Readin from and Writint to Standart I/O

BIL104E: Introduction to Scientific and Engineering Computing

Lecture 5

- Playing with Data Modifiers and Math Functions
- Getting Controls

Pointers

What Is a Pointer?

A pointer is a variable whose value is used to point to another variable.

Calling sin(), cos(), tan() functions

For instance, given an angle x in radians, the sin(x) expression returns the sine of the angle:

The following formula can be used to convert the value of an angle in degrees into the value in radians:

```
radians = degree * (3.141593 / 180.0).
```

Calling sin(), cos(), tan() functions

```
1: /* Using sin(), cos(), and tan() functions */
2: #include <stdio.h>
3: #include <math.h>
4: 5: main()
6: {
7: double x;
8:
9: x = 45.0; /* 45 degree */
10: x *= 3.141593 / 180.0; /* convert to radians */
11: printf("The sine of 45 is: %f.\n", sin(x));
12: printf("The cosine of 45 is: %f.\n", cos(x));
13: printf("The tangent of 45 is: %f.\n", tan(x));
14: return 0;
15:
```

```
Output:

The sine of 45 is: 0.707107.
The cosine of 45 is: 0.707107.
The tangent of 45 is: 1.000000.
```

Calling pow() and sqrt()

The value of the double variable **`x`** is raised to the power of **`y`** with the function **`pow(x,y)`** .

the **`sqrt(x)`** function returns the non-negative square root of **`x`** in the double data type

Calling pow() and sqrt()

```
1: /* 09L05.c: Using pow() and sqrt() functions */
2: #include <stdio.h>
3: #include <math.h>
4:
5: main()
6: {
7: double x, y, z;
8:
9: x = 64.0;
10: y = 3.0;
11: z = 0.5;
12: printf("pow(64.0, 3.0) returns: %7.0f\n", pow(x, y));
13: printf("sqrt(64.0) returns: %2.0f\n", sqrt(x));
14: printf("pow(64.0, 0.5) returns: %2.0f\n", pow(x, z));
15: return 0;
16: }
```

```
Output:

pow(64.0, 3.0) returns: 262144
sqrt(64.0) returns: 8
pow(64.0, 0.5) returns: 8
```

Essentials of Counter-Controlled Repetition

Getting Controls

- The **`if`** statement
- The **`if-else`** statement
- The switch statement
- The **`break`** statement
- The **`continue`** statement
- The **`goto`** statement

The `if` Statement

`if`

This is used to decide whether to do something at a special point, or to decide between two courses of action:

```
if (expression)
   statement;
next statent(s)
```

`if` **with one statement.**

```
if (expression)
{
statement1;
statement2;
. . .
}
next statent(s)
```

`if` **with more than one statements.**

The `if` Statement with `else`

The if-else Statement:

```
if (expression) {
statement1;
statement2;
  . . .
}
else
{
statement_A;
statement_B;
  . . .
}
```

The `if` Statement

```
1: /* 10L02.c Using the if-else statement */
2: #include <stdio.h>
3:
4: main()
5: {
6: int i;
7:
8: printf("Even Number Odd Number\n");
9: for (i=0; i<10; i++)
10: if (i%2 == 0)
11: printf("%d", i);
12: else
13: printf("%14d\n", i);
14:
15: return 0;
16: }
```

```
Output:

Even Number   Odd Number
0             1
2             3
4             5
6             7
8             9
```

The `if` Statement - nested

The if-else Statement:

```
if (expression) {
statement1;
statement2;
  . . .
}
else
{
statement_A;
statement_B;
  . . .
}
```

The `if` Statement - nested

The nested **`if-else`** Statement:

```
if (expression) {
   Statement(s);
   if (expression)
      if (expression)
         statement
}
else
{
   statement(s);
   if (expression) {
      statement(s)
   }
}
```

The `switch` Statement

if statements are used when there is more than one decision to be made. The nested if statements will become very complex if there are many decisions that need to be made, however .

The general form of the switch statement is

```
switch (expression) {
   case expression1:
        statement1;
    case expression2:
        statement2;
    . . .
    default:
        statement-default;
}
```

The switch Repetition Structure

```
1: /* 10L04.c Using the switch statement */
2: #include <stdio.h>
3:
4: main()
5: {
6:    int day;
7:
8:    printf("Please enter a single digit for a day\n");
9:    printf("(within the range of 1 to 3):\n");
10:   day = getchar();
11:   switch (day){
12:      case `1':
13:           printf("Day 1\n");
14:      case `2':
15:           printf("Day 2\n");
16:      case `3':
17:           printf("Day 3\n");
18:      default: 19: ;
20:    }
21: return 0;
22: }
```

```
Output:

Please enter a single digit for a day
(within the range of 1 to 3): 3
Day 3
```