Alternatives to “linear execution”

Repeated actions

> ./myProgram
Infinite bottles of beer. Take one down.
Infinite bottles of beer. Take one down.
Infinite bottles of beer. Take one down.
Infinite bottles of beer. Take one down.

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 ≥ 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?

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

Remember x--; same as x=x-1;
Repeats until x≤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

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

What code will do this for us?

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

What does this code do?

```cpp
int main () {
    int a=5;
    do {
        cout << "one ";
        a--;
        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
for loop
a while loop alternative

for ( init; condition; update )
{
    statement1;
    . . .
    statement N;
}

typical example:
int i, product=1;
for ( i=1; i<=5; i++)
{
    product = product*i;
}

init – initializes variable

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

update – updates the variable after each loop execution

Reviewing scope
Counter i exists outside of loop
int i, product=1;
for ( i=1; i<=5; i++)
{
    product = product*i;
}

Counter i exists only inside of loop
int product=1;
for ( int i=1; i<=5; i++)
{
    product = product*i;
}

What does this code do?
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