while and do-while Loops

CSC 116 – Section 002
February 14, 2005

Repetition Statements

- “control a block of code to be executed for a fixed number of times or until a certain condition is met” [Wu]
- You want to execute one piece of code multiple times

Increment/Decrement Operators

- Increments or decrements an integer variable by 1
- Increment Operator: Double plus (++)
- Decrement Operator: Double minus (--)
- Ex:

  i++ is the same as i = i + 1;

  i-- is the same as i = i – 1;

Post Increment Operator

- Ex:

  i = 1; //Set i to 1

  j = i++; //Set i to 2 and j to 1
- Assigns the value of i to j before incrementing i.

Pre Increment Operator

- Ex:

  i = 1; //Set i to 1
  j = ++i; //Set i to 2 and j to 2
- Adds 1 to i before assigning the value of i to j.

Shorthand Assignment Operators

Operator	Usage	Meaning
+=	a += b;	a = a + b;
-=	a -= b;	a = a – b;
*=	a *= b;	a = a * b;
/=	a /= b;	a = a / b;
%=	a %= b;	a = a % b;

Shorthand Assignment Operators

- Have lower precedence that other operators
- Ex:

  sum *= a + b;

  is the same as

  sum = sum * (a + b);

while Loop

- Executes a piece of code as long as a certain condition is true
- Pretest Loop: The test of the boolean expression is done before the loop body is executed
- Syntax:

  while (*<Boolean expression>*) {

  <loop body - statements>

  }

while Loop (2)

- Ex: Add all numbers from 1 to 100

```
int sum = 0;
int num = 1;
while(num <= 100) {
    sum += num;
    num++;
}
```


Types of Loops

- Count-Controlled Loop: “the loop body is executed for a fixed number of times” [Wu]
- Sentinel-Controlled Loop: “the loop body is executed repeatedly until any one of the designated values, called a sentinel, is encountered.” [Wu]

Priming Read

- Sometimes a variable that is used in the Boolean expression of the loop needs to be primed before the loop is executed
- You'll get an error if a variable in a loop doesn't have a value

Problems with Using Loops

- Infinite Loop – loop never stops executing!
- Ex:

```
int i = 0;
while(i < 100000) {
    i *= 5;
}
```

- Make sure the Boolean expression will eventually be false!

Problems with Using Loops (2)

- Overflow Errors: “occur if you attempt to assign a value larger than the maximum value the variable can hold” [Wu]
- In other programming language overflows crash the program
- In Java, overflows will assign a value of infinity to the variable. No abnormal termination will occur.
  - Overflows only occur with floats and double
  - Ints wrap around from positive to negative

Problems with Using Loops (3)

- Imprecise Loop Counter – don’t use doubles as a counter
- Remember doubles may loose precision!

Problems with Using Loops (4)

- Off-by-one Error
- Ex: We want to execute a loop 5 times

```
int count = 1;
while(count < 5) {
    …
    count++;
}
```


do-while Loop

- Executes a piece of code as long as a certain condition is true
- Posttest Loop: The loop body is executed at least once.
- Syntax:

```
do {
    <loop body – statements>
} while( <Boolean expression> );
```

do-while Loop (2)

- Ex:

```
int age = 0;
do {
    age = userInput();
} while(age >= 21);
```


Loop-and-a-Half Repetition Control

- Used to make a decision in the middle of a loop
- Only the top “half” of the loop is executed during the last repetition
- Used to reduce redundant code
- Use *break* keyword to exit loop block

Loop-and-a-Half Repetition Control (2)

- Ex: Regular while loop

```
int age;
age = userInput();
while(age >= 21) {
    age = userInput();
}
```


Loop-and-a-Half Repetition Control (3)

- Ex: Loop-and-a-half

```
int age;
while(true) {
    age = userInput();
    if(age < 21)
        break;
}
```

References

- Jason Schwarz's Lecture 11 slides: http://courses.ncsu.edu/csc116/