Alternatives to “linear execution”

Repeated actions

> ./myProgram
Hello world.
Hello world.
Hello world.
Hello world.

The while loop

while ( condition )
    statement_to_repeat;

OR

while (condition)
{
    statement_to_repeat1;
    ...  
    statement_to_repeatN;
}

condition – a Boolean expression

Just a reminder from our earlier if-else slides

- Boolean expressions are either true or false
- Conditions often consist of comparisons
  - age \( \geq 21 \) // can buy drinks
  - age < 4 // can ride subway for free
  - year = 2 // you are a sophomore

How can we output “Hello world” 4 times?

```c
int x=4;
while ( x>0 )
{
    cout << "Hello world.\n";
    x--;
}
```

Remember x--; same as x=x-1;
Repeats until x\( \leq 0 \)

Execution of while loop

- If condition is true, enter while loop
- Complete all statements in block
- Return to top (re-evaluate condition)
- Otherwise, continue to statements beyond loop
Execution of while loop

- If **condition is true**, enter while loop
- Complete all statements in block
- Return to top (re-evaluate **condition**)
- Otherwise, continue to statements beyond loop

```c
int x=2;
while ( x>0 ) {
    x--;
    cout << "Hello world.\n";
}
```

**How many “Hello world”s are output?**

What code will do this for us?

```c
> ./myProgram
1 mississippi
2 mississippi
3 mississippi
4 mississippi
5 mississippi
>
```

**a++ vs. ++a**

- `a++` returns value of `a`, then adds 1 to `a`
- `++a` adds 1 to `a`, then returns value of `a`

Different results for:

```c
int a=0;    int a=0;
while (a++ < 3) while (++a < 3)
cout << "Hi!\n";    cout << "Hi!\n";
```

Review of Friday program

```c
Give me a number: 5    cout << "Give num: ";
cin >> num;    cin >> num;
100/5 is 20    while (num!=0) {
    cout<<"100/"<<num
100/33 is 3.333    << " is "<<100.0/num

Cannot divide by 0!    << endl;
            cout << "Give num: ";
cin >> num;
}
```

Modification of Friday program

```c
Give me a number: 5 do {
    cout << "Give num: ";
doin >> num;
100/5 is 20    if(num!=0) {
    cout<<"100/"<<num
100/33 is 3.333    << " is "<<100.0/num

Cannot divide by 0!    << endl;
    cout << "Give num: ";
cin >> num;
}   cin >> num;
} while(num!=0);
```
What does this code do?

```cpp
int main () {
    int a=5;
    do {
        cout << "one ";
        a-=2;
        cout << "two\n";
    } while ( a > 0);
    return 0;
}
```

What does this code do?

```cpp
int main () {
    int a=5;
    do {
        cout << "one ";
        a-=2;
        cout << "two\n";
    } while ( a != 0);
    return 0;
}
```

Beware infinite loops!

- Loops that never stop are called infinite loops
- Typically, write code so each loop will stop

Example in scope:

```cpp
int a=2, b=4;
while(a<10) {
    int b = a*5;
    cout << b << endl;
    a+=3;
}
cout << b << endl;
```

Variables – locations in memory

- Each variable indicates a location in memory
- Each location holds a value
- Value can change as program progresses
- Variable value exists before initialization

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>P</td>
<td>32</td>
</tr>
</tbody>
</table>

for loop

```cpp
for ( init; condition; update )
{
    statement;
    . . .
    statement N;
}
```

typical example:

```cpp
int i;
for ( i=1; i<5; i++ )
{
    cout << i << " banana ";
}
```
Reviewing scope

Counter \( i \) exists outside of loop

```cpp
int i;
for ( i=1; i<=5; i++)
{
    cout << i << " banana ";
}
```

Counter \( i \) exists only inside of loop

```cpp
for ( int i=1; i<=5; i++)
{
    cout << i << " banana ";
}
```

What does this code do?

```cpp
int main () {
    int i, product=1;
    for ( i=1; i<=5; i++ )
    {
        product = product*i;
        cout << i << "! = " << product << endl;
    }
    return 0;
}
```

Beware the misplaced ;

Placing a semicolon after the parentheses of a for loop causes an empty statement as the body of the loop

Picking a loop

- **do-while** if you need to perform the action at least once
- **for** if there is a standard repeated mathematical update to your loop variable (e.g., count++)
- **while** loop for less-standard loop variable updates

"loop variable" is the variable tested by the condition in your given loop

init – initializes variable

condition – statement about variable, must stay true for loop to keep running

update – updates the variable after each loop execution