CSE 220 – C Programming

Loops

Control of Flow

- Selection statements:
  - Select a particular path of execution
- Iteration statements:
  - Repeat a particular fragment
- Jump statements:
  - Jump to another place in the code

While Statement

```
while ( expression ) statement

while ( expression ) { statements }
```

- () required around expression
- expression: controlling expression
- Statement/statements: loop body

```
int distance = 1;
while (distance < 10)
      distance += 2;
```

Note: When to use {}

- You should always use curly brackets ("{" "}") to denote a single or compound statement.
- Even though the C compiler will allow you to omit the braces in some circumstances, you should add them anyways.

```
int distance = 1;
while (distance < 10)
      distance += 2;

int distance = 1;
while (distance < 10) {
      distance += 2;
}
```

Example 1

```
char answer = ‘y’;
int value = 0;
while (answer != ‘n’) {
      value += 10;
      printf(“Increment again?\n”);
      scanf(“%c”, &answer);
}
```

Example 2

```
int i = 3;
while (i > 0) {
      printf("Still positive: (%d). Decrement!\n", i);
      i--;
}
```

Still positive: (3). Decrement!
Still positive: (2). Decrement!
Still positive: (1). Decrement!

Example 3

```
int i = 3;
while (i > 0) {
      printf("Still positive: (%d). Decrement!\n", i--);
}
```

Still positive: (3). Decrement!
Still positive: (2). Decrement!
Still positive: (1). Decrement!

Infinite Loop

```
int i = 3;
while (i > 0) {
      printf("Still positive: (%d). Decrement!\n", i);
      /* i--;  */
}
```

Still positive: (3). Decrement!
Still positive: (3). Decrement!
… … …
Still positive: (3). Decrement!

Infinite Loop

- If the controlling expression is always nonzero, the while statement won't terminate
- Use break or return to exit the loop

```
int count = 0;
while (1) {
    count++;
    printf("Iteration #%d\n", count);
    if (count == 5)
        break;
}
```

- What if we initialize count to 7?

Exercise

Is thw below an infinite loop?

```
int j = 15;
while (j > 10) {
      printf("%d\n", j);
      j--;
}
```

- Yes
- No
- Depends
- Right Answer

Exercise

Is thw below an infinite loop?

```
int j = 5;
while (j > 10) {
      printf("%d\n", j);
      j--;
}
```

- Yes
- No
- Depends
- Right Answer

Exercise

- Write a program that reads a sentence from user input and counts the number of occurrences of letters ‘a’ and ‘A’. The sentence ends when the user enters ‘.’

```
Initialize count to 0
Repeat:
    read one letter
    if letter is ‘.’:                exit
    if letter is ‘a’ or ‘A’:         increment
count
```

Solution

```
char letter;
int count = 0;
while (1) {
      scanf(“%c”, &letter);
      if (letter == ‘a’ || letter == ‘A’) {
          count++;
      }
      if (letter == ‘.’)
           break;
}
printf(“You entered %d a’s and A’s\n”,
count);
```

Do Statement

```
do  statement while (expression);

do  {statement; } while (expression);
```

- () required around expression
- expression: controlling expression
- Statement/statements: loop body
- Similar to while except: the controlling expression is executed after the loop body is executed

While / Do comparison

```
int count = 0;
while (count < 5) {
    printf("%d\t", count++);
}
```

0 1 2 3 4

```
int count = 0;
do {
    printf("%d\t", count++);
} while  (count < 5);
```

0 1 2 3 4

While / Do comparison

```
int count = 10;
while (count < 5) {
    printf("%d\t", count++);
}
```

Nothing is printed
Value of count: 10

```
int count = 10;
do {
    printf("%d\t", count++);
} while  (count < 5);
```

10
Value of count: 11

Example 1 Revisited

```
char answer = ‘y’;
int value = 0;
while (answer != ‘n’) {
      value += 10;
      printf(“Increment?\n”);
      scanf(“%c”, &answer);
}
```

```
char answer;
int value = 0;
do {
      value += 10;
      printf(“Increment?\n”);
      scanf(“%c”, &answer);
} while (answer != ‘n’)
```

Do while loops are rare

- I will never ask you to "do" them.

For statement

```
for (expr1;  expr2;  expr3 ) statement

for (expr1;  expr2;  expr3 ) { statements }
```

```
for (i = 0;  i <10;  i++ ) {
      printf("count at %d\n");
}
```

Evaluate expr1: *initialize i to 0*
Test expr2: *is i < 10?*
If true:
    Execute loop body: *printf("…")*
    Execute expr3: *i++*
If false: exit loop

For statement

- Suitable use: when we have a counting variable
- Common usages:

```
for (i =0; i<n; i++) {…}
    //Count from 0 to n-1
for (i =1; i<=n; i++) {…}
    //Count from 1 to n
for (i =n; i>0; i--) {…}
    //Count down from n to 1
for (i =n-1; i>=0; i--) {…}
    //Count from n-1 to 0
```

Omitting expressions

```
for (i =0; i<n; i++) {…}
```

```
int i = 0;
for (; i<n; i++) {…}
```

```
for (i=0; i<n;)
{
      …
      i++;
}
```

- Can omit all 3 expressions:

```
for (;;) {…}
```

- If second expression is omitted: <u>condition is true</u>
  - the loop does not terminate (unless stopped in the body)

Omitting parts of for loop are rare

- I will never ask you to do them.

Comma Operator

```
expr1, expr2
```

- First expr1 is evaluated, then expr2
- The value of entire expression is the <u>last expression</u> evaluated

```
i=0, j=100                i=0,j++,k=i+j
```

- Use when C requires a single expression, but we need to have multiple expressions

Comma Operators are rare

- I will never ask you to use them.

Multiple expressions

- Need to execute multiple expressions:
  - Initialize multiple variables in expr1
  - Increment multiple variable in expr3

```
for ( (i=50, j=100);  i+j>0;  (i+=5, j-=15) ) {

        …

}
```

Multiple Expressions are rare

- I will never ask you to do them.

Declare and Initialize

- Can declare a variable inside expr1
- The variable is visible only inside loop

```
for (int i=0;  i<3; i++) {
    printf("i is %d\n", i);
}
printf("Now i is: %d\n", i);
```

Wrong!
Variable i is not available outside the loop

Exiting from a loop

- While and for: may exit before body executed
- Do: exit after body is executed at least once
- Exit in the middle or transfer control?
  - break statement
  - continue statement
  - goto

Break

```
break;
```

- Used to jump out of: while, do, for (and switch)

```
int sum = 0;
for (;;) {
    printf("Enter a number\n");
    scanf("%d", n);
    sum += n;
    if (n == 100)
        break;
}
```

Continue

```
continue;
```

- Used to skip the remainder of the current iteration and go to the next one:

```
int sum = 0;
for (i=10; i<111;i++) {
    if (i%3 == 0)
            continue;
    sum += i;
    printf("Added %d\n", i);
}
```

goto

```
goto line-label;
line-label: …
```

- Used to jump to any other statement (with a label) in the function
- Rarely used

goto

```
int sum = 0;
while (1) {
    printf("Enter a number\n");
    scanf("%d", n);
    if (n == 100)
        goto done_reading;
    sum += n;
}
done_reading: printf("Sum is %d\n", sum);
```

Goto's are considered dangerous and are rarely used

- I will never ask you to do them.

Null statement

;

- Used mainly for writing loops with an empty loop body

```
for (d = 2; d < n; d +=5);
printf("d is %d\n", d);
```

```
while( i < 0);
{ i++;}
```

```
if (d == 20);
     printf("d is %d\n", d);
```

Null statements are unnecessary is you always use opening and closing braces

- I will never ask you to use them.

Exercise

- Write a program that given n between 1 and 26, prints the following:

A1 A2 A3 … An

```
int n;
printf("Enter n between 1 and 26:\n");
scanf("%d", &n);
for (int index = 1; index <=n;
index++){
    printf("A%d ", index);
}
```

Exercise

- Write a program that given n between 1 and 26, and a character X prints the following:

X1 X2 X3 … Xn

```
int n;
char c;
printf("Enter a number and a letter:\n");
scanf("%d %c", &n, &c);
for (int index = 1; index <=n; index++){
     printf("%c%d ", c, index);
}
```

Exercise

- Write a program that given n between 1 and 26, prints the following table:

A1 A2 A3 … An

B1 B2 B3 … Bn

….

?1 ?2 ?3 … ?n

where '?' is the nth letter of the alphabet.

<u>n = 2</u>

A1 A2

B1 B2

<u>n = 3</u>

A1 A2 A3

B1 B2 B3

C1 C2 C3

Solution – main idea

Read number n from user;

Validate: make sure n between 1 and 26

If n >= 1: print row ‘1’ for A1 to An

If n >= 2: print row ‘2’ for B1 to Bn, else exit

… … …

If n >= idx: print row ‘idx’ for $R_{idx}1$ to $R_{idx}n$, else exit

… … …

Print row n for ?1 to ?n, exit

Solution – main idea

Read number n from user;

Validate: make sure number is between 1 and 26

```
int idx;
for (idx = 1; idx <=n; idx++) {
     print row idx;
}
```

Solution – main idea

Read number n from user;

Validate

```
int idx;
for (idx = 1; idx <=n; idx++) {
    //print row idx;
  print "?1 "
  print "?2 "
  ....
  print "?n \n"
}
```

Solution – main idea

Read number n from user;

Validate

```
int idx;
for (idx = 1; idx <=n; idx++) {
    for (idx2 = 1; idx2 <= n; idx2++) {
      print “?idx2 ”  //Should know replacement for ‘?’
    }
}
```

Solution – main idea

Read number n from user;

Validate

```
int idx; char letter = ‘A’;
for (idx = 1; idx <=n; idx++) {
    for (idx2 = 1; idx2 <= n; idx2++) {
       print letter idx2
    }
    letter++;
}
```

Solution – main idea

Read number n from user;

Validate

```
int idx; char letter = ‘A’;
for (idx = 1; idx <=n; idx++,letter++) {
    for (idx2 = 1; idx2 <= n; idx++) {
      print letter idx
    }
    //letter++;
}
```

Solution

```
int n, idx, idx2;
char letter = ‘A’;

//Read input
printf(“Enter n:\n”);
scanf(“%d”, &n);

//Validate
if (n < 1 || n > 26) {
    printf(“Not valid.\n”);
    return 1;
}
```

```
//Print table
for (idx = 1; idx<=n;
idx++,letter++) {
    for (idx2 = 1;
       idx2 <= n; idx2++) {
          printf (“%c%d “,
               letter, idx2);
    }
    printf(“\n”);
}

return 0;
```

Exercise

- Write a program to print out the following pattern consisting of x rows where x is an integer read from standard input:

```
!
!!
!!!
!!!!
!!!!!
!!!!!!
```

Exercise

- Write a program to print out the following pattern consisting of x rows where x is an integer read from standard input:

```
!          Row 1: 1 ex. mark
!!         Row 2: 2 ex. marks
!!!              ...
!!!!       Row x: x ex. marks
!!!!!            ...
!!!!!!
```

Exercise

- Write a program to print out the following pattern consisting of x rows where x is an integer read from standard input:

```
for (idx1 = 1; idx1<=n; idx1++) {
    for (idx2 = 1; idx2 <= idx1; idx2++) {
        printf("!");
    }
    printf("\n");
}
```

Summary

- Iteration statements
  - While loop
  - Do loop
  - For loop
- break, continue
- Comma operator
- Null statement

What you actually need to know

- While loops
- For loops
- Break and Continue
- The rest probably shouldn't be used until you are much more comfortable with the language.