The while loop

CMSC 104 Section 02
March 6, 2024

Administrative Notes

Classwork 4 due tonight

Homework 4 due next Monday

Classwork 5 starts today; due next Wednesday

Quiz 2 next Wednesday

- 20 questions; T/F-MC; short answer
- In class, on Blackboard

Iteration - repeating code

A repetition structure allows the programmer to specify that an action is to be repeated while some condition remains true.

- The condition be a single expression or several expressions joined with an and && statement or an or ||statement(s).

There are three repetition structures in C:

1 `while`: runs for zero or many times

2 `for`: runs for a predetermined number of times

3 `do-while`: runs for one or many times

The `while` loop

```
while( condition )
{
statement(s)
}
```

The condition is always a Boolean - it must evaluate to either `true` or `false`

The braces are not required if the loop body contains only a single statement. However, as with if statements, they are a good idea and are required by the CMSC 104 C Coding Standards.

A simple example

```
while( children > 0 )

{

children = children - 1;

cookies = cookies * 2;

}
```

Good Programming Practice

Always place braces around the body of a while loop.

Advantages:

- Easier to read
- You won't forget to add the braces if you have to add statements to the loop body
- Reduces the chance of a semantic error

Indent the body of a while loop by 3 or 4 spaces – be consistent!

A common use for while loops: getting valid input from the user

```
#include <stdio.h>
int main()
{
    int number;
    printf("Enter a positive number: ");
    scanf("%d", &number);
    while( number < 0 )
    { printf("\nThat's incorrect. Try again.\n")
      printf("Enter a positive number: ");
      scanf("%d", &input);
    }
    printf("You entered: %d\n", number);
    return 0; }
```

A slightly more complex example

Problem: Write a program which calculates the average exam grade for a class of ten students.

What are the program inputs?

- the exam grades

What are the program outputs?

- the average exam grade

The C code

```
#include <stdio.h>
int main() {
int counter = 0, grade, total = 0;
float average;
printf(“Enter the number of students: “);
scanf(“%d”, &students);
While (counter <= students) {
      printf("Enter grade: ");
      scanf("%d", &grade);
      total = total + grade;
      counter = counter + 1;}
// to get a proper floating point average, we have to convert the number of students to a float
average = total/float(students);
printf(“Class average: %1.2f\n”, average);
return 0;
}
```

Using a Sentinel Value

- Instead of asking the user for the number of students, we could allow the user to continually enter grades, using a special value to indicate that this task is finished.
- This special value is called the sentinel value.
- We have to make sure the value chosen for the sentinel value isn't a legal value. For example, we can't use 0 as the sentinel value because it's possible that a student earned a zero on the exam.

The Priming Read

- When we use a sentinel value to control a while loop, we have to get the first value from the user before we encounter the loop so that it will be tested and the loop can be entered.
- This is known as the priming read.
- We have to give significant thought to the initialization of variables, the sentinel value, and getting into the loop.

Final C Code

```c
#include <stdio.h>
int main() {
int counter = 0, grade, total = 0;
float average;
printf("Enter grade: ");
scanf("%d", &grade);


// Loop - while grade values aren't negative
while (grade >= 0) {
  total = total + grade;
  counter = counter + 1;
  printf("Enter a grade: ");
  scanf ("%d", &grade);
  }
if (counter == 0) {// avoid dividing by zero error
  printf("No grades entered. \n"):
  } else {
  average = total/float(counter);
  printf("Class average for %d students: %1.2f\n", counter, average);
  }
  return 0;
}
```