CMSC 104 - Lecture 11
John Y. Park, adapted by C Grasso

The C "switch" Statement

The `switch` Statement

Topics

- Multiple Selection vs Binary Selection
- **`switch`** Statement
- **`char`** data type and **`getchar`**( ) function
- Reading newline characters

Binary Selection

- So far, we have only seen **binary selection**.

```
if ( age >= 18 ) {
  printf("Vote!\n") ;
}
```

```
if ( age >= 18 ) {
  printf("Vote!\n") ;
}
else {
  printf("Nope!\n") ;
}
```

Multiple Selection

- Sometimes it is necessary to **branch** in more than two directions.
- We do this using **multiple selection**.

Multiple Selection

```
if (day == 0 ) {
  printf (“Sunday”) ;
}
if (day == 1 ) {
  printf (“Monday”) ;
}
if (day == 2) {
  printf (“Tuesday”) ;
}
if (day == 3) {
  printf (“Wednesday”);
}
```

(continued)

```
if (day == 4) {
    printf (“Thursday”) ;
}
if (day == 5) {
    printf (“Friday”) ;
}
if (day == 6) {
    printf (“Saturday”) ;
}
if ((day < 0) || (day > 6)) {
    printf(“Invalid day.\n”);
}
```

Multiple Selection with if-else

```
if (day == 0 ) {
    printf (“Sunday”) ;
} else if (day == 1 ) {
    printf (“Monday”) ;
} else if (day == 2) {
    printf (“Tuesday”) ;
} else if (day == 3) {
    printf (“Wednesday”) ;
} else if (day == 4) {
    printf (“Thursday”) ;
} else if (day == 5) {
    printf (“Friday”) ;
} else if (day == 6) {
    printf (“Saturday”) ;
} else {
    printf (“Invalid day.\n”)
  ;
}
```

Why is this **`if-else`** structure is more efficient than the corresponding **`if`** structure?

Multiple Selection with if-else

```
if (day == 0 ) {
    printf (“Sunday”) ;
    day = 3;
}
if (day == 1 ) {
    printf (“Monday”) ;
}
if (day == 2) {
    printf (“Tuesday”) ;
}
if (day == 3) {
    printf (“Wednesday”) ;
}
if (day == 4) {
    printf (“Thursday”) ;
}
…
```

VS.

```
if (day == 0 ) {
    printf (“Sunday”) ;
    day = 3;
}
else if (day == 1 ) {
    printf (“Monday”) ;
}
else if (day == 2) {
    printf (“Tuesday”) ;
}
else if (day == 3) {
    printf (“Wednesday”) ;
}
else if (day == 4) {
    printf (“Thursday”) ;
}
…
```

Multiple Selection with `switch`

- The multiple selection mechanism in C is the **`switch`** statement.
  - Test a single integer variable against multiple different constant integer values
  - Execute statements based on the success of each test.
  - Will fall through and execute the statements in the next value unless told otherwise

The switch Multiple-Selection Structure

```
switch ( integer expression )
{
    case constant1 :
        statement(s)
        break ;
    case constant2 :
        statement(s)
        break ;

        . . .
    default:
        statement(s)
        break ;
}
```

case : break;

- The last statement of each **`case`** in the switch should almost always be a **`break`**.
- The **`break`** causes program control to jump to the closing brace of the **`switch`** structure.
- Without the **`break`**, the code flows into the next **`case`**. This is almost never what you want.
- A switch statement will compile without a **`default`** case, but always consider using one.

`default:`

- Include a **`default`** case to catch invalid data.
- Inform the user of the type of error that has occurred (e.g., "Error - invalid day.").
- If appropriate, display the invalid value.
- If appropriate, terminate program execution

switch Example

```
switch ( day )
{
   case 0:  printf (“Sunday\n”) ;
            break ;
   case 1:  printf (“Monday\n”) ;
            break ;
   case 2:  printf (“Tuesday\n”) ;
            break ;
   case 3:  printf (“Wednesday\n”) ;
            break ;
   case 4:  printf (“Thursday\n”) ;
            break ;
   case 5:  printf (“Friday\n”) ;
            break ;
   case 6:  printf (“Saturday\n”) ;
            break ;
   default: printf (“Invalid.\n”);
            break ;
}
```

Is this structure more efficient than the equivalent nested if-else structure?

Why Use a switch Statement?

- A switch statement can be more efficient than an if-else.
- A switch statement may also be easier to read.
- Also, it is easier to add new cases to a switch statement than to a nested if-else structure.

The char Data Type

- The **char** data type holds a single character.

```
char ch;
```

- Example assignments:

```
char grade, symbol;
grade  = 'B';
symbol = '$';
```

- The **char** is held as a one-byte integer in memory. The ASCII code is what is actually stored, so we can use them as characters or integers, depending on our need.

The char Data Type (con't)

- Use

  scanf ("%c", &ch) ;

  to read a single character into the variable ch.

- Use

  printf("%c", ch) ;

  to display the value of a character variable.

char Example

```
#include <stdio.h>
int main ( )
{
   char ch ;

   printf (“Enter a character: “) ;
   scanf (“%c”, &ch) ;
   printf (“The value of %c is %d.\n”, ch, ch) ;
      return 0 ;
}
```

If the user entered an A, the output would be:

The value of A is 65.

The getchar ( ) Function

- The **`getchar`**( ) function is found in the **stdio** library.
- The **`getchar`**( ) function reads one character from **stdin** (the **keyboard**) and returns that character's ASCII value.
- The value can be stored in either a character variable or an integer variable.

getchar ( ) Example

```
#include <stdio.h>
int main ( )
{
   char ch ;      /* int ch  would also work! */

   printf (“Enter a character: “) ;
   ch = getchar( ) ;        /*same as scanf(“%c”, &ch); */

   printf (“The value of %c is %d.\n”, ch, ch) ;
   return 0 ;
}
```

If the user entered an A, the output would be:

The value of A is 65.

Problems with Reading Characters

- When getting characters, whether using **`scanf`**( ) or **`getchar`**( ), realize that you are reading only one character.
- What will the user actually type?
  - The character - followed by pressing ENTER.
- So, the user is actually entering two characters, the response and the **newline character**.
- Unless you handle this, the newline character will remain in the stdin stream causing problems the next time you want to read a character. Another call to **`scanf`**() or **`getchar`**( ) will remove it.

Additional Concerns with Garbage in stdin

- When we were reading integers using **`scanf`**( ), we didn't seem to have problems with the newline character, even though the user was typing ENTER after the integer.
- That is because **`scanf`**( ) was looking for the next integer and ignored the newline (**whitespace**).
- If we use **`scanf`** ("%d", &num); to get an integer, the newline is still stuck in the input stream.
- If the next item we want to get is a character, whether we use **`scanf`**( ) or **`getchar`**( ), we will get the newline.
- We have to take this into account and remove it.

Improved Character Example

```
#include <stdio.h>
int main ( )
{
     char ch, newline ;

     printf (“Enter a character: “) ;
     ch = getchar( ) ;
     newline = getchar( ) ;
     printf (“The value of %c is %d.\n”, ch, ch) ;

     printf (“Enter another character: “) ;
     ch = getchar( ) ;
     newline = getchar( ) ;
     printf (“The value of %c is %d.\n”, ch, ch) ;

     return 0 ;
}
```

Menu.c

```
Choose one option:
  1 – Convert Fahrenheit to Celsius
  2 – Convert kilometers to miles
  3 – Convert knots to miles per hour
  0 – Exit program
```

1. Display this menu to the user's console
2. Get the user's choice as a character
3. Call an appropriate method to perform that conversion

Menu.c

```
int main()
{
   int num = 0;

   do {

   } while ( num < 0 ) ;

   return 0;
}
```

do-while - Get User Input

```
int main()
{
   int num = 0;

   do {
      printf ("Enter a positive number: ");
      scanf ("%d", &num) ;

      if ( num < 0 ) {
            printf ("\nTry again\n");
      }

   } while ( num < 0 ) ;

   return 0;
}
```

do-while -- Menu

```
int main()
{
   char option ;

   do {
      printf ("Choose one option: \n");
      ...
      option = getchar();

      if (option == 0 ) { ... }
      if (option == 1 ) { ... }
      ...

   } while (option < 0 || option > 3 ) ;
}
```

Multiple Selection with if-else

```
if (option == ‘1’ ) {
  convertFahrenhetToCelsius();
}
else if (option == ‘2’) {
  convertKilosToMiles();
}
else if (option == ‘3’) {
  convertKnotsToMph();
}
else if (option == ‘0’ ) {
  return 0 ;
}
```

Multiple Selection with `switch`

```
switch (option)
{
  case '1' :  convertFahrenhetToCelsius();
              break;
  case '2' :  convertKilosToMiles() ;
              break;
  case '3' :  convertKnotsToMph() ;
              break;
  case '0' :  return 0 ;
              break;
  default:    printf("%c is invalid\n",option);
              break;
}
```

Ignoring Whitespace in Input

```
switch (option)
{
  case '1'  :  ...
               break;

  case ' '  :              // blank
  case '\n' :              // newline
  case '\t' :              // tab
               break;

  default:     ...
               break;
}
```

```
class IfElseDemo {
public static void main(String[] args) {
    int testscore = 76;
    char grade;
    if (testscore >= 60) { grade = 'D'; }
    else if (testscore >= 70) { grade = 'C'; }
    else if (testscore >= 80) { grade = 'B'; }
    else if  (testscore >= 90) { grade = 'A'; }
    else { grade = 'F'; }
    System.out.println("Grade = " + grade); }
 }
```