C Programming Introduction

Week 7:Loops

Topic of this week

- Loops
  - Class Lecture Review
    - The for Repetition Structure
    - Notes and Observations
  - Programming Exercises

The `for` Repetition Structure

- Format when using **`for`** loops

`for (` *initialization* **`;`** *loopContinuationTest* **`;`** *increment* **`)`**
  statement

Example:

```
for( int counter = 1; counter <= 10; counter++ )
    printf( "%d\n", counter );
```

No semicolon after last expression

  - Prints the integers from one to ten.

The `for` Repetition Structure (II)

- **`For`** loops can usually be rewritten as **`while`** loops:

```
initialization;
while ( loopContinuationTest) {
   statement
   increment;
}
```

- Initialization and increment
  - Can be comma-separated lists

```
for (int i = 0, j = 0;  j + i <= 10; j++, i++)
   printf( "%d\n", j + i );
```

The For Structure: Notes and Observations

- Arithmetic expressions
  - Initialization, loop-continuation, and increment can contain arithmetic expressions. If `x = 2` and `y = 10`

```
for ( j = x; j <= 4 * x * y; j += y / x )
```

  is equivalent to

```
for ( j = 2; j <= 80; j += 5 )
```

- "Increment" may be negative (decrement)
- If loop continuation condition initially `false`
  - Body of `for` structure not performed
  - Control proceeds with statement after `for` structure

The For Structure: Notes and Observations (II)

- Control variable
  - Often printed or used inside **`for`** body, but not necessary
- **`For`** flowcharted like **`while`**

Example

- Example of For

```
for (i=1;i<=100;i++) {
   x += i;
   if ((x % i) == 0) { i--; }
}
```

```
for (i=0, j=strlen(s)-1; i<j; i++,j--)
  { c = s[i], s[i] = s[j], s[j] = c; }
```

```
char c;
int count;
for (count=0; (c=getchar() != '.'); count++)
  { }
printf("Number of characters is %d\n", count);
```

Exercise 7.1

- Write a program that prints ten integers and their squares.

```
1     1
2     4
3     9
...
10    100
```

Solution

```
#include <stdio.h>

int main()
{
   int i;

   for(i = 1; i <= 10; i = i + 1)
   printf("%d  %d\n", i, i * i);

   return 0;
}
```

Exercise 7.2

- Write a program that prints out a triangle like:

```
*
**
***
****
*****
******
*******
********
*********
**********
```

Solution

```
#include <stdio.h>

int main()
{
    int i, j;

    for(i = 1; i <= 10; i = i + 1)
    {
    for(j = 1; j <= i; j = j + 1)
    printf("*");
    printf("\n");
    }

    return 0;
}
```

Exercise 7.3

- Write a program that lists numbers which is greater than 27 from 1 to 100.

Solution

```
#include <stdio.h>

int main()
{
int i;
int count = 0;

for(i = 1; i <= 100; i = i + 1)
   {
   if(i > 27)
   count = count + 1;
   }
printf("%d numbers were greater than 27\n", count);
return 0;
}
```

Exercise 7.4

- Write a program that lists prime numbers which is smaller than 100.
- Use math.h library to use some mathematical functions: sqrt,...

Solution

```
#include <stdio.h>
#include <math.h>

main()
{
int i, j;

printf("%d\n", 2);

for(i = 3; i <= 100; i = i + 1)
    {
    for(j = 2; j < i; j = j + 1)
    {
    if(i % j == 0)
    break;
```

Solution

```
if(j > sqrt(i))
                {
                printf("%d\n", i);
                break;
                }
          }
      }
return 0;
}
```

Exercise 7.5

- Alter the exercise 7.4 above by eliminating the even numbers to avoid calling sqrt function many times.

Solution

```
#include <stdio.h>
#include <math.h>

int main()
{
int i, j;
double sqrti;

printf("%d\n", 2);

for(i = 3; i <= 100; i = i + 2)
    {
    sqrti = sqrt(i);
```

Solution

```
    for(j = 2; j < i; j = j + 1)
        {
        if(i % j == 0)
                break;                      /* not prime */
        if(j > sqrt(i))
                {                           /* prime */
                printf("%d\n", i);
                break;
                }
        }
    }

return 0;
}
```

Exercise 7.6

- Try the following program in your compiler.

```
/* Counting down to blast-off */
#include <stdio.h>

int main(void)
{
    int time, start;

    printf("Enter starting time (an integer) in seconds> ");
    scanf("%d", &start);
    printf("\nBegin countdown\n");

    for (time = start; time > 0; time = time - 1)
    {
    printf("T - %d\n", time);
    }

    printf("Blast-off!\n");
    return (0);
}
```

Exercise 7.7

- Write a program that converts temperatures from Celsius to Fahrenheit.
- Notice the conditions of the loops continuation and the way in which #define macros are used to set constant values.
- fahrenheit = 1.8 * celsius + 32.0;

Solution

```
#include <stdio.h>

/* Constant macros */
#define CBEGIN 10
#define CLIMIT -5
#define CSTEP 5

int
main(void)
{
    /* Variable declarations */
    int    celsius;
    double fahrenheit;

    /* Print the table heading */
    printf("  Celsius  Fahrenheit\n");
```

Solution

```
/* Print the table */
    for  (celsius = CBEGIN;
          celsius >= CLIMIT;
          celsius = celsius - CSTEP) {
       fahrenheit = 1.8 * celsius + 32.0;
       printf("  %3d    %7.2f\n", celsius,
  fahrenheit);
    }

    return (0);
}
```

Exercise 7.8

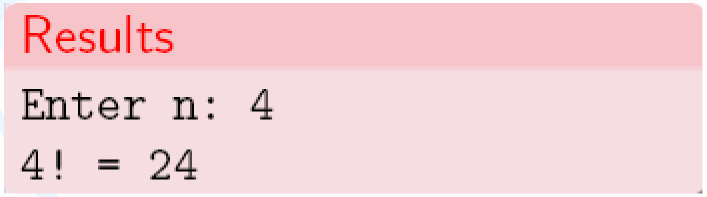
- Sometimes we need to have loops within loops, this is called nested loops. This program demonstrates how this works. By running it you should see in what sequence the code is called.

exercise7_8.c

```
#include <stdio.h>

int main(void)
{
    int i, j;        /* loop control variables */

    printf("         I    J\n");                /* prints column labels */

    for  (i = 1;  i < 4;  i = i + 1)  /* heading of outer for loop */
     {
       printf("Outer %6d\n", i);
       for  (j = 0;  j < i;  j = j + 1)          /* heading of inner loop */
     {
          printf("  Inner%9d\n", j);
       } /* end of inner loop */
    }    /* end of outer loop */
     return (0);
}
```

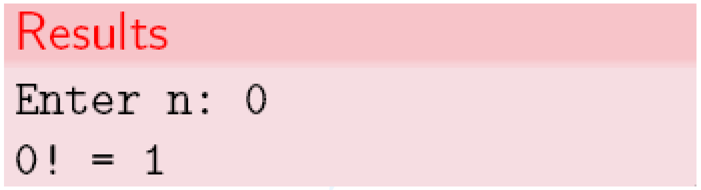
Exercise 7.9

- Write a program that uses *for structure* to calculate the value of n!.
- Some outputs:



Results

```
Enter n: 4
4! = 24
```



Results

```
Enter n: 0
0! = 1
```

Solution

```
# include <stdio.h>
int main ()
{
    int i, n, f;
    printf (" Enter n: ");
    scanf ("%d", &n);
    f = 1; /* 0! */
    for (i = 1; i <= n; ++i) {
    f *= i; /* Now , f = i! */
    }
    printf ("%d! = %d\n", n, f);
    return 0;
}
```

Exercise 7.10

- In mathematics, a **perfect number** is defined as a positive integer which is the sum of its proper positive divisors, that is, the sum of the positive divisors not including the number itself. E.g: 6=1+2+3
- Write a program that lists perfect numbers which is smaller than inputed N.

Solution

```
#include <stdio.h>

void main()
{
  int n, i, j, tong;

  printf("\nEnter N= : ");
  scanf("%d", &n);
  for (i=2; i<=n; i++)
  {
    tong = 1;
    for (j=2; j<=i/2; j++)
      if (i%j == 0)
        tong += j;
      if (tong == i)
        printf("\n%d", i);
  }
}
```