum = " << sum
        << endl;
   return 0;
}
```

Control Input with a Sentinel

Code:

```
int main(){
   int n, count = 0, sum = 0;
   cout << "Enter positive integers (0 to quit): " << endl;
   for(;;){
        cout << "\t" << count + 1 << " : ";
        cin >> n;
        if(n <= 0) break;
        ++count;
        sum += n;
   }
   cout << "The average of those " << count <<
         " positive numbers is " << float(sum)/count << endl;
   return 0;
}
```

Nested for Loop

Code:

```
#include <iomanip>

int main(){
   for(int x = 1; x <= 12; x++){
       for(int y = 1; y <= 12; y++){
           cout << setw(4) << x * y;
       }
       cout << endl;
   }
   return 0;
}
```

Output:

Let's build a filled box

Code:

```
int main(){
   cin >> n;
   for(int x = 1; x <= n; x++){
       for(int y = 1; y <= n; y++){
           cout << '*';
       }
       cout << endl;
   }
   return 0;
}
```

Output:

Let's build an empty box

Code:

```
int main(){
   cin >> n;
   for(int x = 1; x <= n; x++){
       for(int y = 1; y <= n; y++){
          if(x==1 || x==n || y==1
               || y==n)
              cout << '*';
          else
              cout >> ' ';
       }
       cout << endl;
   }
   return 0;
}
```

Output:

The break vs continue Statement

- break;
  - Skips the rest of the statements in the loop's block
  - Jumps immediately to the next statement outside of the loop
- continue;
  - Skips the rest of the statements in the loop's block
  - Does the next iteration of the loop

A loop with break

Code:

```
int main() {
        for (int i = 0; i < 10; i++) {
                cout << "Top half :" << i << endl;
                if (i > 5 ) break;
                cout << "Bottom half:" << i << endl;
        }
        cout << "Outside of loop.";
        return 0;
}
```

Output:

A loop with continue

Code:

```
int main() {
        for (int i = 0; i < 10; i++) {
                cout << "Top half :" << i << endl;
                if (i > 5 ) continue;
                cout << "Bottom half:" << i << endl;
        }
        cout << "Outside of loop.";
        return 0;
}
```

Output:

Be careful when using continue

Code:

```
int main() {
        int i = 0;
        while( i < 10) {
                cout << "Top half :" << i << endl;
                if (i > 5 ) continue;
                cout << "Bottom half:" << i << endl;
                i++;          //  be careful of this
        }
        cout << "Outside of loop.";
        return 0;
}
```

What happen when i>5 ??!!

Output:

A break with Nested Loops

Code:

```
#include <iomanip>

int main(){
   for(int x = 1; x <= 12; x++){
      for(int y = 1; y <= 12; y++){
         if(y > x) break;
         else
            cout << setw(4) << x * y;
      }
      cout << endl;
   }
   return 0;
}
```

Output:

Using a goto Statement

Code:

```
int main(){
   const int N = 5;
   for(int i = 0; i < N; i++){
       for(int j = 0; j < N; j++){
          for(int k = 0; k < N; k++){
              if(i + j + k > N) goto esc;
              else cout << i + j + k << " ";
          } // for k
          cout << "* ";
       } // for j
       esc: cout << "." << endl;
   } // for i
   return 0;
}
```

Output:

Goto make the execution jumps from one point directly to another point

Using a Flag to Break Out

Code:

```
int main(){
   const int N = 5;
   bool done = false; // done is a flag
   for(int i = 0; i < N; i++){
       for(int j = 0; j < N && !done; j++){
           for(int k = 0; k < N && !done; k++)
               if(i + j + k > N) done = true;
               else cout << i + j + k << " ";
           cout << "* ";
       } // for j
       cout << "." << endl;
       done = false; // reset flag
   } // for i
   return 0;
}
```

Output:

!done will be false when we check the condition

Breaking-a-Loop Summary

Breaking a Loop

- break;
- exit(0);
- return 0;
- Using a flag
  - done = true;
- goto somewhere;

Skipping a Loop

- continue;
- goto somewhere;

Take Home Messages

- Loop = execute statement(s) repeatedly until the condition is false
  - Involves counter(s) and updating counter(s)
  - Or a sentinel
- If the condition is always true, terminate the loop by break, exit(0), return, or goto
- Three kinds of commands
  - while
  - do..while
  - for

References

- คณาจารย์คณะวิศวกรรมศาสตร์. (n.d.). **เอกสารชุดนำเสนอภาพและบรรยายวิชาการเขียนโปรแกรม (ส่วนกลาง)**
- คณาจารย์คณะวิศวกรรมศาสตร์. (n.d.). **เอกสารประกอบการสอนวิชาการเขียนโปรแกรม (ส่วนกลาง)**
- รศ. วิโรจน์ ทวีปวรเดช. (2554). **การเขียนโปรแกรมคอมพิวเตอร์ Computer Programming.** พิมพ์ครั้งที่ 2 โรงพิมพ์มหาวิทยาลัยขอนแก่น
- Cplusplus.com. (n.d.). **C++ Documentation**. สืบค้นเมื่อ 18 กุมภาพันธ์ 2555, http://www.cplusplus.com/
- Cplusplus.com. (n.d.). **Library Reference**. สืบค้นเมื่อ 18 กุมภาพันธ์ 2555, http://www.cplusplus.com/
- ISO/IEC 14882 Programming Language – C++

EXTRA SLIDES

Prime Numbers

```
int main() {
        long n;
        cout << "Enter a positive integer: ";
        cin >> n;
        bool is_prime = true;
  if ( n >= 2 ) {                    // 2 is prime need to test here
    for( int i = 2; i < n ; i++) {   // because
      if ( n % i == 0) {             // this statement fail to check 2
        is_prime = false;       // is prime
      }
    }
  }
  if ( is_prime )
    cout << n << " is prime";
  else
    cout << n << " is not prime";
}
```

Prime Numbers (cont'd.)

```
for( int i = 2; i < n ; i++) {
   if ( n % i == 0) {
     is_prime = false;
     break;
   }
}
```

No need to test other numbers because the number can be divided at just one, it is not prime.

Prime Numbers (cont'd.)

- If the number is has divisor by any event numbers it is also has divisor by 2.

- 20 % 4 = 0 => 20 % 2 = 0 is also true.

- Just test if the number is a even number then it is not prime, no need to check event divisors.

Prime Numbers (cont'd.)

```
int main() {
            long n;
            cout << "Enter a positive integer: ";
            cin >> n;
            bool is_prime = true;
   if ( n >= 3 ) {        // 2 and 3 is prime
    if ( n % 2 == 0 ) {
      is_prime = false;
    } else {
      for( int i = 3; i < n ; i += 2) {
        if ( n % i == 0) {
          is_prime = false;
          break;
        }
       }
     }
   }
  if ( is_prime ) cout << n << " is prime";
 else cout << n << " is not prime";
}
```

Prime Numbers (cont'd.)

It is no need to test from 2 to n-1 just testing from 2 to $\sqrt{n}$ is enough, because $\sqrt{n} \times \sqrt{n} = n$, if number larger than $\sqrt{n}$ the result of multiplication will larger than n too.

```
..
long n;
..
cin >> n;
long limit = sqrt(n);
 ...
  for( int i = 3; i < limit ; i += 2)
 ...
```

Prime Numbers (cont'd.)

Using combine exit condition

```
int main() {
        long n;
        cout << "Enter a positive integer: ";
        cin >> n;
  long limit = sqrt(n);
        bool is_prime = true;

  if ( n >= 3 ) {
                if ( n % 2 == 0 ) is_prime = false;
        for( int i = 3; i < limit && is_prime ; i += 2)
        if ( n % i == 0) is_prime = false;
  }

  if ( is_prim ) {
    cout << n << " is prime";
  } else {
    cout << n << " is not prime";
  }
}
```

Several Control Variables

Code:

```
int main(){
   for(int m=95, n=11; m%n > 0; m-=3, n++){
       cout << m << " % " << n << " = " << m % n
            << endl;
   }
   return 0;
}
```

Output: