CS F111: Computer Programming

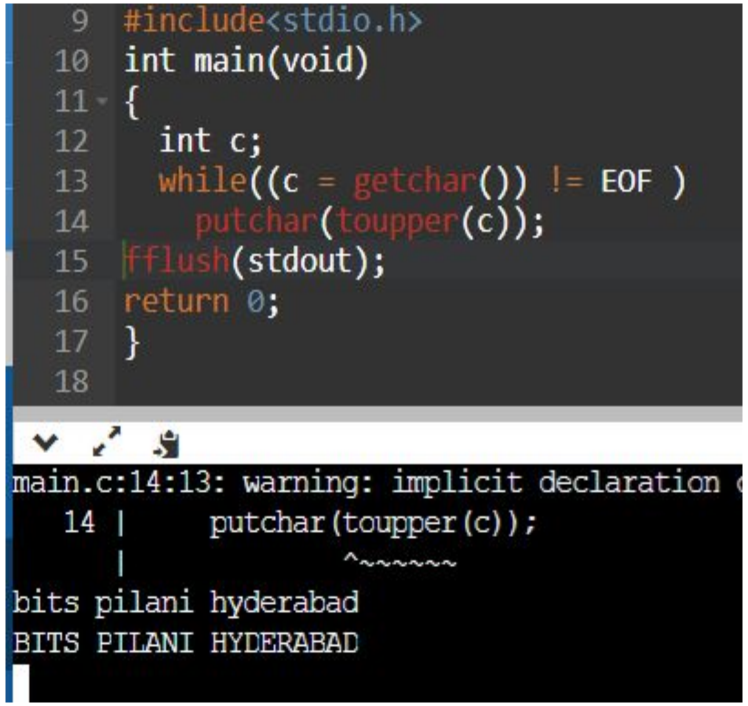
(Second Semester 2020-21)

Lect 12: Switch, Goto

Dr. Nikumani Choudhury
Asst. Prof., Dept. of Computer Sc. & Information Systems
nikumani@hyderabad.bits-pilani.ac.in

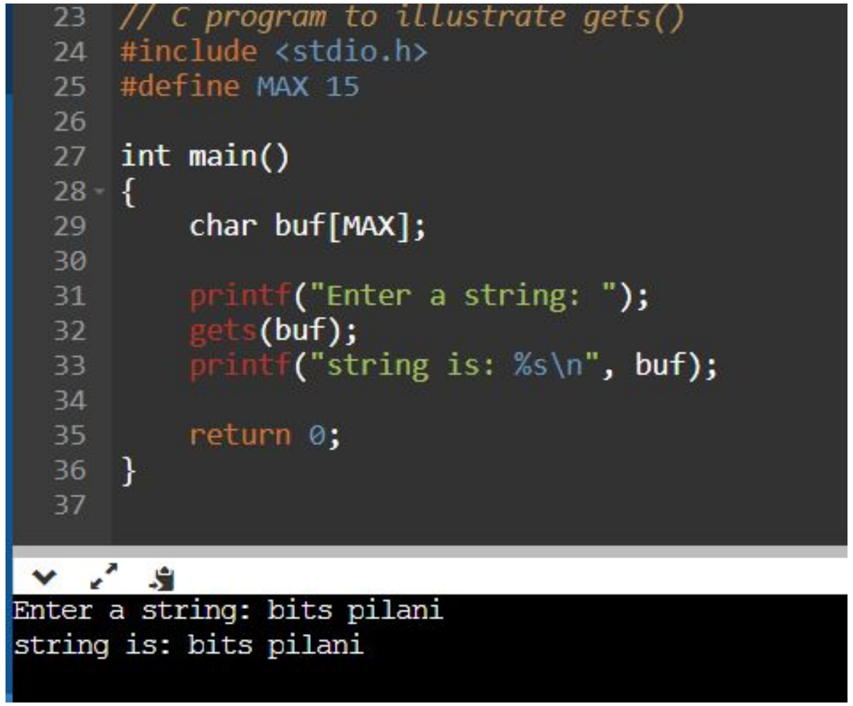
BITS Pilani

Hyderabad Campus



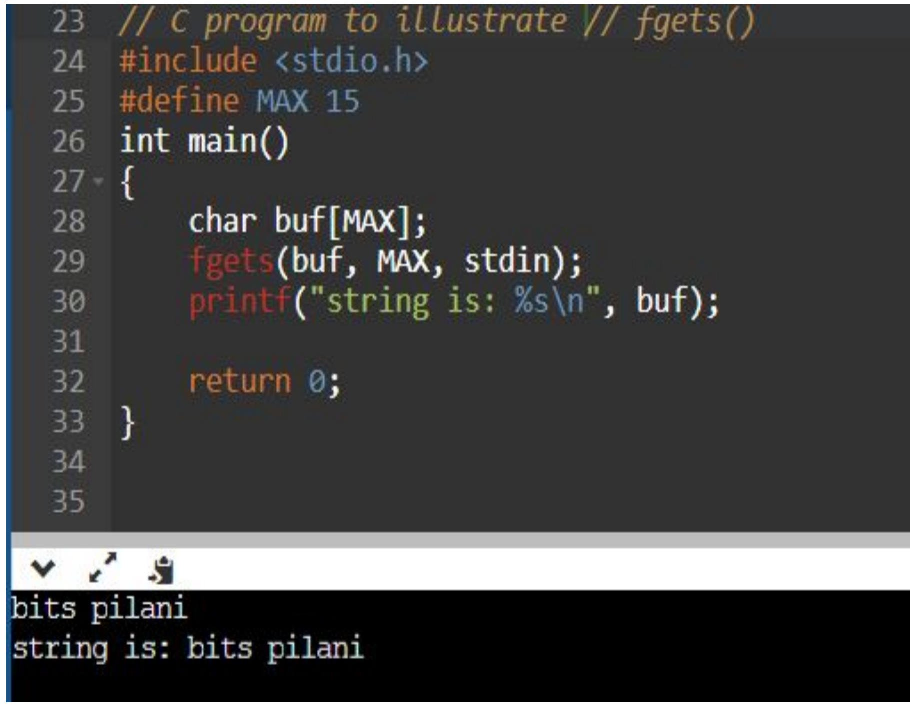
```c
#include<stdio.h>
int main(void)
{
  int c;
  while((c = getchar()) != EOF )
    putchar(toupper(c));
fflush(stdout);
return 0;
}
```

```
main.c:14:13: warning: implicit declaration
   14 |     putchar(toupper(c));
      |             ^~~~~~~
bits pilani hyderabad
BITS PILANI HYDERABAD
```



```c
// C program to illustrate gets()
#include <stdio.h>
#define MAX 15

int main()
{
    char buf[MAX];

    printf("Enter a string: ");
    gets(buf);
    printf("string is: %s\n", buf);

    return 0;
}
```

```
Enter a string: bits pilani
string is: bits pilani
```



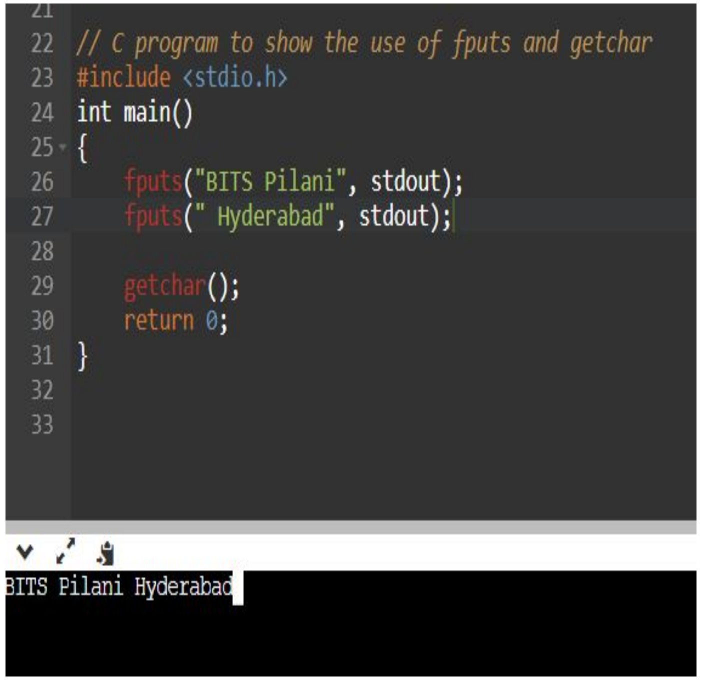
```c
// C program to illustrate // fgets()
#include <stdio.h>
#define MAX 15
int main()
{
    char buf[MAX];
    fgets(buf, MAX, stdin);
    printf("string is: %s\n", buf);

    return 0;
}
```

```
bits pilani
string is: bits pilani
```

puts(str) vs printf(str);

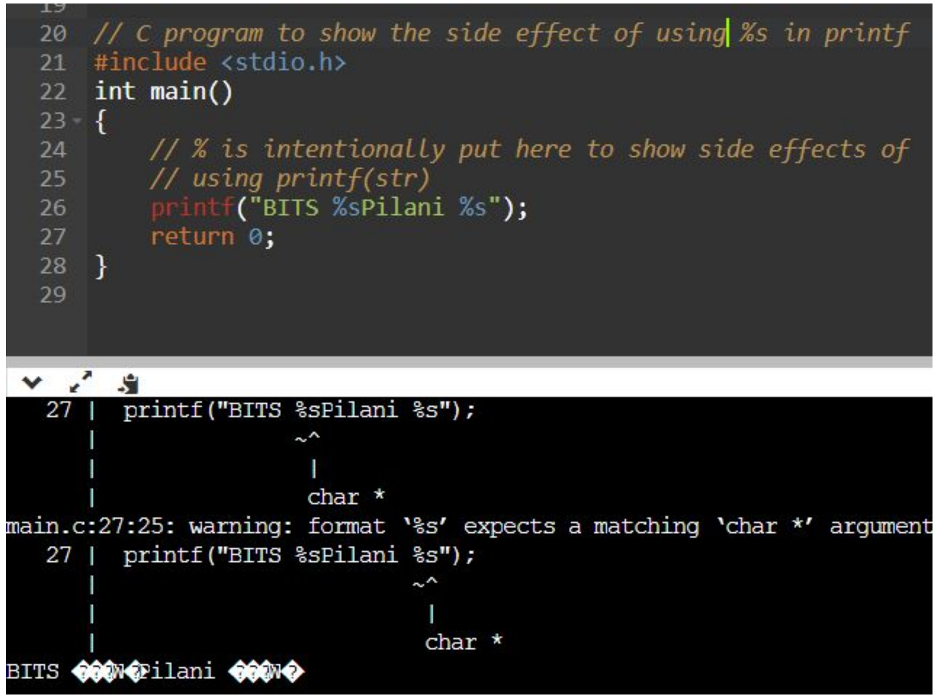
- puts() can be preferred for printing a string because it is generally less expensive (implementation of puts() is generally simpler than printf())
- if the string has formatting characters like ‘%s’, then printf() would give unexpected results. Also, if str is a user input string, then use of printf() might cause security issues
- puts() moves the cursor to next line. If you do not want the cursor to be moved to next line, then you can use following variation of puts().

fputs(str, stdout)



```c
// C program to show the use of fputs and getchar
#include <stdio.h>
int main()
{
    fputs("BITS Pilani", stdout);
    fputs(" Hyderabad", stdout);

    getchar();
    return 0;
}
```

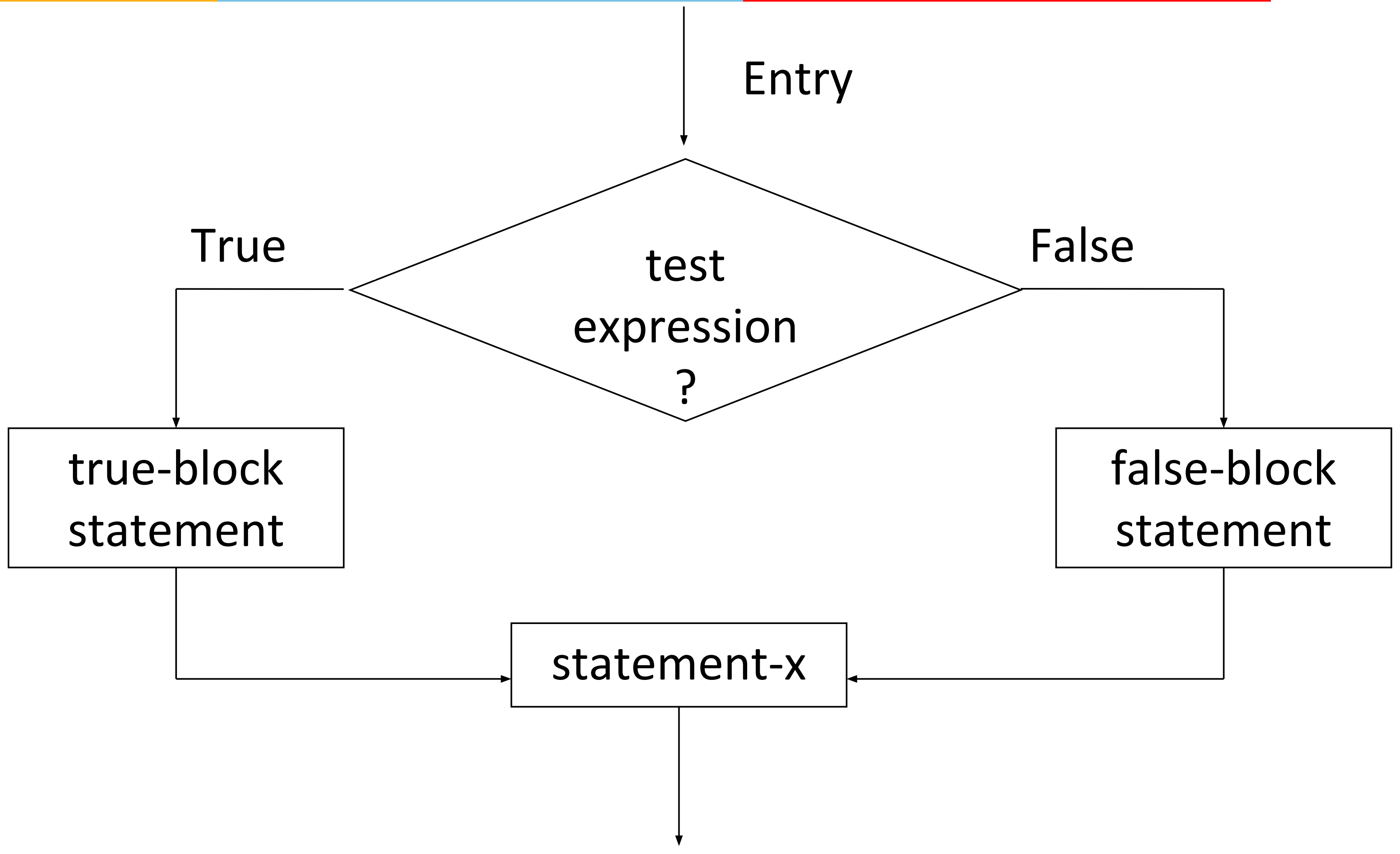
```
BITS Pilani Hyderabad
```



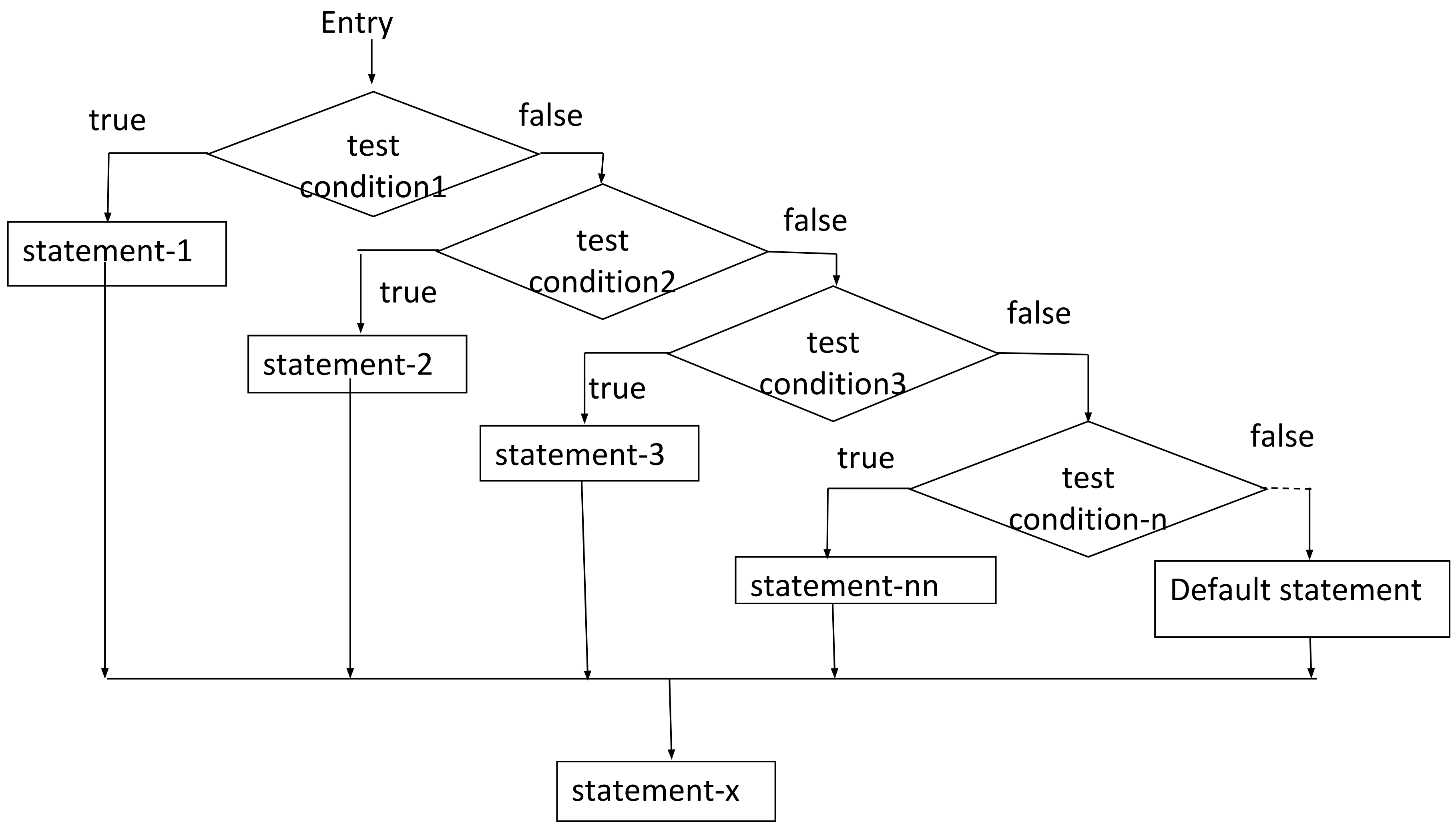
```c
// C program to show the side effect of using %s in printf
#include <stdio.h>
int main()
{
    // % is intentionally put here to show side effects of
    // using printf(str)
    printf("BITS %sPilani %s");
    return 0;
}
```

```
   27 |  printf("BITS %sPilani %s");
      |                ~^
      |                 |
      |                 char *
main.c:27:25: warning: format ‘%s’ expects a matching ‘char *’ argument
   27 |  printf("BITS %sPilani %s");
      |                        ~^
      |                         |
      |                         char *
BITS ���N�Pilani ���N�
```

Flowchart of if……else control



The else if ladder



A Word of Caution

What will be the output of the following programs:

```
int main( )
{
  int i;
  printf (“Enter the value of i:”);
  scanf (“%d”, &i);
  if ( i = 7 )  // should have been ==
    printf (“You entered 7”);
  else
    printf (“You entered other than 7”);
}
```

```
Enter the value of i:45
You entered 7
```

```
Enter the value of i:5
You entered 7
```

```
int main ( )
{
   int i;
   printf (“Enter value of i”);
   scanf (“%d”, &i);
   if (7 == i);
       printf (“You entered 7”);
}
```

```
if (7==i)
      ;
printf (“You entered 7”);
```

Find errors, if any, in each of the following segments:

```
if ( x + y = z )
    printf (“ \n”);
```

```
if  ( x <0 ) || (y<0)
    printf (“ sign is negative”);
```

```
if ( x > 1 ) ;
    x ++ ;
else
    x=0
```

```
if ( x > 1 )
    x ++ ;
    printf (“%d”, x);
else
    x=0 ;
```

What is the output of the following C segment:

```
x = 120 ;
y = 30;
if (( x > 100) && (y = 50))
   z = x + y;
else
   z = x - y;
printf( " x=%d, y=%d, z=%d\n", x, y, z);
```

Short-circuiting Concept

- &&

```
if(e1 && e2)
{
    set of statements
}
```

if e1 is false then e2 will not be evaluated.

- ||

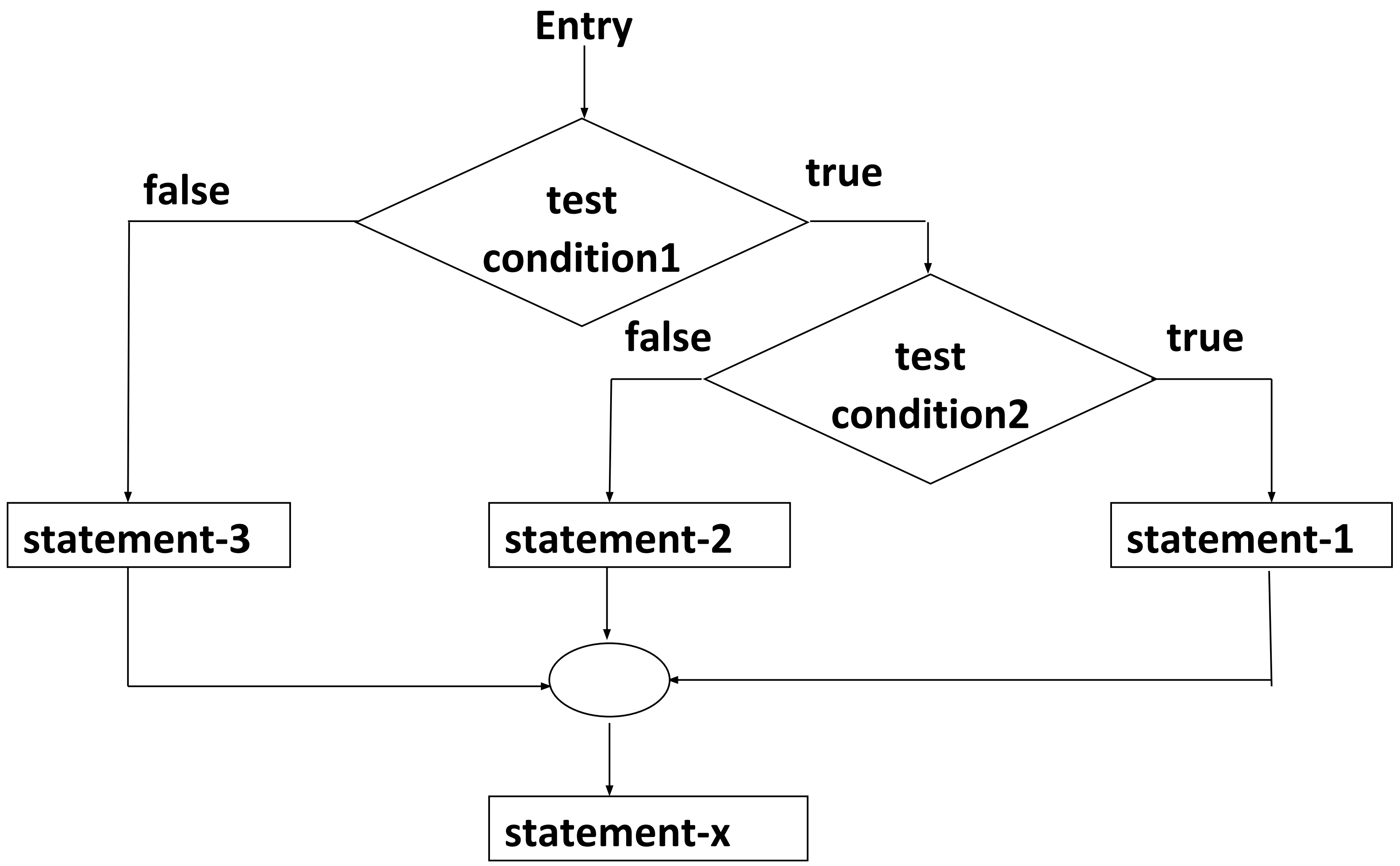
```
if(e1 || e2)
{
    set of statements
}
```

if e1 is true then e2 will not be evaluated.

Nesting of if……else statement

```
if ( test condition-1 )
{
   if ( test condition-2 )
   {
  statement-1;
   }
   else
   {
  statement-2;
   }
}
else
{
   statement-3;
}
statement-x;
```

Nesting of if……else



Largest of the three numbers

```
#include<stdio.h>
main()
{
   float a,b,c;
   printf("Enter three values\n");
   scanf("%f%f%f",&a,&b,&c);
   printf("\nLargest value is ");
   if(a>b)
   {
    if(a>c)
        printf("%f\n",a);
    else
        printf("%f\n",c);
   }
   else
   {
    if(c>b)
        printf("%f\n",c);
    else
        printf("%f\n",b);
   }
}
```

Nesting of if……else statement

- Note: else is always paired with the most recent unpaired if.

```
if (a >= 10)
    if (a < 20)
        a = a + 2;
else
        a = a + 1;
```

Dangling Else

```
if (x != 10)
  if (y > 3)
    z = z / 2;
else
    z = z * 2;
```

Else is always associated with closest unassociated if.

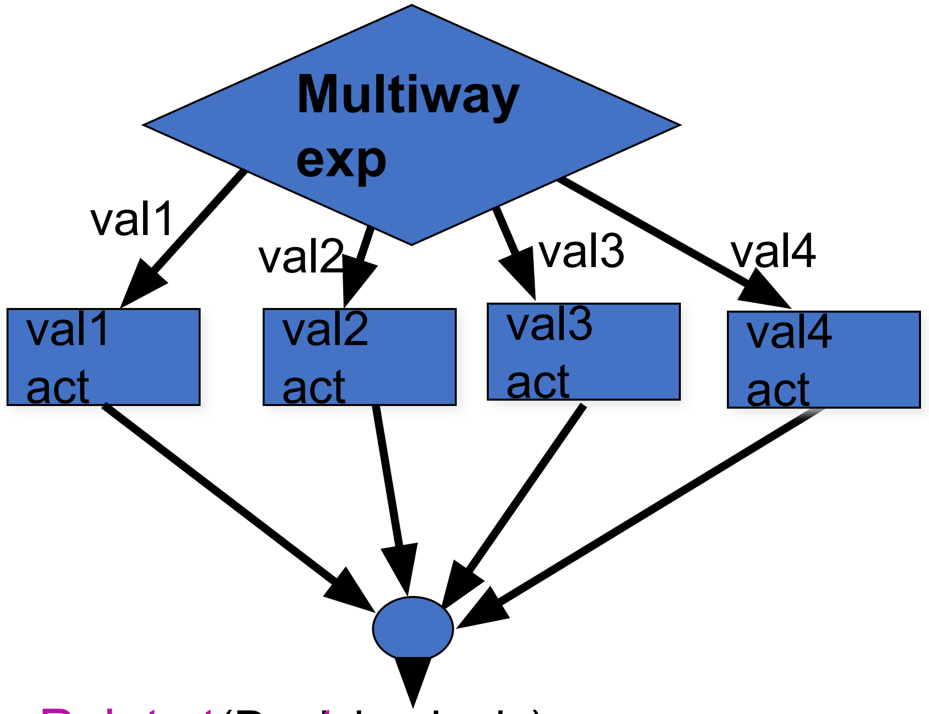
is the same as...

```
if (x != 10) {
  if (y > 3)
    z = z / 2;
  else
    z = z * 2;
}
```

is NOT the same as...

```
if (x != 10) {
  if (y > 3)
    z = z / 2;
}
else
    z = z * 2;
```

Multi-way selection : switch



(Decision logic)

```
switch (expression)
 {
   case const1   : statement
                   …
                   statement
   case const2   : statement
                   …
                   statement
   …
   default     : statement
              …
              statement
```

Points to note:

- No two case labels can have the same constant value.
- Two case labels can be associated with same set of actions.
- Default case is optional (only one default), but may be anywhere.

The switch statement

- The switch statement is a multi-way decision that tests whether an expression matches one of a number of *constant* values, and branches accordingly.

```
switch (expression)
{
  case value1:
      block1
      break;
  case value2:
      block2
      break;
  ……
  default:
      default-block
      break;
 }
statements-x;
```

The expression is an integer/char expression or characters.
value1 value2…are integer-valued/char constant expressions.

The switch statement

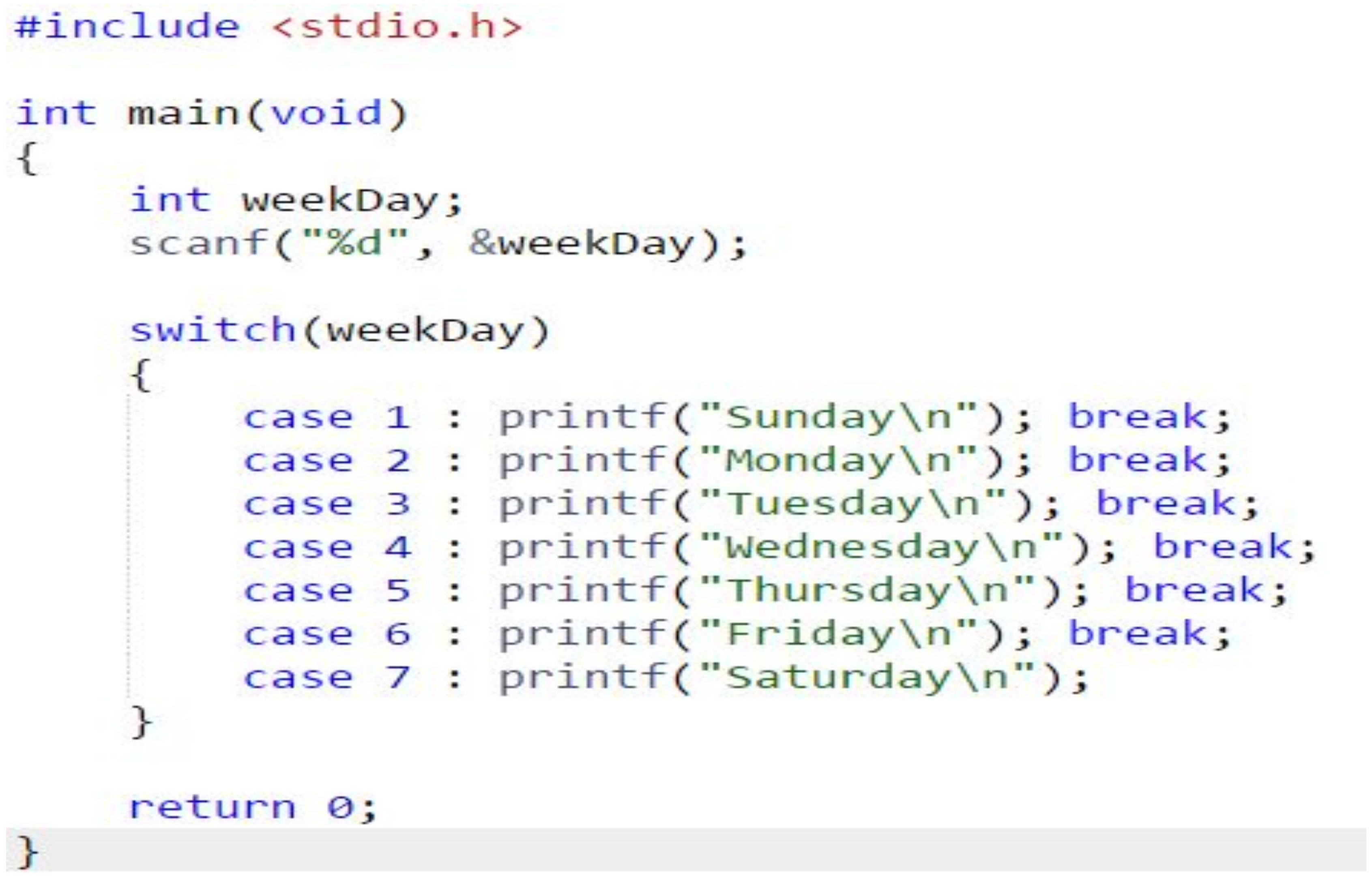
- Cases and the default clause can occur in any order.
- The break statement at the end of each block signals the end of a particular case and causes an exit from the switch statement, transferring the control to the statement following the switch.

break statement

- Used to exit from a switch or terminate from a loop

- With respect to switch, the break statement causes a transfer of control out of the entire switch statement, to the first statement following the switch statement.

Problem: Week of the Day

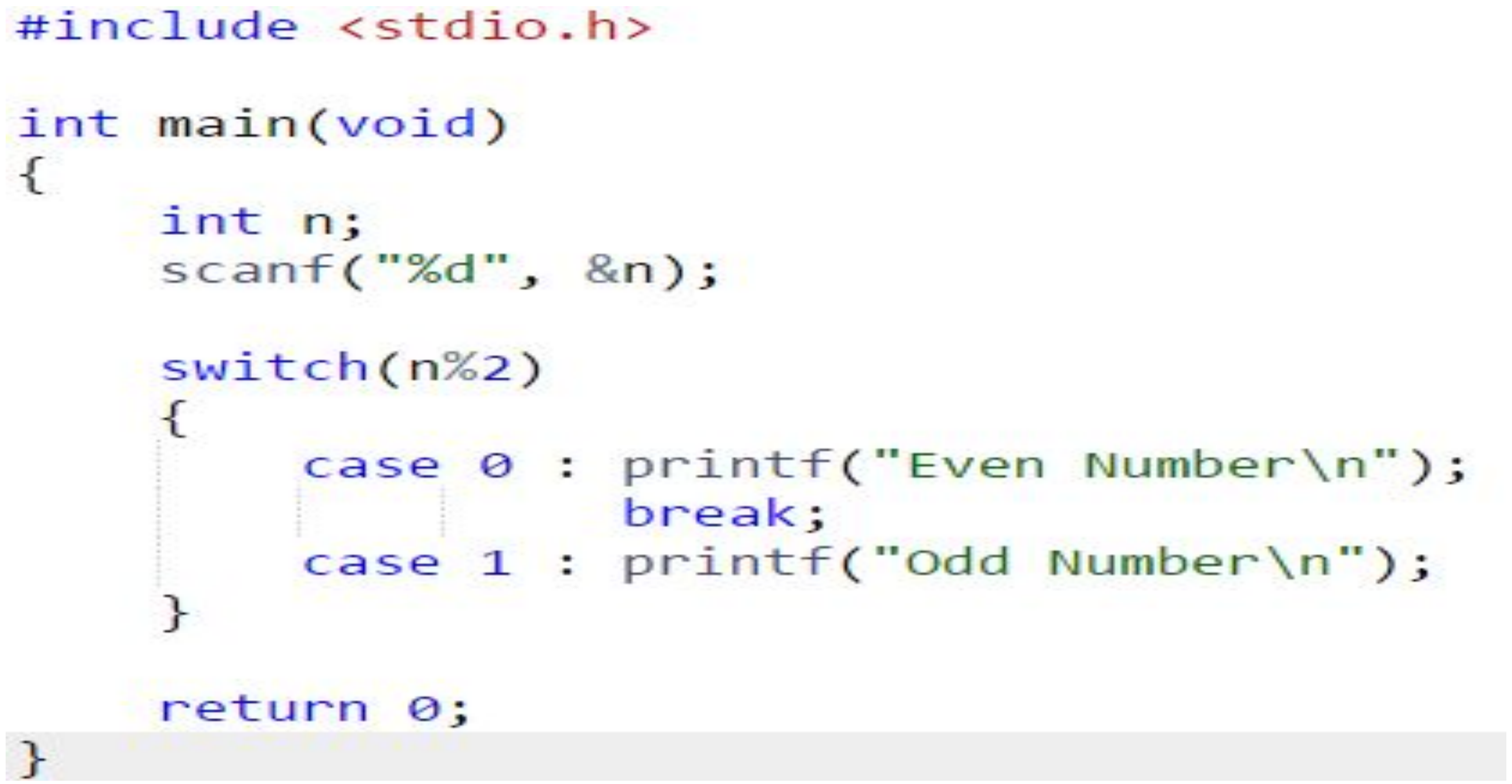


```c
#include <stdio.h>

int main(void)
{
	int weekDay;
	scanf("%d", &weekDay);

	switch(weekDay)
	{
		case 1 : printf("Sunday\n"); break;
		case 2 : printf("Monday\n"); break;
		case 3 : printf("Tuesday\n"); break;
		case 4 : printf("Wednesday\n"); break;
		case 5 : printf("Thursday\n"); break;
		case 6 : printf("Friday\n"); break;
		case 7 : printf("Saturday\n");
	}

	return 0;
}
```

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

   printf("\t--TRAVEL GUIDE--\n\n");
   printf("A  Air Timings\n");
   printf("T  Train Timings\n");
   printf("B  Bus Service\n");
   printf("X  To skip\n");

   printf("\n  Enter your choice\n");

   scanf("c", &choice);

   switch(choice)
   {
    case 'A':
        printf("You select by Air\n"); break;
    case 'B':
        printf("You select by train\n");break;
    case 'T':
        printf("You select by bus\n");break;
    case 'X':
        printf("You skip\n"); break;
    default:
        printf("No choice\n");
   }
}
```

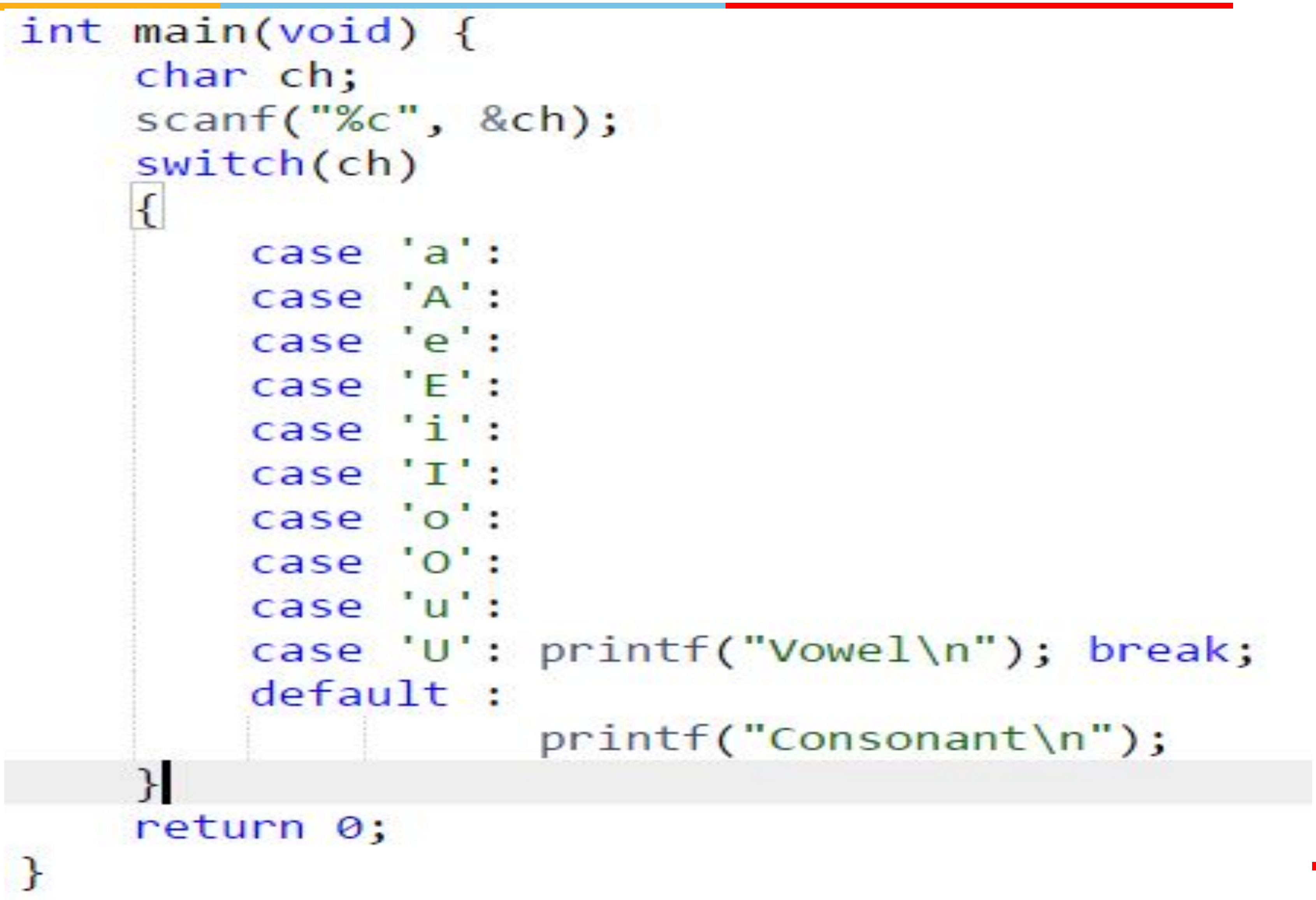
The switch statement is often used for menu selection.

Problem: Even or odd



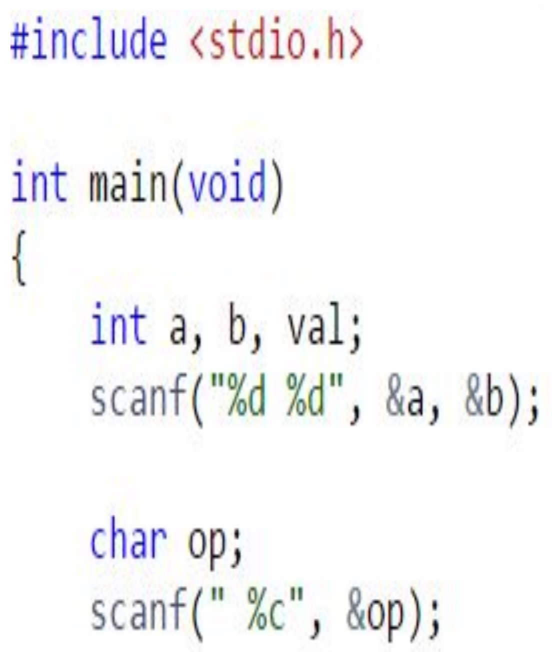
```
#include <stdio.h>

int main(void)
{
        int n;
        scanf("%d", &n);

        switch(n%2)
        {
                case 0 : printf("Even Number\n");
                         break;
                case 1 : printf("Odd Number\n");
        }

        return 0;
}
```

Problem: Vowel

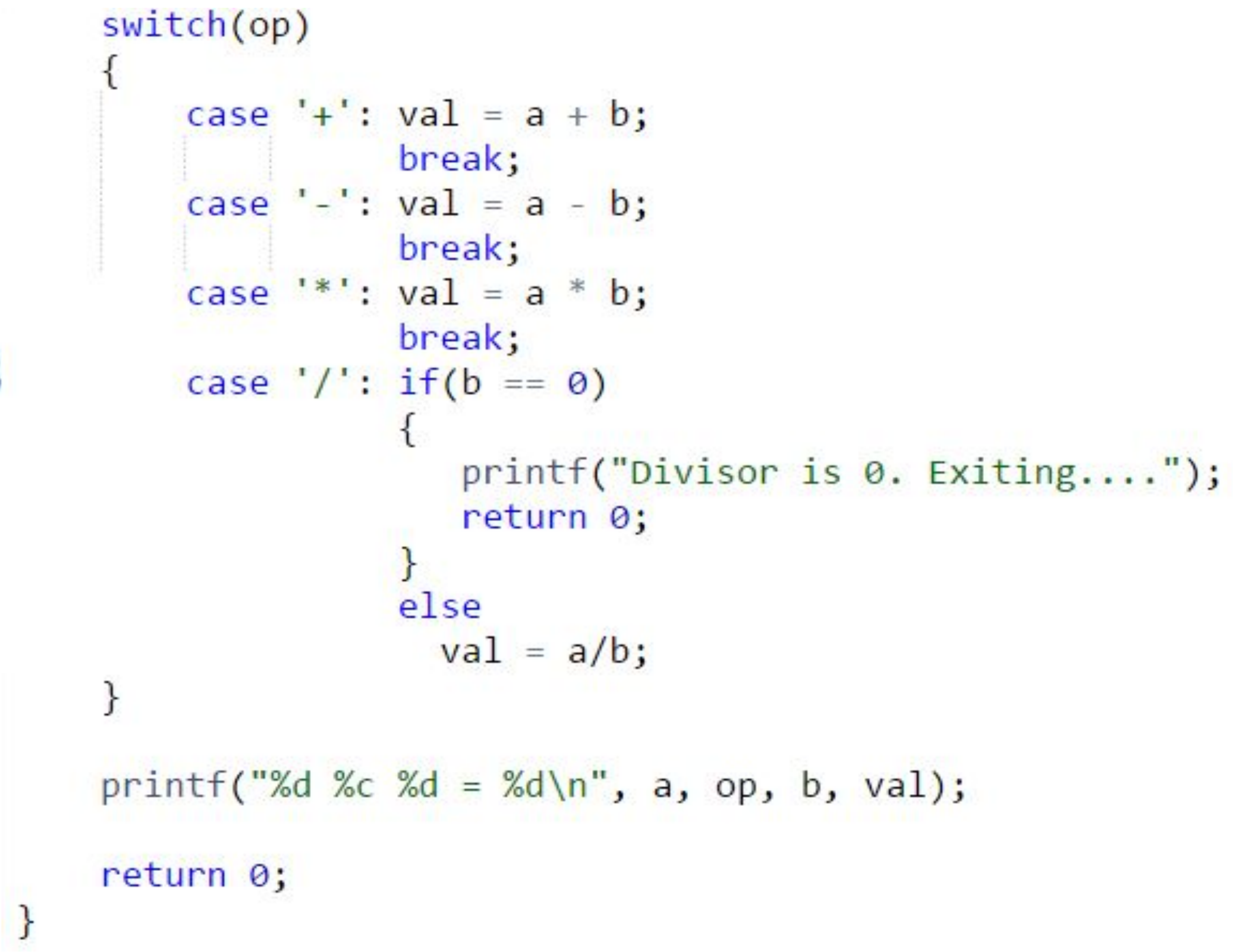


```
int main(void) {
    char ch;
    scanf("%c", &ch);
    switch(ch)
    {
        case 'a':
        case 'A':
        case 'e':
        case 'E':
        case 'i':
        case 'I':
        case 'o':
        case 'O':
        case 'u':
        case 'U': printf("Vowel\n"); break;
        default :
                  printf("Consonant\n");
    }
    return 0;
}
```

Problem : Operator



```c
#include <stdio.h>

int main(void)
{
    int a, b, val;
    scanf("%d %d", &a, &b);

    char op;
    scanf(" %c", &op);
```



```c
    switch(op)
    {
        case '+': val = a + b;
                  break;
        case '-': val = a - b;
                  break;
        case '*': val = a * b;
                  break;
        case '/': if(b == 0)
                  {
                      printf("Divisor is 0. Exiting....");
                      return 0;
                  }
                  else
                     val = a/b;
    }

    printf("%d %c %d = %d\n", a, op, b, val);

    return 0;
}
```

What is the ouput if ch = A?

```
#include <stdio.h>
int main ()
{
        char ch;
        scanf(“%c”, &ch);
        switch(ch)
        {
          case 'A':
                  printf ("Excellent\n");
          case 'B':
                  printf ("Good\n");
          case 'T':
                  printf ("Eh\n");
          case 'X':
                  printf ("Failed\n");

        }

}
```

```
#include <stdio.h>
int main ()
{
        char ch;
        scanf(“%c”, &ch);
        switch(ch)
        {
        case 'A':
                printf ("Excellent\n");
        case 'B':
                printf ("Good\n");break;
        case 'D':
                printf ("I guess");break;
        case 'T':
                printf ("Eh\n");
        case 'X':
                printf ("Failed\n");

        }
```

What would be the output?

```
#include <stdio.h>
int main ()
{
  int i = 3;
  switch (i)
  {
    default:
            printf (“\n A mouse is an elephant”);
     case 1:
            printf(“\n Right Practice makes a man perfect”);
            break;
     case 2:
            printf(“\n money is the root of all wealth”);
   }
}
```

Output:

```
A mouse is an elephant
Right Practice makes a man perfect
```

What is the ouput if ch=B?

```
#include <stdio.h>
int main ()
{
      char ch;
      scanf(“%c”, &ch);
      switch(ch)
      {
        case 'A':
        case 'B':
        case 'C':
        case 'D':
                printf ("Passes");break;
        case 'T':
        default:
                printf ("Failed\n");
      }
}
```

Output:
Passes

What is the output ???

```
int a=1， b=0；
switch(a)
{
    case 1：
        switch ( b )
        {
            case 0： printf ( “**0**” )； break ；
            case 1： printf ( “**1**”)；  break ；
        }
        break;
    case 2： printf ( “** 2 **”) ； break ；
}
```

Rules for switch statement

- The switch expression must be an integer/char type.
- Case labels must be constants or constant expressions.
- Case labels must be unique.
- Case labels must end with colon.
- The break statement transfers the control out of the switch statements.
- The break statement is optional. That is, two or more case labels may belong to the same set of statements.
- The default label is optional. If present, it will be executed when the expression does not find a matching case label.
- There can be at most one default label.
- The default may be placed anywhere but usually placed at the end.
- It is permitted to nest switch statements.

Find out the errors if any…

```
#include <stdio.h>
int main ()
{
   int i = 2, j = 2;
   switch (i)
   {
      case 1:
                  printf("\n Practice makes a man perfect");
                  break;
       case j:
                  printf("\n Money is the root of all wealth");
                  break;
    }
}
```

j is integer var.

Error: case label does not reduce to an integer constant

Find out the errors if any…

```
#include <stdio.h>
int main ()
{
   int i = 2;
   const int j = 2;
   switch (i)
   {
      case 1:
                  printf(“\n Practice makes a man perfect”);
                  break;
       case j:
                  printf(“\n Money is the root of all wealth”);
                  break;
    }
}
```

Error: case label does not reduce to an integer constant

Find out the errors if any…

```
#include <stdio.h>
int main ()
{
   int i = 1;
   switch (i)
   {
             printf (“Hello, how are you”);
      case 1:
                 printf(“\n Practice makes a man perfect”);
                 break;
       case 2:
                 printf(“\n money is the root of all wealth”);
                 break;
    }
}
```

/*will never get executed*/

However, no error

Possible Errors

1. Case label cannot be float or double or string constant

2. Case label cannot be a variable

The Conditional Operator ? :

- **This makes use of an expression that is either true or false. An appropriate value is selected, depending on the outcome of the logical expression.**

expr1 ? expr2 : expr3;

- **Example :**

```
(marks >= 35) ? printf("Passed \n") : printf("Failed \n");

char x = ((a>=65) && (a<=90)) ? a+32 : a ;
```

Continued…

the segment

```
if ( x<0 )
    flag = 0;
else
    flag=1;
```

can be written as

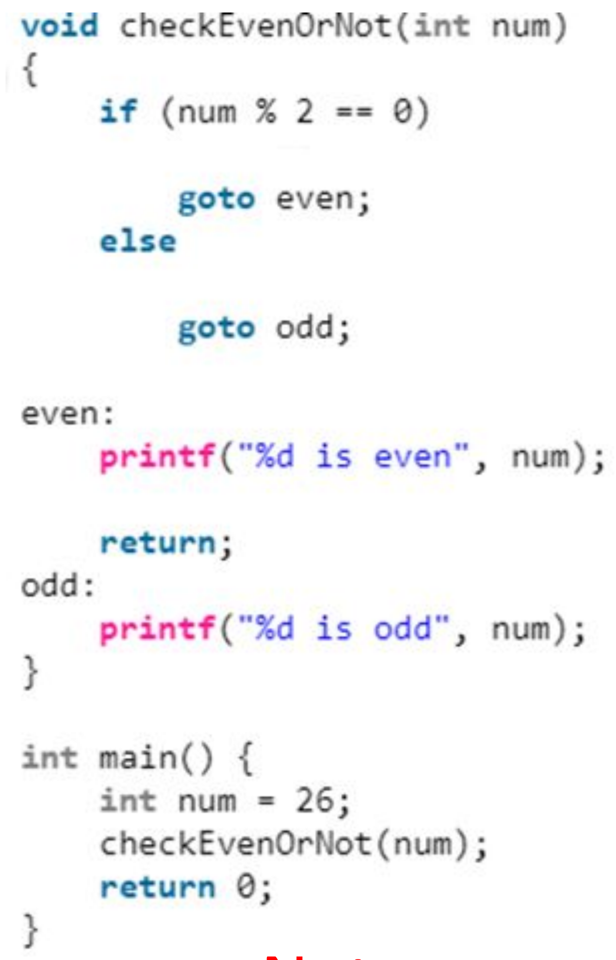
```
flag = ( x<0 ) ? 0 : 1;
```

Output

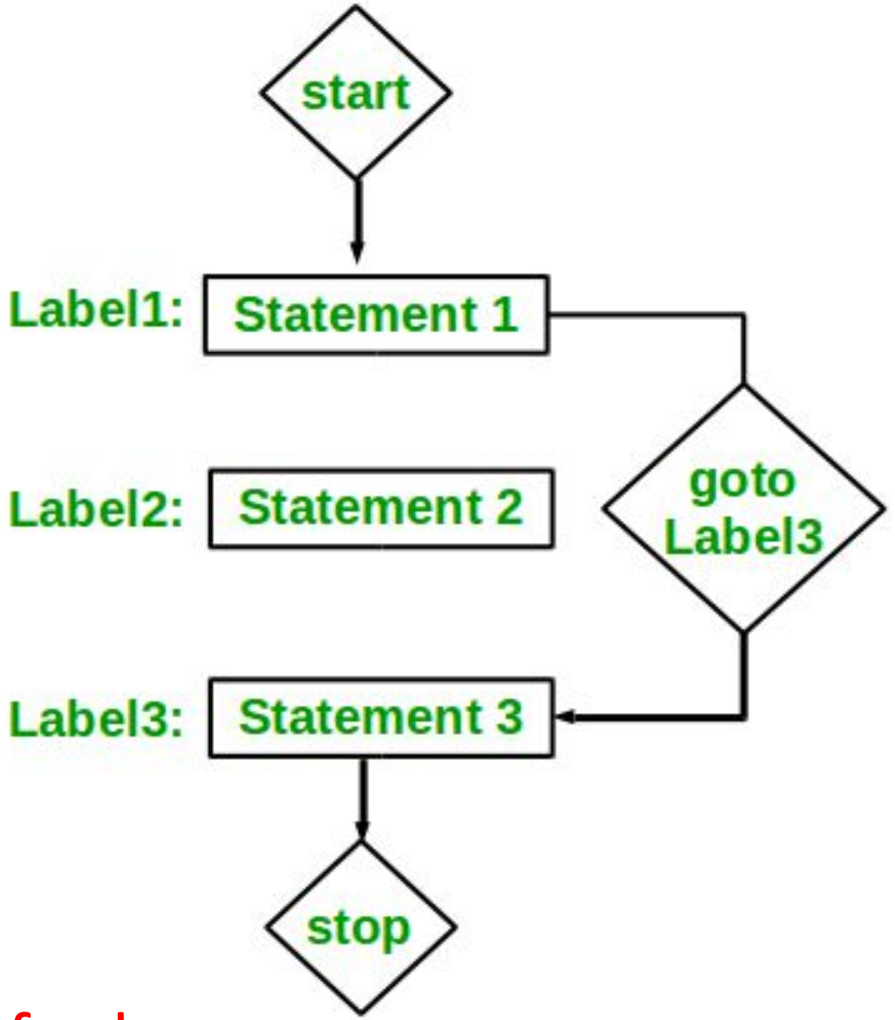
```
#include <stdio.h>
int main()
{
    int a = 10, b;

    printf( "Value of b is %d\n", (a == 1) ? 20: 30 );
    printf( "Value of b is %d\n", (a == 10) ? 20: 30 );

    return 0;
}
```

goto statements in C

- unconditional jump stat



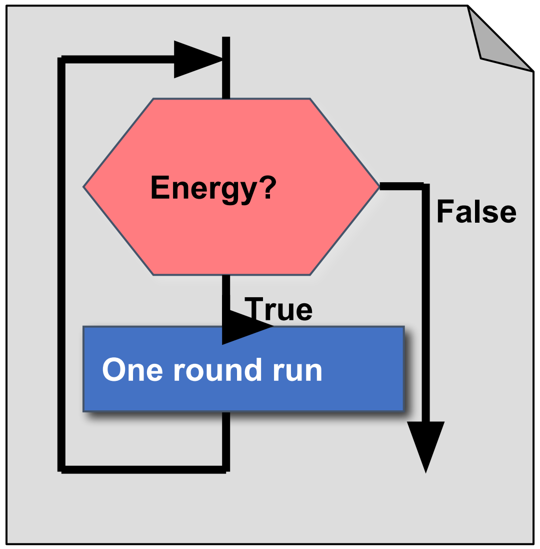
```
void checkEvenOrNot(int num)
{
    if (num % 2 == 0)

        goto even;
    else

        goto odd;

even:
    printf("%d is even", num);

    return;
odd:
    printf("%d is odd", num);
}

int main() {
    int num = 26;
    checkEvenOrNot(num);
    return 0;
}
```

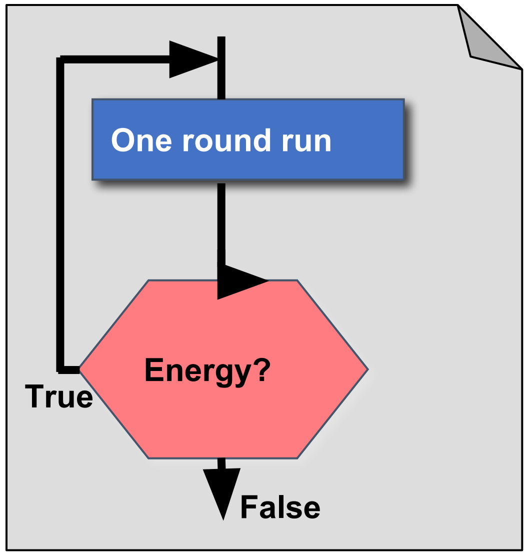


Not recommended for heavy usage.

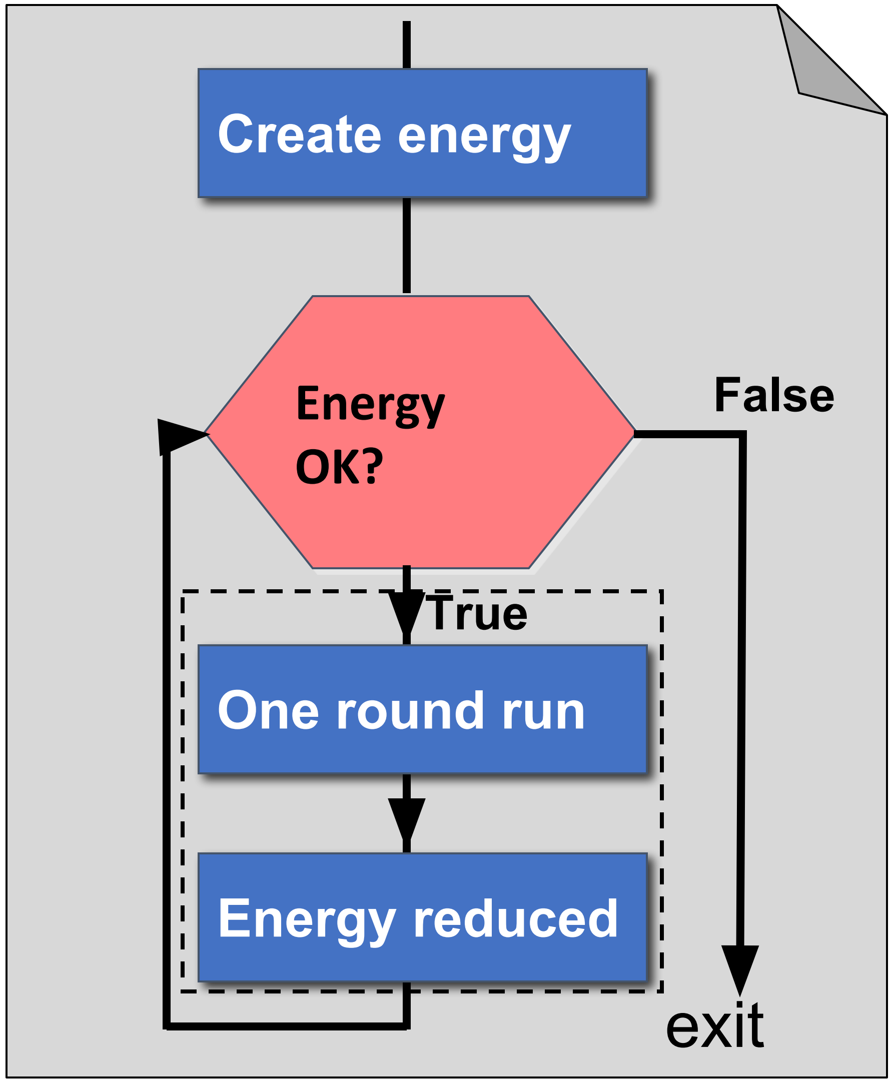
Loops

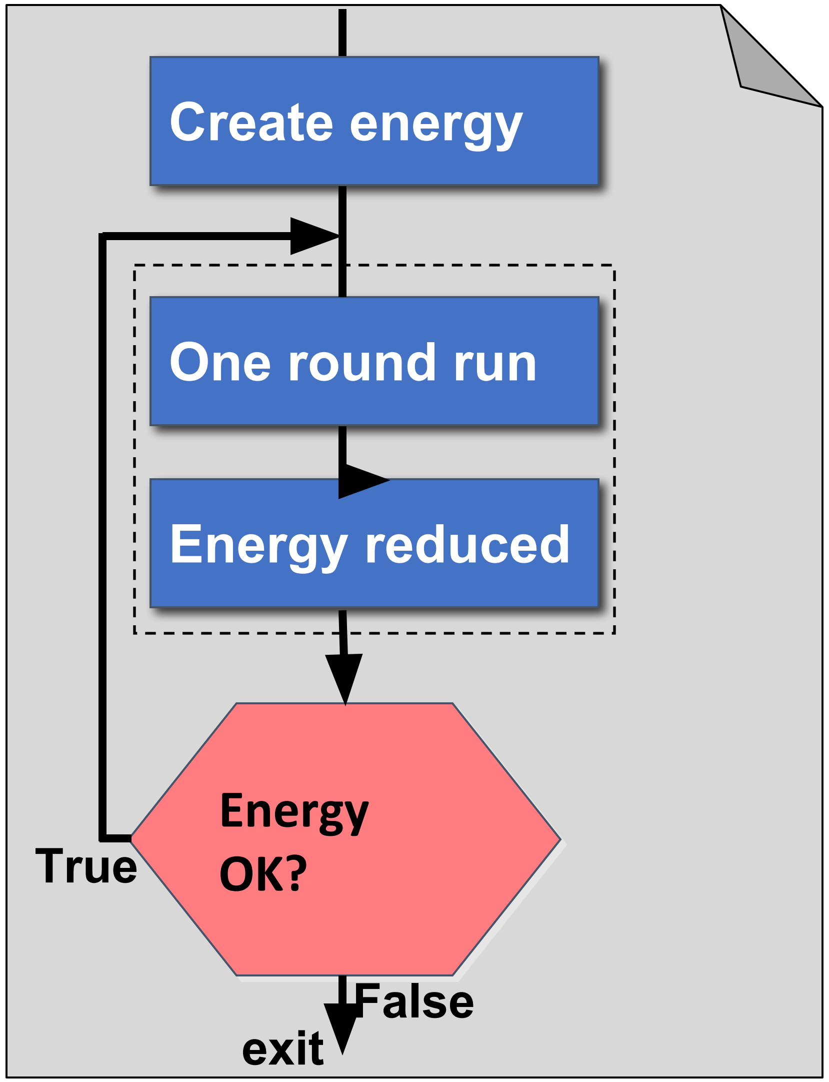
- Ability of a computer to repeat an operation or a series of operations many times.





Example continued…

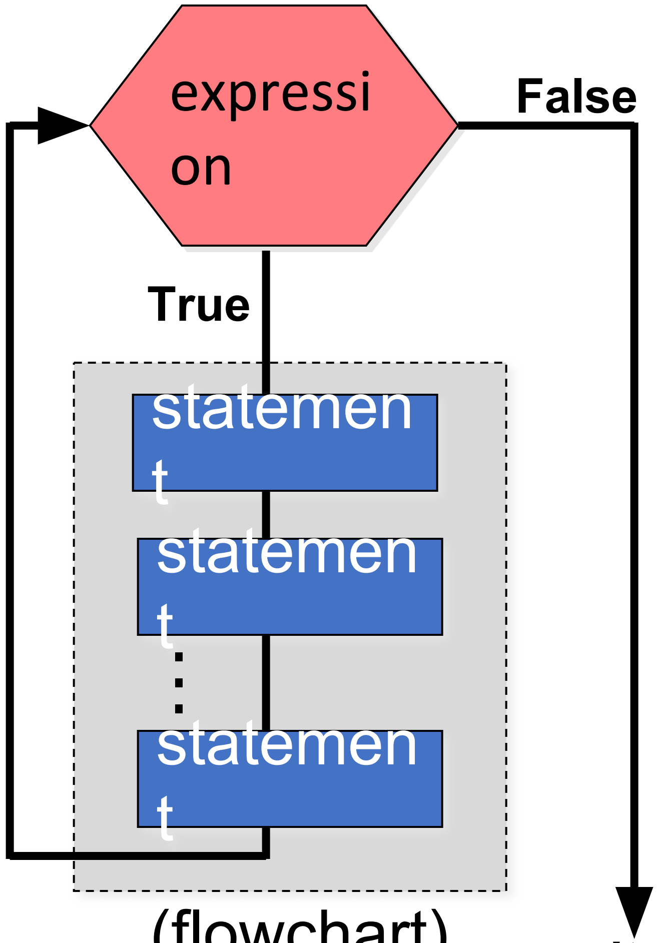




The while Loop

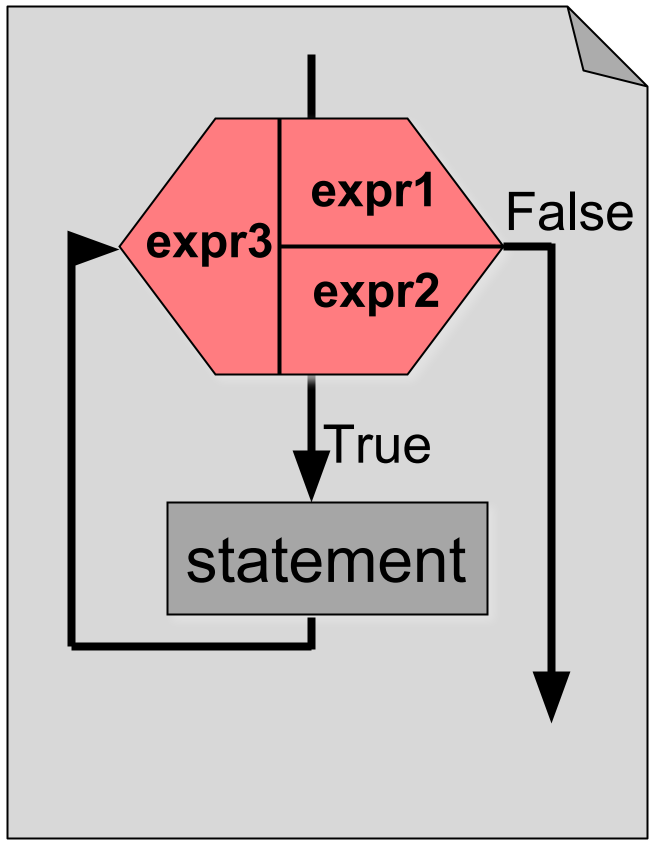
```
while (expression)
 {
     statement;
     statement;
      …
     statement;
  }
```

(code)

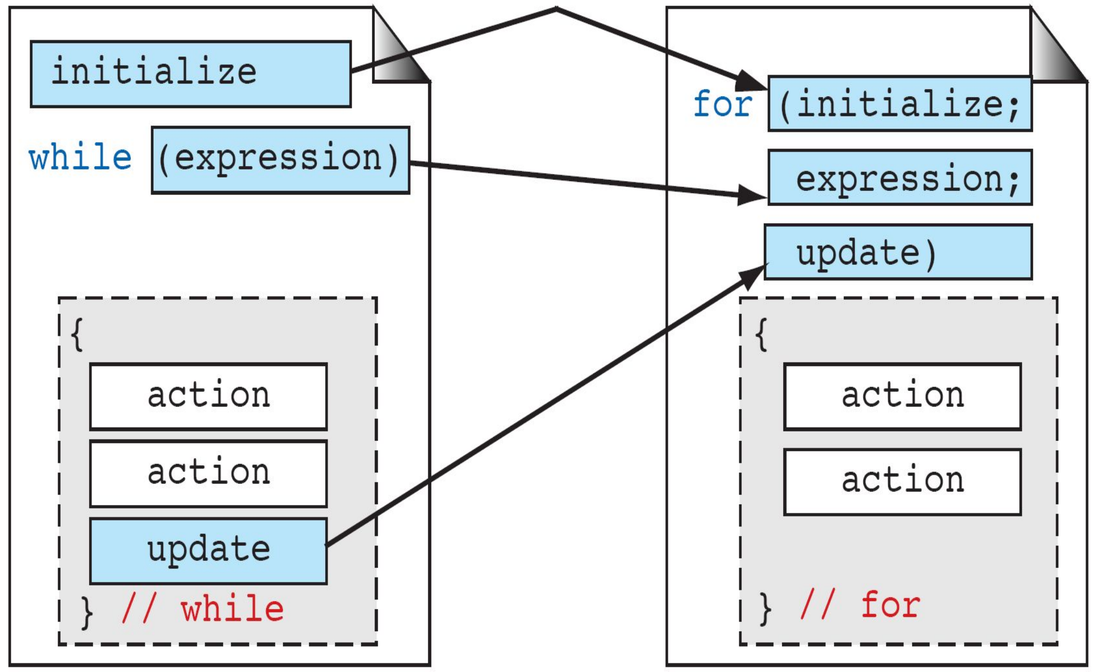


(flowchart)

The for loop



(Flowchart)


initialize
while (expression)
{
action
action
update
} // while
for (initialize;
expression;
update)
{
action
action
} // for