Loops

What loops are and how they are used
How to build various loops
\textbf{i++, }j+=2

\begin{align*}
&i++; \\
&++i; \\
&i--; \\
&--i; \\
&i+=2; \\
&i-=2;
\end{align*}
\[ \text{i++, j+=2} \]

```c
i++;  
i = i + 1;

++i;  
i = i + 1;

i--;  
i = i - 1;

--i;  
i = i - 1;

i+=2;  
i = i + 2;

i-=2;  
i = i - 2;
```
i++;
i = i + 1;

++i;
i = i + 1;

i--;
i = i - 1;

--i;
i = i - 1;

i+=2;
i = i + 2;

i-=2;
i = i - 2;
cout << i++;

cout << ++i;

cout << i--;

cout << --i;

cout << (i+=2);

cout << (i-=2);
cout << i++;  
cout << i;   i = i + 1;

cout << ++i;
i = i + 1; cout << i;

cout << i--;  
cout << i;   i = i - 1;

cout << --i;
i = i - 1; cout << i;

cout << (i+=2);
i = i + 2; cout << i;

cout << (i-=2);
i = i - 2; cout << i;
Loops

What loops are and how they are used
How to build various loops
Computers Excel @

- Doing same thing over and over again
- Software never gets tired
What's a loop?

- Code that may be executed more than once in a row
- Either a fixed number of times
- Or until a specific condition is met
- Each pass through the loop is an *iteration*
While Loop

- Syntax
  
  ```
  while (condition)
  statement;
  ```

- If (condition) is **true**
  - Execute statement inside loop body

- Repeat, until (condition) evaluates to **false**
Body of while loop

while (<condition>) <statement>

- Can be a **single** simple statement
  
  while (<condition>)  
  
  - Can be a **single** compound statement (containing multiple statements)
    
    while (<condition>) {int t=x; x=y; y=t;}
    while (<condition>) if (b) x = y; else y = x;

- Can be a **single** empty statement
  
  while (<condition>) ;
  while (<condition>) {}
Same applies to body of if

- Can be a single simple statement

```plaintext
if (<condition>) s+=i;
```

- Can be a single compound statement (containing multiple statements)

```plaintext
if (<condition>) {int t=x; x=y; y=t;}
if (<condition>) if (b) x = y; else y = x;
```

- Can be a single empty statement

```plaintext
if (<condition>) ;
if (<condition>) {}
Can be a **single** simple statement

if (<condition>) <statement1> else s+=i;

Can be a **single** compound statement (containing multiple statements)

if (<condition>) <statement1> else {int t=x; x=y; y=t;}
if (<condition>) <statement1> else if (b) x = y; else y = x;

Can be a **single** empty statement

if (<condition>) <statement1> else ;
if (<condition>) <statement1> else {}
While Loop

condition

true

statements

false

EXIT
Examples of:

- Count Controlled Loops
Example 1.1: What prints?

```cpp
int number = 1;
while ( number <= 5 )
{
    cout << “Hello”;
    number++;  
}

cout << number;
```
Example 1.1: What prints?

```cpp
int number = 1;

while ( number <= 5 )
{
    cout << "Hello";
    number++;
}

cout << number;
```
Example 1.1: What prints?

```cpp
int number = 1; // initialize loop control variable
while ( number <= 5 ) // check loop condition
{
    cout << "Hello";
    number++; // update loop control variable
}

cout << number;
```
int number = 1; // initialization

while ( number <= 5 ) // condition
{
    cout << "Hello";
    number++; // update
}

cout << number;
int number = 1;

while ( number <= 5 )
{
    cout << “Hello”; 
    number++;
}

cout << number;
int number = 6;

while ( number <= 5 )
{
    cout << “Hello”; 
    number++; 
}

cout << number;
int number = 1;

while ( number <= 5 )
{
    cout << "Hello";
}

cout << number;
Example 1.4: What prints?

```cpp
int number = 1;
while ( number <= 100 )
{
    cout << "Hello";
    number++;
}

cout << number;
```
Example 1.5: What prints?

```cpp
int number = 1;
while ( number < 100 )
{
    cout << "Hello";
    number++;
}
cout << number;
```
Example 1.6: What prints?

```cpp
int number = 1;
while ( number <= 100 )
    cout << "Hello";
    number++;

cout << number;
```
Example 1.7: What prints?

```cpp
int number = 1;
while ( number <= 100 )
{
    cout << "Hello";
    number++;
}
cout << number;
```
Example 2.1: What prints?

```cpp
int x = 1;

while ( x <= 5 )
{
    cout << x << endl;
    x++;
}

```

```cpp
cout << x << endl;
```

Example 2.1: What prints?

```cpp
int x = 1;
while ( x <= 5 )
{
    cout << x << endl;
    x++;
}
```

Example 2.1: What prints?

```cpp
int x = 1; // initialize loop control variable

while ( x <= 5 ) // check loop condition
{
    cout << x << endl; // update loop control variable
    x++;}
```

```cpp
    cout << x << endl;
```
Example 2.1: What prints?

```cpp
int x = 1; // initialization

while ( x <= 5 ) // condition
{
    cout << x << endl;
    x++; // update
}

cout << x << endl;
```
Example 2.2: What prints?

```cpp
int x = 6;

while ( x <= 5 )
{
    cout << x << endl;
    x++;
}
```

cout << x << endl;
Example 2.3: What prints?

```cpp
int x = 1;
while ( x <= 5 )
{
    cout << x << endl;
}
```
int x = 1;

while ( x <= 100 )
{
    cout << x << endl;
    x++;
}

cout << x << endl;
Example 2.5: What prints?

```cpp
int x = 1;

while ( x < 100 )
{
    cout << x << endl;
    x++;
}
```

```cpp
cout << x << endl;
```
Example 2.6: What prints?

```cpp
int x = 1;
while ( x <= 100 )
    cout << x << endl;
x++;
```
int x = 1;

while ( x <= 100 )
{
    cout << x << endl;
    x++;
}

cout << x << endl;
Example 2.4: What prints?

```cpp
int x = 1;

while ( x <= 100 )
{
    cout << x << endl;
    x++;
}
```
int x = 100;

while ( x > 0 )
{
    cout << x << endl;
    x--;
}


int x = 100;

while ( x > 0 )
{
    cout << x << endl;
    x--;
}

cout << x << endl;;
int x = 100;

while ( x > 0 )
{
    cout << x << endl;
    x--;
}
Example 2.8: What prints?

```cpp
int x = 100;
while ( x > 0 )
{
    cout << x << endl;
    x--;
}
```

int x = 100;

while ( x >= 0 )
{
    cout << x << endl;
    x--;
}

Example 2.9: What prints?
Example 2.8: What prints?

```
int x = 100;
while ( x > 0 )
{
    cout << x << endl;
    x--;
}
```
`int x = 100;
while ( x > 0 )
{
    x--;
    cout << x << endl;
}
`
Example 2.4: What prints?

```cpp
int x = 1;
while ( x <= 100 )
{
    cout << x << endl;
    x++;
}
```

Example 2.11: What prints?

int x = 1;

while ( x <= 100 )
{
    if ( x % 13 == 0 )
    {
        cout << x << endl;
    }
    x++;
}
cout << x << endl;
int year;
cin >> year;

while ( year <= 0 )
{
    cout << "Year must be greater than 0"
    << endl
    << "Try again.."
    << endl;

    cin >> year;
}
Example 3.1: Not count controlled

```cpp
int year;
cin >> year;

while ( year <= 0 )
{
    cout << "Year must be greater than 0"
         << endl
         << "Try again.."
         << endl;
    cin >> year;
}
```

```cpp
}
Example 3.1: Not count controlled

```cpp
int year;

// initialize loop control variable
cin >> year;

// check
while ( year <= 0 )
{
    cout << "Year must be greater than 0"
        << endl
        << "Try again.."
        << endl;

    // update
    cin >> year;
}
```
Example 3.1: Not count controlled

```cpp
int year;
cin >> year;

while ( year <= 0 )
{
    cout << "Year must be greater than 0"
    << endl
    << "Try again.."
    << endl;

    cin >> year;
}
```
Example 3.2

int year;

// initialize loop control variable
cin >> year;

while ( cin >> year && year <= 0 )
{
    cout << "Year must be greater than 0"
    << endl
    << "Try again.."
    << endl;

    cin >> year;
}
Example 4: Average Salary

- Say, don't know how many employees apriori
  - Then loop can't be "count" controlled

- Then can stop loop by entering negative salary
```cpp
int count = 0;
double total = 0;
double salary;

while ( salary >= 0 ) {
    total += salary;
    count++;
    cin >> salary;
}

if ( count > 0 ) {
    cout << "Average is: " << total / count;
} else {
    cout << "No data" << endl;
}
```
Example 4.1: Average Salary

```cpp
int count = 0;
double total = 0;
double salary;

// initialization
cin >> salary;

while ( salary >= 0 ) {  // check
    total += salary;
    count++;
    cin >> salary;  // update
}

if ( count > 0 ) {
    cout << "Average is: " << total / count;
} else {
    cout << "No data" << endl;
}
```
```cpp
int count = 0;
double total = 0;
double salary;
cin >> salary;

while ( cin >> salary && salary >= 0 ) {
    total += salary;
    count++;
    cin >> salary;
}

if ( count > 0 ) {
    cout << "Average is: " << total / count;
} else {
    cout << "No data" << endl;
}
```
Example 5: Average number

- Say, don't know how many numbers apriori
- Also say, numbers can be negative
- Then, can stop loop by entering a non-number
Example 5: Average number

int count = 0;
double total = 0;
double number;

while ( cin >> number ) {
    total += number;
    count++;
}

if ( count > 0 ) {
    cout << "Average is: " << total / count;
} else {
    cout << "No data" << endl;
}
```c++
int count = 0;
double total = 0;
double number;

while ( cin >> number ) {
    total += number;
    count++;
}

if ( count > 0 ) {
    cout << "Average is: " << total / count;
} else {
    cout << "No data" << endl;
}
```

alpha will put "cin" into a "fail" state
need to clear before any further input takes place

`cin.clear();`
int sum(int n) {
    int sum = 0;
    int i = 1;
    while (i <= n) {
        sum = sum + i;
        i = i + 1;
    }
    return sum;
}
Sum of first n positive integers

// Requires:
// Effects:

int sum(int n)
{

}
// Requires: n > 0
// Effects:

int sum(int n)
{

}
// Requires: \( n > 0 \)
// Effects: Returns the sum of the first \( n \) integers
// That is, returns \( 1 + 2 + \ldots + n \)
int sum(int n)
{

}
// Requires: \( n > 0 \)
// Effects: Returns the sum of the first \( n \) integers
// That is, returns \( 1 + 2 + \ldots + n \)
int sum(int n)
{
    int sum = 0;
    int i = 1;
    while (i <= n)
    {
        sum = sum + i;
        i = i + 1;
    }
    return sum;
}
/\textbf{Sum of first }n\textbf{ positive integers}\n
// Requires: $n > 0$
// Effects: Returns the sum of the first $n$ integers
// That is, returns $1 + 2 + \ldots + n$
int sum(int n)
{

}

while ( )
{

}

}
// Requires: \( n > 0 \)
// Effects: Returns the sum of the first \( n \) integers
// That is, returns \( 1 + 2 + \ldots + n \)
int sum(int n)
{
    int i = 1;
    while ( i <= n )
    {
        i = i + 1;
    }
    return sum;
}
Sum of first n positive integers

// Requires: n > 0
// Effects: Returns the sum of the first n integers
//          That is, returns 1 + 2 + ... + n
int sum(int n)
{
    int s = 0;
    int i = 1;
    while ( i <= n )
    {
        s = s + i;
        i = i + 1;
    }
    return s;
}
// Requires: n > 0
// Effects: Returns the sum of the first n integers
// That is, returns 1 + 2 + ... + n
int sum(int n)
{
    int s = 0;
    int i = 1;
    while ( i <= n )
    {
        s = s + i;
        i = i + 1;
    }
    return s;
}
// Requires: n > 0
// Effects: Returns the sum of the first n integers
//          That is, returns 1 + 2 + … + n
int sum(int n)
{
    int s = 0;
    int i = 1;
    while ( i <= n )
    {
        s = s + i;
        i = i + 1;
    }
    return s;
}
Summary

- **while loops**
- Examples