FACULTY OF ENGINEERING AND TECHNOLOGY

COMPUTER SCIENCE DEPARTMENT

COMP1310
Introduction to Computer and Computing Ethics

SELECTION STRUCTURES: IF AND SWITCH

Control Structures

- A control structure is a combination of individual instructions into a single logical unite with one entry point and one exit point.
- They are usually bracketed by { and }

- A selection control structure is a control structure that chooses among alternative program statements.
- This means that there is a condition the guides the selection control structure on which alternative to choose.
- A condition is an expression that is either false or true.
- Note that false can be represented as 0, and true can be represented as a non-zero number, usually 1.

Relational and Equality Operators

- Conditions usually have one of the following forms:
  - *variable relational-operator* *variable/constant*
  - *variable equality-operator* *variable/constant*

Operator	Meaning	Type
<	Less than	Relational
>	Greater than	Relational
<=	Less than or equal	Relational
>=	Greater than or equal	Relational
==	Equal to	Equality
!=	Not equal to	Equality

Logical Operators

- A logical expression is an expression that uses one or more logical operators.

Operator	Meaning
&&	AND
||	OR
!	NOT

Truth Tables - AND

operand1	operand2	operand1 && operand2
True	True	True
True	False	False
False	True	False
False	False	False

Truth Tables - OR

operand1	operand2	operand1 || operand2
True	True	True
True	False	True
False	True	True
False	False	False

Truth Tables - NOT

operand	!operand
True	False
False	True

Operator Precedence - Updated

From the highest to the lowest

- Function calls
- Unary operators (! + -)
- * / %
- + -
- < <= >= >
- == !=
- &&
- ||
- =

English Conditions as C Expressions

Let's assume here that x = 3.0, y = 4.0, and z = 2.0

English Condition	Logical Expression	Evaluation

English Conditions as C Expressions

Let's assume here that x = 3.0, y = 4.0, and z = 2.0

English Condition	Logical Expression	Evaluation
x and y are greater than z		

English Conditions as C Expressions

Let's assume here that x = 3.0, y = 4.0, and z = 2.0

English Condition	Logical Expression	Evaluation
x and y are greater than z	x > z && y > z	

English Conditions as C Expressions

Let's assume here that x = 3.0, y = 4.0, and z = 2.0

English Condition	Logical Expression	Evaluation
x and y are greater than z	x > z && y > z	T && T is T

English Conditions as C Expressions

Let's assume here that x = 3.0, y = 4.0, and z = 2.0

English Condition	Logical Expression	Evaluation
x and y are greater than z	x > z && y > z	T && T is T
x is equal to 1.0 or 3.0		

English Conditions as C Expressions

Let's assume here that x = 3.0, y = 4.0, and z = 2.0

English Condition	Logical Expression	Evaluation
x and y are greater than z	x > z && y > z	T && T is T
x is equal to 1.0 or 3.0	x == 1.0 || x == 3.0	

English Conditions as C Expressions

Let's assume here that x = 3.0, y = 4.0, and z = 2.0

English Condition	Logical Expression	Evaluation
x and y are greater than z	x > z && y > z	T && T is T
x is equal to 1.0 or 3.0	x == 1.0 || x == 3.0	F || T is T

English Conditions as C Expressions

Let's assume here that x = 3.0, y = 4.0, and z = 2.0

English Condition	Logical Expression	Evaluation
x and y are greater than z	x > z && y > z	T && T is T
x is equal to 1.0 or 3.0	x == 1.0 || x == 3.0	F || T is T
x is in the range z to y inclusive		

English Conditions as C Expressions

Let's assume here that x = 3.0, y = 4.0, and z = 2.0

English Condition	Logical Expression	Evaluation
x and y are greater than z	x > z && y > z	T && T is T
x is equal to 1.0 or 3.0	x == 1.0 || x == 3.0	F || T is T
x is in the range z to y inclusive	z <= x && x <= y	

English Conditions as C Expressions

Let's assume here that x = 3.0, y = 4.0, and z = 2.0

English Condition	Logical Expression	Evaluation
x and y are greater than z	x > z && y > z	T && T is T
x is equal to 1.0 or 3.0	x == 1.0 || x == 3.0	F || T is T
x is in the range z to y inclusive	z <= x && x <= y	T && T is T

English Conditions as C Expressions

Let's assume here that x = 3.0, y = 4.0, and z = 2.0

English Condition	Logical Expression	Evaluation
x and y are greater than z	x > z && y > z	T && T is T
x is equal to 1.0 or 3.0	x == 1.0 || x == 3.0	F || T is T
x is in the range z to y inclusive	z <= x && x <= y	T && T is T
x is outside the range z to y		

English Conditions as C Expressions

Let's assume here that x = 3.0, y = 4.0, and z = 2.0

English Condition	Logical Expression	Evaluation
x and y are greater than z	x > z && y > z	T && T is T
x is equal to 1.0 or 3.0	x == 1.0 || x == 3.0	F || T is T
x is in the range z to y inclusive	z <= x && x <= y	T && T is T
x is outside the range z to y	!(z <= x && x <= y)	

English Conditions as C Expressions

Let's assume here that x = 3.0, y = 4.0, and z = 2.0

English Condition	Logical Expression	Evaluation
x and y are greater than z	x > z && y > z	T && T is T
x is equal to 1.0 or 3.0	x == 1.0 || x == 3.0	F || T is T
x is in the range z to y inclusive	z <= x && x <= y	T && T is T
x is outside the range z to y	!(z <= x && x <= y)	!(T && T) is !T is F

English Conditions as C Expressions

Let's assume here that x = 3.0, y = 4.0, and z = 2.0

English Condition	Logical Expression	Evaluation
x and y are greater than z	x > z && y > z	T && T is T
x is equal to 1.0 or 3.0	x == 1.0 || x == 3.0	F || T is T
x is in the range z to y inclusive	z <= x && x <= y	T && T is T
x is outside the range z to y	!(z <= x && x <= y)	!(T && T) is !T is F
	z > x || x > y	

English Conditions as C Expressions

Let's assume here that x = 3.0, y = 4.0, and z = 2.0

English Condition	Logical Expression	Evaluation
x and y are greater than z	x > z && y > z	T && T is T
x is equal to 1.0 or 3.0	x == 1.0 || x == 3.0	F || T is T
x is in the range z to y inclusive	z <= x && x <= y	T && T is T
x is outside the range z to y	!(z <= x && x <= y)	!(T && T) is !T is F
	z > x || x > y	F || F is F

Logical Assignment

- We can assign the result of a conditional statement to a variable, such as an integer.
- The value that will be stored in the variable will be:
  - *1 if the conditional statement is true*
  - *0 if the conditional statement is false*
- For example, if we want to print the value of a conditional statements that checks if a grade is a passing grade or not.

Logical Assignment – cont.

```c
#include <stdio.h>

void main() {
    printf("Please enter a grade: ");
    int grade;
    scanf("%d", &grade);
    int pass = (grade >= 60);
    printf("Grade %d is a pass = %d.", grade, pass);
}
```

The *if* statement

- With if statements we can tell the compiler to do something specific only when some condition is true.
- if statements can be used to run one statement, or compound statements.
- if statements can have one alternative:
  - *do something if the condition is true.*
- or they can have two alternatives:
  - *do something if the condition is true.*
  - *do something else if the condition is false.*

The structure of *if* statements

One alternative, single statement

```
if (condition)
        something;
```

The structure of *if* statements

Two alternatives, single statements

```
if (condition)
        something;

else
        something;
```

The structure of *if* statements

```
if (condition) {
        something;
}
else {
        something;
}
```

Two alternatives, compound statements

The structure of *if* statements

Two alternatives, compound statements

```
if (condition) {
        something;
}
else {
        something;
}
```

if statements can be made of any combination of alternatives and statements.

Example – print ‘pass’ or ‘fail’

- Let’s modify the example we used previously:
  - *Write a program that reads a student’s grade and prints out if the student passed or failed.*

- The condition:

Example – print ‘pass’ or ‘fail’

- Let’s modify the example we used previously:
  - *Write a program that reads a student’s grade and prints out if the student passed or failed.*

- The condition: the student’s grade is larger than or equal to 60.

Example – print ‘pass’ or ‘fail’

- Let’s modify the example we used previously:
  - *Write a program that reads a student’s grade and prints out if the student passed or failed.*

- The condition: the student’s grade is larger than or equal to 60.
- If the condition is true:

Example – print ‘pass’ or ‘fail’

- Let’s modify the example we used previously:
  - *Write a program that reads a student’s grade and prints out if the student passed or failed.*

- The condition: the student’s grade is larger than or equal to 60.
- If the condition is true: print that the student passed.

Example – print ‘pass’ or ‘fail’

- Let’s modify the example we used previously:
  - *Write a program that reads a student’s grade and prints out if the student passed or failed.*

- The condition: the student’s grade is larger than or equal to 60.
- If the condition is true: print that the student passed.
- If the condition is false:

Example – print ‘pass’ or ‘fail’

- Let’s modify the example we used previously:
  - *Write a program that reads a student’s grade and prints out if the student passed or failed.*

- The condition: the student’s grade is larger than or equal to 60.
- If the condition is true: print that the student passed.
- If the condition is false: print that the student failed.

Example – print ‘pass’ or ‘fail’

```
1   #include <stdio.h>
2
3   void main() {
4       printf("Please enter a grade: ");
5       int grade;
6       scanf("%d", &grade);
7       if (grade >= 60)
8           printf("The student passed.");
9       else
10          printf("The student failed.");
11
12  }
```

Example – multiple alternative conditions

- Write a program that reads a student's average and prints out their letter grade based on the following criteria:

Average	Letter grade
$0 \leq \text{average} < 60$	F
$60 \leq \text{average} < 70$	D
$70 \leq \text{average} < 80$	C
$80 \leq \text{average} < 90$	B
$90 \leq \text{average}$	A

Example – multiple alternative conditions

```c
#include <stdio.h>

void main() {
    printf("Please enter a grade: ");
    int grade;
    scanf("%d", &grade);
    if (grade < 60)
        printf("F");
    else
        if (grade < 70)
            printf("D");
        else
            if (grade < 80)
                printf("C");
            else
                if (grade < 90)
                    printf("B");
                else
                    printf("A");
}
```

Conditions and Brackets

- What is the output of the following program:

```c
#include <stdio.h>

void main() {
    int x = 0;
    if (x)
        printf("condition is true\n");
    printf("this statement\n");
}
```

Conditions and Brackets

- What is the output of the following program:

```
1  #include <stdio.h>
2
3  void main() {
4      int x = 0;
5      if (x)
6          printf("condition is true\n");
7      printf("this statement\n");
8  }
```

Conditions and Brackets

- What is the output of the following program:

```
1  #include <stdio.h>
2
3  void main() {
4      int x = 8;
5      if (x)
6          printf("condition is true\n");
7      printf("this statement\n");
8  }
```

Conditions and Brackets

- What is the output of the following program:

```
1  #include <stdio.h>
2
3  void main() {
4      int x = 8;
5      if (x)
6          printf("condition is true\n");
7      printf("this statement\n");
8  }
```

Conditions and Brackets

- What is the output of the following program:

```
1  #include <stdio.h>
2
3  void main() {
4      int x = 8;
5      if (x)
6          printf("condition is true\n");
7      printf("this statement\n");
8  }
```

Conditions and Brackets

- What is the output of the following program:

```c
#include <stdio.h>

void main() {
    int x = 0;
    if (x = 8)
        printf("condition is true\n");
    printf("this statement\n");
}
```

Conditions and Brackets

- What is the output of the following program:

```
1  #include <stdio.h>
2
3  void main() {
4      int x = 0;
5      if (x = 8)
6          printf("condition is true\n");
7      printf("this statement\n");
8  }
```

Conditions and Brackets

- What is the output of the following program:

```
1  #include <stdio.h>
2
3  void main() {
4      int x = 0;
5      if (x = 8)
6          printf("condition is true\n");
7      printf("this statement\n");
8  }
```

Conditions and Brackets

- What is the output of the following program:

```c
#include <stdio.h>

void main() {
    int x = 8;
    if (x == 8)
        printf("condition is true\n");
    else
        printf("condition is false\n");
}
```

Conditions and Brackets

- What is the output of the following program:

```
1  #include <stdio.h>
2
3  void main() {
4      int x = 8;
5      if (x == 8)
6          printf("condition is true\n");
7      else
8          printf("condition is false\n");
9  }
```

Conditions and Brackets

- What is the output of the following program:

```c
#include <stdio.h>

void main() {
    int x = 8;
    if (x == 8) {
        printf("condition is true\n");
        printf("x equals 8\n");
    }
    else {
        printf("condition is false\n");
        printf("x does not equal 8\n");
    }
}
```

Conditions and Brackets

- What is the output of the following program:

```
1   #include <stdio.h>
2
3   void main() {
4       int x = 8;
5       if (x == 8) {
6           printf("condition is true\n");
7           printf("x equals 8\n");
8       }
9       else {
10          printf("condition is false\n");
11          printf("x does not equal 8\n");
12      }
13  }
```

Conditions and Brackets

- What is the output of the following program:

```
1   #include <stdio.h>
2
3   void main() {
4       int x = 8;
5       if (x == 8) {
6           printf("condition is true\n");
7           printf("x equals 8\n");
8       }
9       else
10          printf("condition is false\n");
11          printf("x does not equal 8\n");
12  }
```

Conditions and Brackets

- What is the output of the following program:

```
1   #include <stdio.h>
2
3   void main() {
4       int x = 8;
5       if (x == 8) {
6           printf("condition is true\n");
7           printf("x equals 8\n");
8       }
9       else
10          printf("condition is false\n");
11          printf("x does not equal 8\n");
12  }
```

Conditions and Brackets

- What is the output of the following program:

```
1   #include <stdio.h>
2
3   void main() {
4       int x = 8;
5       if (x == 8) {
6           printf("condition is true\n");
7           printf("x equals 8\n");
8       }
9       else
10          printf("condition is false\n");
11          printf("x does not equal 8\n");
12  }
```

Conditions and Brackets

- What is the output of the following program:

```
1   #include <stdio.h>
2
3   void main() {
4       int x = 8;
5       if (x == 8)
6           printf("condition is true\n");
7           printf("x equals 8\n");
8       else {
9           printf("condition is false\n");
10          printf("x does not equal 8\n");
11      }
12  }
```

Conditions and Brackets

- What is the output of the following program:

```
1  #include <stdio.h>
2
3  void main() {
4      int x = 8;
5      if (x == 8)
6          printf("condition is true\n");
7          printf("x equals 8\n");
8      else {
9          printf("condition is false\n");
10         printf("x does not equal 8\n");
11     }
12 }
```

Syntax Error

Nested *if* Statements

- Let's write a program that asks the user to enter the number of the day of the week (an integer between 1 and 7) and prints the corresponding name of the day of the week, starting with Sunday.
- This means 1 = Sunday, 2 = Monday, etc.

Nested *if* Statements

```c
#include <stdio.h>

void main() {
    printf("Enter the number of the day of the week (between 1 and 7): ");
    int day;
    scanf("%d", &day);

    if (day == 1)
        printf("Today is Sunday.\n");
    else if (day == 2)
        printf("Today is Monday.\n");
    else if (day == 3)
        printf("Today is Tuesday.\n");
    else if (day == 4)
        printf("Today is Wednesday.\n");
    else if (day == 5)
        printf("Today is Thursday.\n");
    else if (day == 6)
        printf("Today is Friday.\n");
    else if (day == 7)
        printf("Today is Saturday.\n");
    else
        printf("Error - the number you entered is invalid.");
}
```

Nested *if* Statements

- When we have multiple alternatives based on a single value of an integer or a character, we can replace if statements with a switch statement.

- Switch statements do not work with floats, doubles, or strings.

switch Statement

```
switch (integer or character) {
        case constant1:
                statement(s)
                break;
        case constant2:
                statement(s)
                break;
        ...
        default:
                statement(s)
                break;
}
```

switch Statement

```
switch (integer or character) {
        case constant1:
                statement(s)
                break;
        case constant2:
                statement(s)
                break;
        ...
        default:
                statement(s)
                break;
}
```

if the integer or character equals this value the following statements will be executed

switch Statement

```
switch (integer or character) {
        case constant1:
                statement(s)
                break;
        case constant2:
                statement(s)
                break;
        ...
        default:
                statement(s)
                break;
}
```

if the integer or character equals this value the following statements will be executed

Break causes the program to exit the switch statement

switch Statement

```
switch (integer or character) {
        case constant1:
                statement(s)
                break;
        case constant2:
                statement(s)
                break;
        ...
        default:
                statement(s)
                break;
}
```

if the integer or character equals this value
the following statements will be executed

Break causes the program to exit the switch statement

default is an optional section
it gets executed if all the ‘cases’ fail

Example – day of the week

- Let’s rewrite the previous example using a switch statement instead.

Example – day of the week

```c
#include <stdio.h>

void main() {
    printf("Enter the number of the day of the week (between 1 and 7): ");
    int day;
    scanf("%d", &day);

    switch (day) {
        case 1:
            printf("Today is Sunday.\n");
            break;
        case 2:
            printf("Today is Monday.\n");
            break;
        case 3:
            printf("Today is Tuesday.\n");
            break;
```

Example – day of the week

```
18              case 4:
19                  printf("Today is Wednesday.\n");
20                  break;
21              case 5:
22                  printf("Today is Thursday.\n");
23                  break;
24              case 6:
25                  printf("Today is Friday.\n");
26                  break;
27              case 7:
28                  printf("Today is Saturday.\n");
29                  break;
30              default:
31                  printf("Error – the number you entered is invalid.");
32                  break;
33          }
34      }
```

break Statement

- What would happen if we forgot the break statements in the previous program?
- Let’s suppose we input 1.

break Statement

```c
#include <stdio.h>

void main() {
    printf("Enter the number of the day of the week (between 1 and 7): ");
    int day;
    scanf("%d", &day);

    switch (day) {
        case 1:
            printf("Today is Sunday.\n");
        case 2:
            printf("Today is Monday.\n");
        case 3:
            printf("Today is Tuesday.\n");
        case 4:
            printf("Today is Wednesday.\n");
        case 5:
            printf("Today is Thursday.\n");
        case 6:
            printf("Today is Friday.\n");
        case 7:
            printf("Today is Saturday.\n");
        default:
            printf("Error - the number you entered is invalid.");
    }
}
```

After the statement of case 1 gets executed, there's nothing to tell the program it needs to exit the control structure.
So, it will continue to execute every statement is finds until it meets a break, or the end of the control structure.

Why Use *switch* Statements?

- The code we wrote using the nested if statements was 24 lines.
- The code we wrote using the switch statements was 34 lines.
- Why do we use switch statements then?

Why Use *switch* Statements?

- The code we wrote using the nested if statements was 24 lines.
- The code we wrote using the switch statements was 34 lines.
- Why do we use switch statements then?

- switch statements are:
  1. *Easier to read.*
  2. *Easier to extend (add new cases).*

Practical Use of *switch* Statements – Menus

- switch statements are often used when creating menus for you program.
- For example, you want to create a calculator that can do all five of basic arithmetic operations: +, -, *, \, %.
- And you want to let the user select which operations to do.
- Print out a menu that assigns each operations to a number and ask the user to enter the number of the operation they want to do.
- Use a switch statement to call the appropriate function.

Practical Use of *switch* Statements – Menus

```
1   #include <stdio.h>
2
3   void main()
4   {
5       printf("Menu\n");
6       printf("1. Addition\n");
7       printf("2. Subtraction\n");
8       printf("3. Multiplication\n");
9       printf("4. Division\n");
10
11      printf("Please select the number of the arithmetic opeeration you want to do: ");
12      int choice;
13      scanf("%d", &choice);
14
15      double num_1, num_2;
16      printf("Please enter the first number: ");
17      scanf("%lf", &num_1);
18      printf("Please enter the second number: ");
19      scanf("%lf", &num_2);
20
```

Practical Use of *switch* Statements – Menus

```
21      double result;
22      switch(choice) {
23          case 1:
24              result = num_1 + num_2;
25              printf("%f + %f = %f", num_1, num_2, result);
26              break;
27
28          case 2:
29              result = num_1 - num_2;
30              printf("%f - %f = %f", num_1, num_2, result);
31              break;
32          case 3:
33              result = num_1 * num_2;
34              printf("%f x %f = %f", num_1, num_2, result);
35              break;
36          case 4:
37              result = num_1 / num_2;
38              printf("%f / %f = %f", num_1, num_2, result);
39              break;
40          default:
41              printf("The number you entered is not valid.");
42      }
43  }
```

Practical Use of *switch* Statements – Menus

- Let’s modify the previous code to include the modulo operation.
- Remember that modulo can only be performed on integer numbers.

Practical Use of *switch* Statements – Menus

```c
#include <stdio.h>

void main()
{
    printf("Menu\n");
    printf("1. Addition\n");
    printf("2. Subtraction\n");
    printf("3. Multiplication\n");
    printf("4. Division\n");
    printf("5. Modulo\n");

    printf("Please select the number of the arithmetic opeeration you want to do: ");
    int choice;
    scanf("%d", &choice);

    if (choice == 5) {
        int num_1, num_2;
        printf("Please enter the first number: ");
        scanf("%d", &num_1);
        printf("Please enter the second number: ");
        scanf("%d", &num_2);

        int result = num_1 % num_2;
        printf("%d %% %d = %d", num_1, num_2, result);
    }
```

Practical Use of *switch* Statements – Menus

```
26      else {
27          double num_1, num_2;
28          printf("Please enter the first number: ");
29          scanf("%lf", &num_1);
30          printf("Please enter the second number: ");
31          scanf("%lf", &num_2);
32
33          double result;
34          switch(choice) {
35              case 1:
36                  result = num_1 + num_2;
37                  printf("%f + %f = %f", num_1, num_2, result);
38                  break;
39
40              case 2:
41                  result = num_1 - num_2;
42                  printf("%f - %f = %f", num_1, num_2, result);
43                  break;
44              case 3:
45                  result = num_1 * num_2;
46                  printf("%f x %f = %f", num_1, num_2, result);
47                  break;
48              case 4:
49                  result = num_1 / num_2;
50                  printf("%f / %f = %f", num_1, num_2, result);
51                  break;
52              default:
53                  printf("The number you entered is not valid.");
54          }
55      }
56  }
```

Example – Even or Odd

- Write a program that reads a number from the user and prints a message saying if the number is even or odd.

Example – Even or Odd

```c
#include <stdio.h>

void main()
{
    printf("Please enter an integer numbers:\n");
    int a;
    scanf("%d", &a);

    if (a % 2) {
        printf("The number is odd");
    } else {
        printf("The number is even");
    }
}
```

Example – Sides of a Triangle

- Write a program that checks if three entered integers can form a triangle.

- In a triangle, the sum of lengths of two of its sides must always be larger than the length of the third side.
  - *If the triangle has sides a, b, and c, then the following statements must always be true:*
    - a + b > c
    - a + c > b
    - b + c > a

Example – Sides of a Triangle

```c
#include <stdio.h>

void main() {
    printf("Please enter three integer numbers:\n");
    int a, b, c;
    scanf("%d%d%d", &a, &b, &c);

    if (a + b > c && a + c > b && b + c > a) {
        printf("These numbers can form a triangle.");
    } else {
        printf("These numbers cannot form a triangle.");
    }
}
```

Example – Maximum of Three Number

- Write a program that reads three numbers from the user and prints the maximum number.

Example – Maximum of Three Number

```c
#include <stdio.h>

void main()
{
    printf("Please enter three numbers:\n");
    double a, b, c;
    scanf("%lf%lf%lf", &a, &b, &c);

    double max = a;
    if (max < b)
        max = b;

    if (max < c)
        max = c;

    printf("The maximum number is %f", max);
}
```

Example – English Alphabet

- Write a program to read a character from the user and check if it is an English letter.
- We need to check if the letter is in the range of A – Z or if it is in the range of a - z.
- Since characters are stored in the computer as numbers, we can use relational operators with them.

Example – English Alphabet

```c
#include <stdio.h>

void main() {
    printf("Please enter a character:\n");
    char c;
    scanf("%c", &c);

    if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z')) {
        printf("This character is an English alphabet.");
    } else {
        printf("This character is an English alphabet.");
    }
}
```

Example – Day of the Year

- Write a program that takes a positive integer in the range 1 to 365 as input and outputs the day of the week. Assume that day 1 is Sunday.

Example – Day of the Year

```c
#include <stdio.h>

void main() {
    printf("Please enter the day of the year:\n");
    int day;
    scanf("%d", &day);

    int day_of_week = day % 7;

    switch (day) {
        case 0:
            printf("The day is Sunday.\n");
            break;
        case 1:
            printf("The day is Monday.\n");
            break;
```

Example – Day of the Year

```
17              case 2:
18                  printf("The day is Tuesday.\n");
19                  break;
20              case 3:
21                  printf("The day is Wednesday.\n");
22                  break;
23              case 4:
24                  printf("The day is Thursday.\n");
25                  break;
26              case 5:
27                  printf("The day is Friday.\n");
28                  break;
29              case 6:
30                  printf("The day is Saturday.\n");
31                  break;
32          }
33      }
```

Example – Rounding Using if

- Remember the rounding formula we saw in the previous chapter that used the floor function? Let's try to replicate what it does using if statements.

- What we want to do:
  - *Try to isolate the digit after the one we are rounding.*
  - *Determine if this digit is more or equal to five or if it is less than 5.*
  - *If it is more or equal to five, we add one to the digit before it.*
  - *If it is less than five, we don't add anything to the digit before.*

Example – Rounding Using if

```c
#include <stdio.h>
#include <stdlib.h>

void main() {
    printf("Please enter a number:");
    double number;
    scanf("%lf", &number);

    //To round to the third digit after the floating point:
    //isolate the number in the fourth digit after floating point
    int temp_number = number * 10000; //one more zero than the one we want to round
    int digit = temp_number % 10;

    temp_number = temp_number / 10; //remove hte digit from the number

    //check if the digit is more than or equal to 5:
    if(digit >= 5) {
        //add 1 to the number:
        temp_number += 1;
    }
    //if it is not, we don't need to do anything.

    //get the number back to its original value:
    number = temp_number / 1000.0;

    printf("The number rounded to the third digit is %lf", number);
}
```