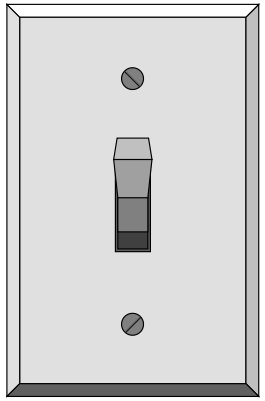
Program to print Day of the week

```
#include <stdio.h>
void main ()
{ int day;
/* ask user for day number */
do
 {
 printf ("Enter a number (1-7): ");
 scanf (" %d", &day);
} while (day!=1 || day!=2 || day!=3 || day!=4 || day!=5 || day!=6 || day!=7);
/* test the alternatives */
if (day ==1) printf (“MONDAY! \n");
else if (day ==2) printf (“TUESDAY! \n");
else if (day ==3) printf (“WEDNESDAY! \n");
else if (day ==4) printf (“THURSDAY! \n");
else if (day ==5) printf (“FRIDAY! \n");
else if (day ==6) printf (“SATURDAY! \n");
else if (day ==7) printf (“SUNDAY! \n");
/* no else case necessary here */
}
```

Comparing Exact Values

Switch Statements



The Switch Statement

- The *switch statement* provides another way to decide which statement to execute next
- The `switch` statement evaluates an expression, then attempts to match the result to one of several possible *cases*
- The match between the expression and the case value must be an exact match.

```
switch ( expression ){
    case value1 :
        statement-list1
    case value2 :
        statement-list2
    case value3 :
        statement-list3
    case  ...

}
```

The Switch Statement

- Each case contains a value and a list of statements
- The flow of control transfers to statement associated with the first case value that matches

```
switch ( expression ){
    case value1 :
        statement-list1
    case value2 :
        statement-list2
    case value3 :
        statement-list3
    case  ...

}
```

Switch - syntax

- The general syntax of a `switch` statement is:

```
switch ( expression ){
    case value1 :
        statement-list1
    case value2 :
        statement-list2
    case value3 :
        statement-list3
    case  ...

}
```

switch and **case** are reserved keywords

Where the expression can be a variable or an arithmetic expression

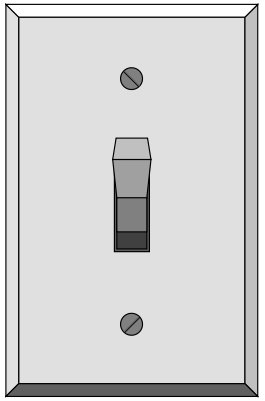
If *expression* matches *value3*, control jumps here and executes statement-list3

Switch - syntax

- The general syntax of a `switch` statement is:

If *expression* matches *value1*, control jumps here and executes statement-list1

```
switch ( expression ){
   case value1 :
      statement-list1
   case value2 :
      statement-list2
   case value3 :
      statement-list3

}
```

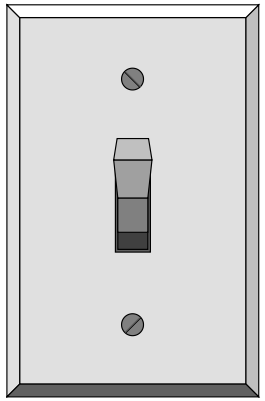


Switch - syntax

- The general syntax of a `switch` statement is:

If *expression* matches *value1*, control jumps here and executes statement-list1 AND statement-list2 AND statement-list3

```
switch ( expression )
{
    case value1 :
        statement-list1
    case value2 :
        statement-list2
    case value3 :
        statement-list3


}
```



The Switch Statement

- The *break statement* can be used as the last statement in each case's statement list
- A `break` statement causes control to transfer to the end of the `switch` statement
- If a `break` statement is not used, the flow of control will continue into the next case

```
switch ( expression ){
    case value1 :
        statement-list1
        break;
    case value2 :
        statement-list2
        break;
    case value3 :
        statement-list3
        break;
    case  ...

}
```

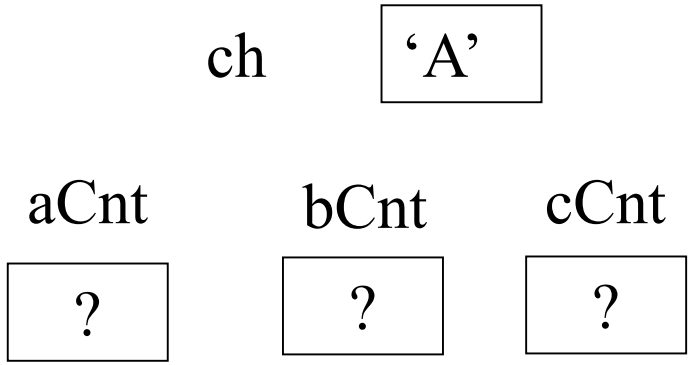
Switch
– breaks OR no breaks!!!

```
switch (ch){
   case 'A':
      aCnt++;
      break;
   case 'B':
      bCnt++;
      break;
   case 'C':
      cCnt++;
      break;
}
```

```
switch (ch){
   case 'A':
      aCnt++;
   case 'B':
      bCnt++;
   case 'C':
      cCnt++;
}
```

Switch
– breaks OR no breaks!!!

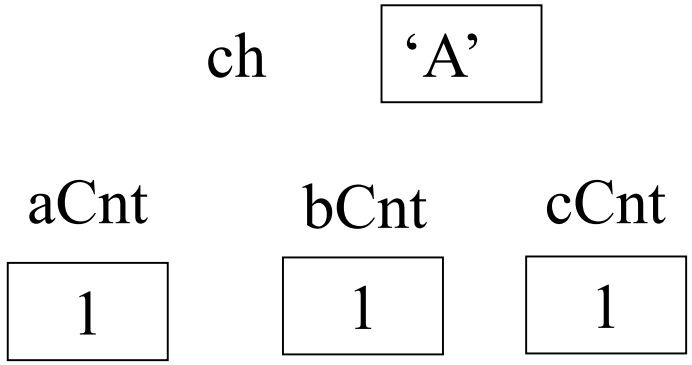
```
char ch;
int aCnt=0, bCnt=0, cCnt=0;
ch=getch();

switch (ch){
   case 'A':
      aCnt++;
   case 'B':
      bCnt++;
   case 'C':
      cCnt++;
}
```



Switch
– breaks OR no breaks!!!

```
char ch;
int aCnt =0, bCnt=0, cCnt=0;
ch=getch();
```

```
switch (ch){
   case 'A':
      aCnt++;
   case 'B':
      bCnt++;
   case 'C':
      cCnt++;
}
```



Switch
– breaks OR no breaks!!!

```
char ch;
int aCnt=0, bCnt=0, cCnt=0;
ch=getch();

switch (ch){
   case 'A':
      aCnt++;
      break;
   case 'B':
      bCnt++;
      break;
   case 'C':
      cCnt++;
      break;
}
```

ch	'A'

aCnt	bCnt	cCnt
?	?	?

Switch
– breaks OR no breaks!!!

```
char ch;
int aCnt=0, bCnt=0, cCnt=0;
ch=getch();

switch (ch){
   case 'A':
      aCnt++;
      break;
   case 'B':
      bCnt++;
      break;
   case 'C':
      cCnt++;
      break;
}
```

ch	'A'

aCnt	bCnt	cCnt
1	0	0

Switch - default

- A **`switch`** statement can have an optional ***default*** *case*
- The default case has no associated value and simply uses the reserved word **`default`**
- If the default case is present, control will transfer to it if no other case value matches
- If there is no default case, and no other value matches, control falls through to the statement after the switch

The switch Statement

- Switch with default case:

```
switch (ch){
    case 'A':
        aCnt++;
        break;
    case 'B':
        bCnt++;
        break;
    case 'C':
        cCnt++;
        break;
    default:
        otherCnt++;
        break;
}
```

ch: ‘D’

aCnt: 0

bCnt: 0

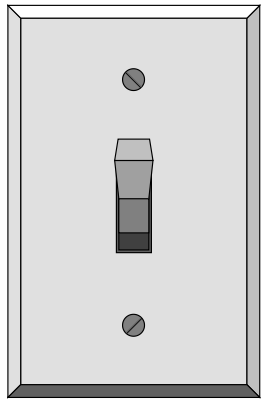
cCnt: 0

otherCnt: 1

NOTE: break is optional in default case when appearing as the last case

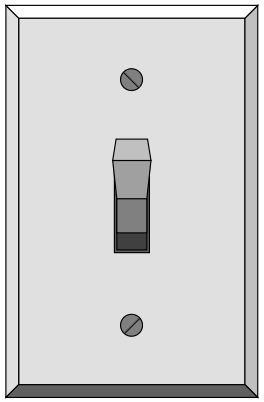
switch Statement: Example

- **If you have scored 95, what grade will you get?**
- **scanf("%d",&score);**



```
switch(score/10){
        case 10:
        case 9: printf("Grade = A");
        case 8: printf("Grade = B");
        case 7: printf("Grade = C");
        case 6: printf("Grade = D");
        default:printf("Grade = F");
}
```

switch Statement: Example



- scanf("%d",&score);

```
switch(score/10){
        case 10:
        case 9: printf("Grade = A");
                    break;
        case 8: printf("Grade = B");
                    break;
        case 7: printf("Grade = C");
                    break;
        case 6: printf("Grade = D");
                    break;
        default:printf("Grade = F");
}
```

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 (“Error - invalid day.\n”) ;
}
```

switch Example 1

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

```
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 ("Error -- invalid day.\n") ;
           break ;
}
```

Example 2: Print corresponding month of the year

```
switch (month) {
case 1 : printf (”January\n” ); break ;
case 2 : printf (”February\n” ); break ;
case 3 : printf (”March\n” ); break ;
case 4 : printf (” April\n” ); break ;
case 5 : printf (”May\n” ); break ;
case 6 : printf (”June\n” ); break ;
case 7 : printf (” July\n” ); break ;
case 8 : printf (”August\n” ); break ;
case 9 : printf (”September\n” ); break ;
case 10: printf (”October\n” ); break ;
case 11: printf (”November\n” ); break ;
case 12: printf (”December\n” ); break ;
default : printf (”No such month\n” ); // Break is not need here}
```

`switch` Statement: Example

```
#include <stdio.h>
int main()
{   char answer;
    printf( "Is Programming Using C an easy course? (y/n):”);
    answer=getch();


    .
    .
    .
    .
    return 0;
}
```

switch Statement: Example 3

```
#include <stdio.h>
int main()
{   char answer;
    printf( "Is Programming Using C an easy course? (y/n):”);
    answer=getch();

    switch (answer)
    {
            .
            .
            .
            .
    }
    return 0;
}
```

switch Statement: Example 3

```
#include <stdio.h>
int main()
{   char answer;
    printf( "Is Programming Using C an easy course? (y/n):”);
    answer=getch();
    switch (answer)
    {
    case 'y': printf("I think so too!“);
              break;
    case 'n': printf("Are you kidding?“);
              break;
    default:
              printf("Is that a yes or no?“);
    }
    return 0;
```

Y or y
N or n

switch Statement: Example 3

```
#include <stdio.h>
int main()
{   char answer;
    printf( "Is Programming Using C an easy course? (y/n):”);
    answer=getch();
    switch (answer)
    {
    case 'Y':
    case 'y': printf("I think so too!“);
              break;
    .
    .
    .
    }         return 0;
}
```

switch Statement: Example 3

```
#include <stdio.h>
int main()
{   char answer;
    printf( "Is Programming Using C an easy course? (y/n):”);
    answer=getch();
    switch (answer)
    { case 'Y':
    case 'y': printf("I think so too!“);
              break;
    case 'N':
    case 'n': printf("Are you kidding?“);
              break;
    default:
              printf("Is that a yes or no?“);
    }         return 0;
}
```

`switch` Statement : Example 4

If watts = 25

→ lifespan =2500

If watts = 40 OR 60

→ lifespan =1000

If watts = 75

→ lifespan =700

Otherwise

→ lifespan = 0

`switch` Statement with Multiple Labels:

If watts = 25
→ lifespan =2500
If watts = 40 OR 60
→ lifespan =1000
If watts = 75
→ lifespan =700
Otherwise
→ lifespan = 0

```
switch (watts) {
    case 25 :  lifespan = 2500;
        break;
    case 40 :
    case 60 :  lifespan = 1000;
        break;
    case 75 :  lifespan = 750;
        break;
    default :
        lifespan = 0;
} // end switch
```

Example 5: To find if the entered character is a digit or not

```
#include <stdio .h>
int main () {
char c ;
scanf (” %c” , &c );
switch (c) {
case ’0 ’ :
case ’1 ’ :
case ’2 ’ :
case ’3 ’ :
case ’4 ’ :
case ’5 ’ :
case ’6 ’ :
case ’7 ’ :
case ’8 ’ :
case ’9 ’ : printf (”%c is a Digit \n” , c );      break;
default : printf (”%c is not a digit \n” , c ); } }
```

To Switch or not to Switch

- The **expression** of a **`switch`** statement must result in an ***integral type***, meaning an **integer** (`byte`, `short`, `int`, `long`) or a **`char`**
- It **cannot** be a **`Boolean`** value or a **floating point** value (`float` or `double`)
- The implicit **Boolean** condition in a **`switch`** statement is **equality**
- **You cannot perform relational checks with a `switch` statement**

Creating Menus/ Menu Driven Program

- When you want to give your user a choice on what to do next, you can display a set of choices (a menu). The user then enters his or her choice. You must validate the choice to make sure it is valid before you continue the program!

Example 6: Sample Program

```
int choice;
choice = 0;
do
{
    printf( "my menu\n\n" );
    printf( "1 – Find number is Prime or not\n" );
    printf( "2 – Find the number is Armstrong or not\n" );
    printf( "3 – Find the Factorial of number\n" );
    printf( "4 – Exit\n" );
    printf( "enter your choice: " );
    scanf( "%d", &choice  );
switch( choice )
{    case 1: printf( "Prime or Not\n" );
                //code for finding the Prime number
            break;
    case 2:  printf( "Armstrong Number or Not\n" );
            //code for finding the Armstrong NUMBER
            break;
    case 3:  printf( "Factorial of the number\n" );
            //code for finding the factorial of the Number
            break;
    case 4:  printf( "EXIT\n" );
            return 0;
    default: printf( "Invalid choice!\n" );
            break;
} } while ( ( choice >= 1 ) && ( choice <= 4 ) );}
```

Example 7: Sample Program: Menu Driven Program for CALCULATOR

```
int main ()
{
char operator ;
int op1 , op2 ;
printf (”Enter operator (+ ,−,∗,/) , q to quit : ” );
scanf (”%c” , &operator );
if (operator == ’q ’ ) return 0; // Exit from main
else
{
printf (”Enter operands : ” );
scanf (”%d %d” , &op1 , &op2 ); }
}
```

```
switch (operator)
 {
case ’+’ : printf (”%d %c %d = %d\n” , op1 , operator , op2 , op1 + op2 );
            break ;
case ’−’ : printf (”%d %c %d = %d\n” , op1 , operator , op2 , op1 − op2 );    break ;
case ’∗ ’ : printf (”%d %c %d = %d\n” , op1 , operator , op2 , op1 ∗ op2 );          break ;
case ’/ ’ : if (op2 != 0)
            printf (”%d %c %d = %d\n” , op1 , operator, op2 , op1 / op2 );
          else   printf (” Division by 0 not possible\n” );
            break ;
default : printf (”Invalid operator\n” );
}
```

Menu driven programs Exercise 8 & 9

- Write a program in C which is a Menu-Driven Program to compute the area of the various geometrical shape.
- Write a program in C which is a Menu-Driven Program to perform a simple calculation on any two numbers. (Addition, subtraction, multiplication, division)

switch Statement summary

- 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.

switch Statement summary

- A switch statement will compile without a default case, but always consider using one.
  - 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.
  - A break is not required in default case when appearing as the last option

The Tips and Traps

- A few useful tips about the usage of switch and a few pitfalls to be avoided:

(a) The earlier program that used switch may give you the wrong impression that you can use only cases arranged in ascending order, 1, 2, 3 and default.

- You can, in fact, put the cases in any order you please.

The cases in any order you please

```
# include <stdio.h>
int main( )
{
int i = 22 ;
switch ( i ) {
case 121 : printf ( "I am in case 121 \n" ) ; break ;
case 7 : printf ( "I am in case 7 \n" ) ; break ;
case 22 : printf ( "I am in case 22 \n" ) ; break ;
default : printf ( "I am in default \n" ) ;
}
return 0 ; }
```

The output of this program would be:
I am in case 22

The Tips and Traps

(b) You are also allowed to use char values in case and switch as shown in the following program:

```
 # include <stdio.h>
int main( ) {
char c = 'x' ;
switch ( c ) {
case 'v' : printf ( "I am in case v \n" ) ; break ;
case 'a' : printf ( "I am in case a \n" ) ; break ;
 case 'x' : printf ( "I am in case x \n" ) ; break ;
default : printf ( "I am in default \n" ) ; } return 0 ; }
```

The output:

I am in case x

In fact here when we use ‘v’, ‘a’, ‘x’ they are actually replaced by the ASCII values (118, 97, 120) of these character constants.

The Tips and Traps

(c) At times we may want to execute a common set of statements for multiple cases.

```
# include <stdio.h>
int main( ) {
char ch ;
printf("Enter any one of the alphabets a, or b") ;
scanf("%c", &ch ) ;
switch( ch ) {
case 'a' :
case 'A' : printf ( "a as in ashar\n" ) ; break ;
case 'b' :
case 'B' : printf ( "b as in brain\n" ) ; break ;
default : printf ( "wish you knew what are alphabets\n" ) ; }
return 0 ; }
```

Here, we are making use of the fact that once a case is satisfied; the control simply falls through the switch till it doesn't encounter a break statement.
That is why if an alphabet a is entered, the case 'a' is satisfied and since there are no statements to be executed in this case, the control automatically reaches the next case, i.e., case 'A' and executes all the statements in this case.

The Tips and Traps

(d) Even if there are multiple statements to be executed in each case, there is no need to enclose them within a pair of braces (unlike if and else).

The Tips and Traps

(e) Every statement in a switch must belong to some case or the other. If a statement doesn’t belong to any case, the compiler won’t report an error. However, the statement would never get executed.

The Tips and Traps

For example, in the following program, the printf( ) never goes to work:

```
# include <stdio.h>
int main( ) {
int i, j ;
printf ( "Enter value of i" ) ;
scanf ( "%d", &i ) ;
switch ( i ) {
printf ( "Hello\n" ) ;
case 1 : j = 10 ; break ;
case 2 : j = 20 ; break ;
}
return 0 ; }
```

The Tips and Traps

(f) If we have no default case, then

- the program simply falls through the entire switch and continues with the next instruction (if any,) that follows the closing brace of switch.

The Tips and Traps

(g) Is switch a replacement for if?

Yes and no.

Yes, because it offers a better way of writing programs as compared to if, and no, because, in certain situations, we are left with no choice but to use if.

The disadvantage of switch is that one cannot have a case in a switch which looks like: **case i <= 20 :**

All that we can have after the case is an int constant or a char constant or an expression that evaluates to one of these constants.

Even a float is not allowed.

The advantage of switch over if is that it leads to a more structured program and the level of indentation is manageable, more so, if there are multiple statements within each case of a switch.

The Tips and Traps

(h) We can check the value of any expression in a switch. Thus, the following switch statements are legal:

switch (i + j * k)

switch (23 + 45 % 4 * k)

switch (a < 4 && b > 7)

Expressions can also be used in cases provided they are constant expressions.

Thus, case 3 + 7 is correct, however, case a + b is incorrect.

The Tips and Traps

(i) The break statement when used in a switch takes the control outside the switch. However, use of continue will not take the control to the beginning of switch as one is likely to believe. This is because switch is not a looping statement unlike while, for or dowhile.

(j) In principle, a switch may occur within another, but in practice, this is rarely done. Such statements would be called nested switch statements.

(k) The switch statement is very useful while writing menu driven programs

The Tips and Traps

(l) Cases can never have variable expressions (for example, it is wrong to say **case a +3 :**).

(m) Multiple cases cannot use same expressions. Thus the following switch is illegal:

```
switch ( a )
{
case 3 : ...
case 1 + 2 : ...
}
```

Why Use a switch Statement?

DIY:

- Make a comparison chart between IF and Switch Statement.
- Make a list of cases, where switch statement should be used and the cases where if statement should be used.
- Menu-Driven Program to compute the area of the various geometrical shape.
- Program to Simulate the traffic lights

Menu-Driven Program to compute the area of the various geometrical shape.

```
#include <stdio.h>
void main ()
{ int choice,r,l,w,b,h;
float area;
printf("Input 1 for area of circle\n");
printf("Input 2 for area of rectangle\n");
printf("Input 3 for area of triangle\n");
printf("Input your choice : ");
scanf("%d",&choice);
if(choice == 1)    {
         printf("Input radious of the circle : ");
         scanf("%d",&r);
         area=3.14*r*r; }
else if(choice ==2) {
         printf("Input length and width of the rectangle : ");
         scanf("%d %d",&l,&w);
         area=l*w;}
else if(choice ==3) {
         printf("Input the base and height of the triangle :");
         scanf("%d%d",&b,&h);
         area=0.5*b*h; }
printf("The area is : %f\n",area); }
```

DIY: Re-write the Program using switch - case

Exercise 10: Program to Simulate the traffic lights

DIY: Re-write the program using switch-case

```
#include <stdio.h>
void main ()
{
char colour;
/* ask user for colour */
do  {
 printf ("Enter the colour of the light (R,G,Y): ");
 scanf (" %c", &colour); /* very important to leave space before %c here */
 } while (colour!='R' && colour!='r' && colour!='G' && colour!='g' && colour!='Y' && colour!='y' );
/* test the alternatives */
if (colour =='R‘ || colour == 'r‘) /* red light */
            printf ("STOP! \n");
else if (colour == 'Y‘ || colour == 'y‘) /* yellow */

            printf ("CAUTION! \n");
else if( colour == 'G‘ || colour == 'g‘) /* green light */
           printf ("GO! \n");
 /* no else case necessary here */
} }
```

```
Switch(colour)
{
case ‘R’:
Case ‘r ‘: printf(“Stop”);
break;
```