Programming with C

Terry Marris November 2010

8 Switch With Case

Previously we looked at the logical operators. Now we continue our study of selections by considering the switch-with-case statement and the conditional operator.

8.1 Switch

Our first program inputs a date in the form *dayNumber*, *monthNumbe*r, e.g. 22 11, and outputs the date in an expanded from e.g. 22nd November. First, we deal with translating month numbers into month names.

```
char string[BUFSIZ];
char month[15];
int mm;

printf("Month (1..12)? ");
gets(string);
mm = atoi(string);

switch (mm) {
case 1:
  strcpy(month, "January");
  break;
case 2:
  strcpy(month, "February");
  break;
case 3:
  strcpy(month, "March");
  break;
...
case 10:
  strcpy(month, "October");
  break;
case 11:
  strcpy(month, "November");
  break;
case 12:
  strcpy(month, "December");
  break;
default:
  printf("Invalid month\n");
  exit(EXIT_FAILURE);
}
...
```

switch() is a bit like a sequence of *if ... else if ...* The value of *mm* is compared with each *case* label, *1, 2, 3*, .. If a match is found, the corresponding action is taken. For example, if *mm* has the value *11*, *November* is copied into *month*, and then the *break* statement results in the immediate skip to the end. *break* breaks out of the switch-with-case statement.

The *default* case is a bit like the *else* leg of an *if ...* statement. It deals with none of the above cases.

If the break statement is left out, then control crashes into the next case. This is sometimes useful, as in the inclusion of a day number suffix, shown below.

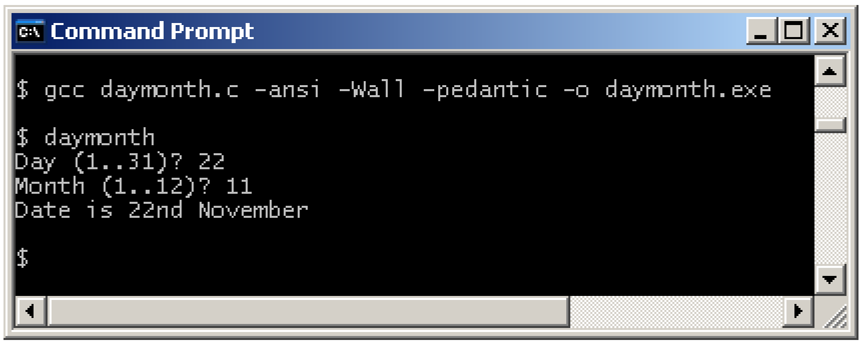
```
        char string[BUFSIZ];
        char suffix[5];
        int dd;

        printf("Day (1..31)? ");
        gets(string);
        dd = atoi(string);

        switch (dd) {
        case 1:
        case 21:
        case 31:
          strcpy(suffix, "st");
          break;
        case 2:
        case 22:
          strcpy(suffix, "nd");
          break;
        case 3:
        case 23:
          strcpy(suffix, "rd");
          break;
        default:
          strcpy(suffix, "th");
          break;
        }
```

case 1, *case 21* and *case 31* are all treated identically: *st* is copied into *suffix*. What if *dd* is less than 0 or more than 31? We deal with that by providing a guard, as shown by the *if ...* statement in the program below.

```
      /* daymonth.c: expands two integers input into a descriptive date */

      #include <stdio.h>
      #include <stdlib.h>
      #include <string.h>

      int main()
      {
        char string[BUFSIZ];
        char month[15], suffix[5];
        int dd, mm;

        printf("Day (1..31)? ");
        gets(string);
        dd = atoi(string);
        printf("Month (1..12)? ");
        gets(string);
        mm = atoi(string);

        switch (mm) {
        case 1:
          strcpy(month, "January");
          break;
        case 2:
          strcpy(month, "February");
          break;
```

```
  case 3:
    strcpy(month, "March");
    break;
  case 4:
    strcpy(month, "April");
    break;
  case 5:
    strcpy(month, "May");
    break;
  case 6:
    strcpy(month, "June");
    break;
  case 7:
    strcpy(month, "July");
    break;
  case 8:
    strcpy(month, "August");
    break;
  case 9:
    strcpy(month, "September");
    break;
  case 10:
    strcpy(month, "October");
    break;
  case 11:
    strcpy(month, "November");
    break;
  case 12:
    strcpy(month, "December");
    break;
  default:
    printf("Invalid month\n");
    exit(EXIT_FAILURE);
  }

  if (dd < 0 || dd > 31) {
    printf("Invalid day\n");
    exit(EXIT_FAILURE);
  }

  switch (dd) {
  case 1:
  case 21:
  case 31:
    strcpy(suffix, "st");
    break;
  case 2:
  case 22:
    strcpy(suffix, "nd");
    break;
  case 3:
  case 23:
    strcpy(suffix, "rd");
    break;
  default:
    strcpy(suffix, "th");
    break;
  }
  printf("Date is %d%s %s\n", dd, suffix, month);
  return 0;
}
```

Here is an example of a program run.



```
$ gcc daymonth.c -ansi -Wall -pedantic -o daymonth.exe

$ daymonth
Day (1..31)? 22
Month (1..12)? 11
Date is 22nd November

$
```

The input to a *switch()* can only be an integer or a *char*. The *case* and *default* statement can occur in any order without affecting the logic. *break*, *exit()* and *return* can all be used to exit from a case, the difference between them is that *return* exits from a function (more on that later), *exit()* terminates program execution, and *break* just exits from the switch-with-case statement. The final break is not really necessary, but is useful if somebody decides to add another case onto the end of your switch-with-case statement.

8.2 The Conditional Operator

The statement

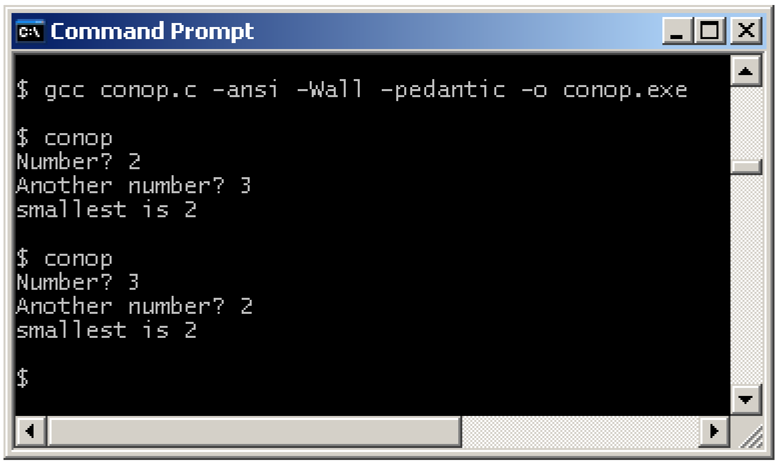
```
if (a < b)
  min = a;
else
  min = b;
```

can also be written

```
min = (a < b)? a : b;
```

(a - b)? is the *Boolean expression*. If true (i.e. non-zero), *a* is copied into *z*. If false, *b* is copied into *min.*.

```
/* conop.c: uses the conditional operator */

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

int main()
{
  char string[BUFSIZ];
  int a, b, min;

  printf("Number? ");
  gets(string);
  a = atoi(string);
  printf("Another number? ");
  gets(string);
  b = atoi(string);
  min = (a < b) ? a : b;
  printf("smallest is %d\n", min);
  return 0;
}
```



```
$ gcc conop.c -ansi -Wall -pedantic -o conop.exe

$ conop
Number? 2
Another number? 3
smallest is 2

$ conop
Number? 3
Another number? 2
smallest is 2

$
```

8.3 Precedence

We look at the precedence of the operators met so far.

Operator	*Description*	*Precedence*
( )	brackets	highest priority
++ --	increment and decrement operators	
* / %	times, divide, mod	
+ -	add, subtract	
< <= > >=	relational operators	
== !-	equality operators	
&&	logical and	
||	logical or	
?:	conditional operator	
=	assignment operator	lowest priority

So, with

```
  min = (a < b) ? a : b;
```

for example, the assignment, =, is executed last.

Exercise 8.1

1. Write and test a program that inputs a grade e.g. A, B, C, D, E and outputs a textual description of that grade e.g. brilliant, very good, good, fair, appalling.

2. A year is a leap year if it is divisible by 4, except that years divisible by 100 are not leap years, but years divisible by 400 are. Design, write and test a program that inputs a year and outputs whether it is a leap year.

We have seen how to use the switch-with-case statement and how to use the logical operator. **Next** we start our study of iterations, also known as repetitions or loops.

Bibliography

Kernighan B and Ritchie D *The C Programming Language* Prentice Hall 1988
Mark Williams Company *ANSI C A Lexical Guide* Prentice Hall 1988