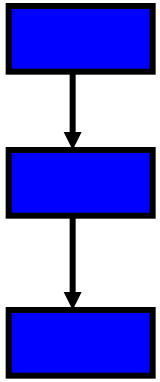
Branching

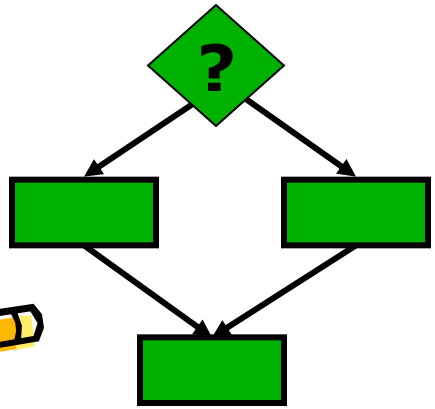
Control Flow

- "Control flow" is the order in which statements are executed

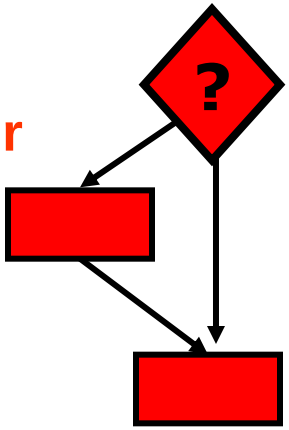


Sequential control flow – the next statement executed is the next one that appears, in order, in the C program

Conditional control flow – choosing which of two (or more) statements to execute before continuing



OR - choosing whether or not to skip a statement before continuing



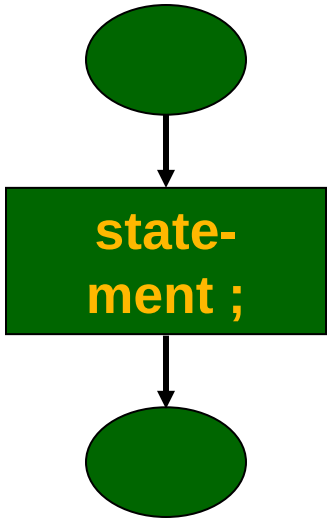
Broadly Classification

- Branching control
- Looping control

Expression statements

Expression statements are expressions followed by a semicolon;

```
Statement ;
```



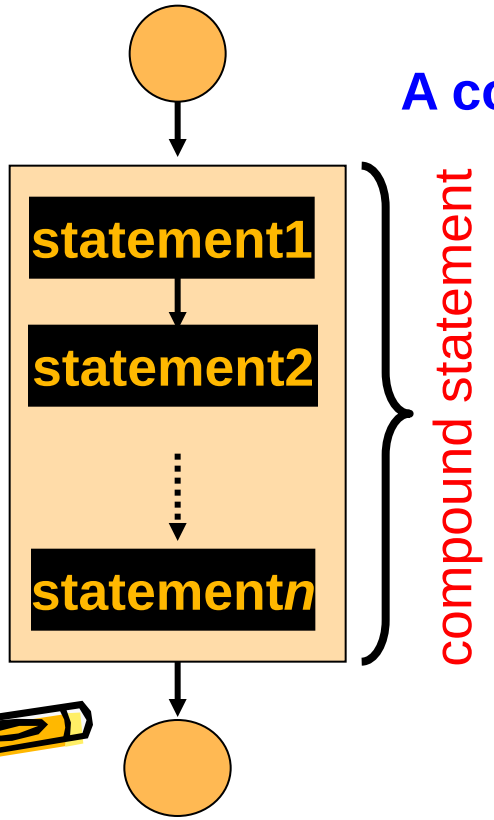
these have to be evaluated (including side effects) before proceeding to the next step.

Examples

```
a = b; /*  assignment statement  */

a + b + c;/*  legal, but not useful    */

; /*  empty statement; sometimes necessary in
  "if" constructs */
printf("%d\n",a);/*  a function call  */
```

Compound statements

A group of statements in braces { } is a compound statement;



A compound statement is itself a statement.

```
{ statement1
  statement2
  /* ... */
  statementn }
```

Examples

```
1.{printf("Hello");
    k++;
    k *= 5;
  }
2.  void main()
    {
        . . .
        . . .
    }
```

NOTE: There is no semi-colon at the end of a block.

Conditional Execution

- A **conditional statement** allows the computer to **choose** an execution path depending on the value of a variable or expression

if the withdrawal is more than the bank balance, **then** print an error

if today is my birthday, **then** add one to my age

```
if (condition)
        statement;
```

- In parentheses is a condition, also called a "**Logical**" or "**Boolean**" expression
- Made up of variables, constants, arithmetic expressions, and the relational operators

Value of condition is:

TRUE any **non-zero** value

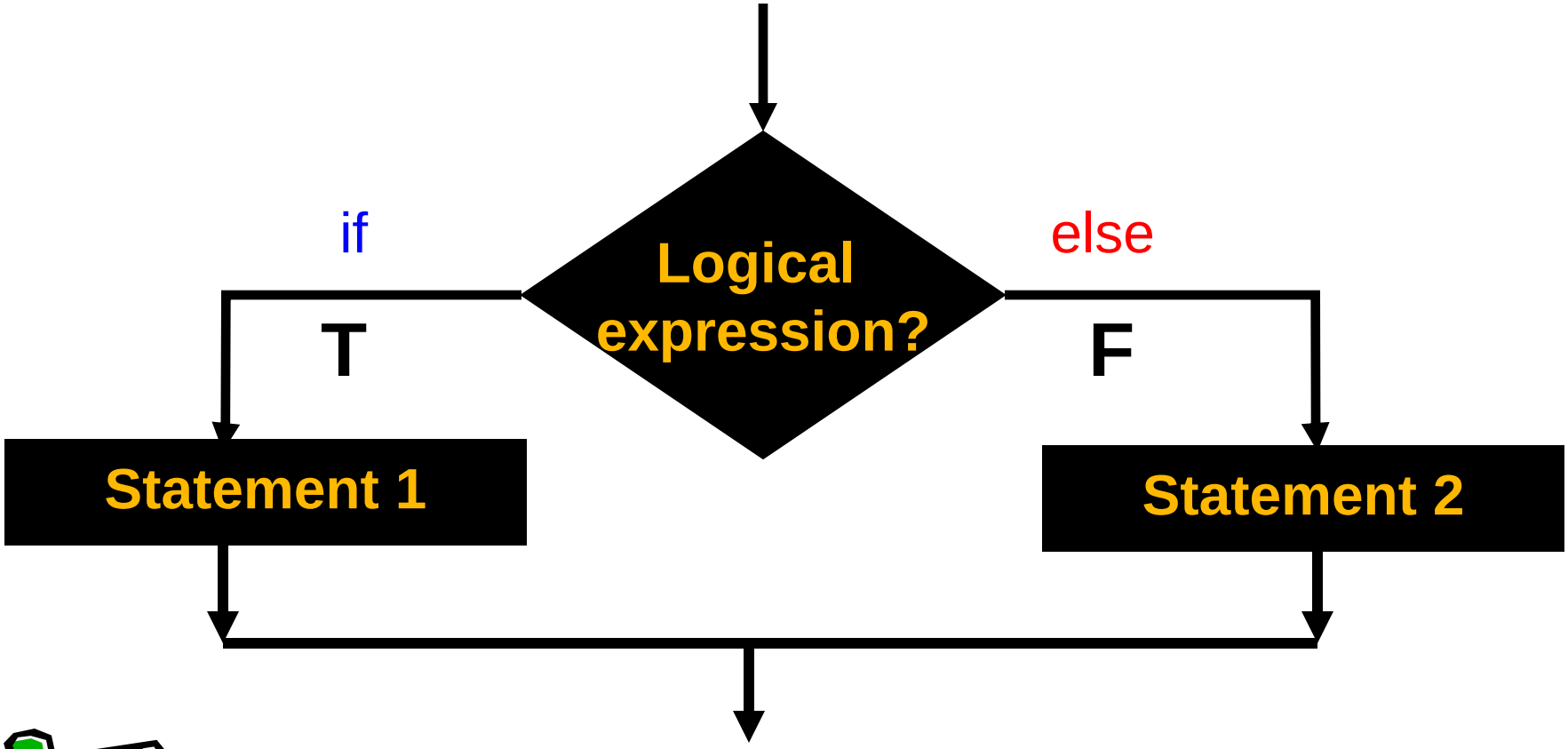
FALSE is **0**

Basic Syntax

```
if (logexpr)
    statement1;
 else
    statement2;
```

Could be either a simple statement or a block

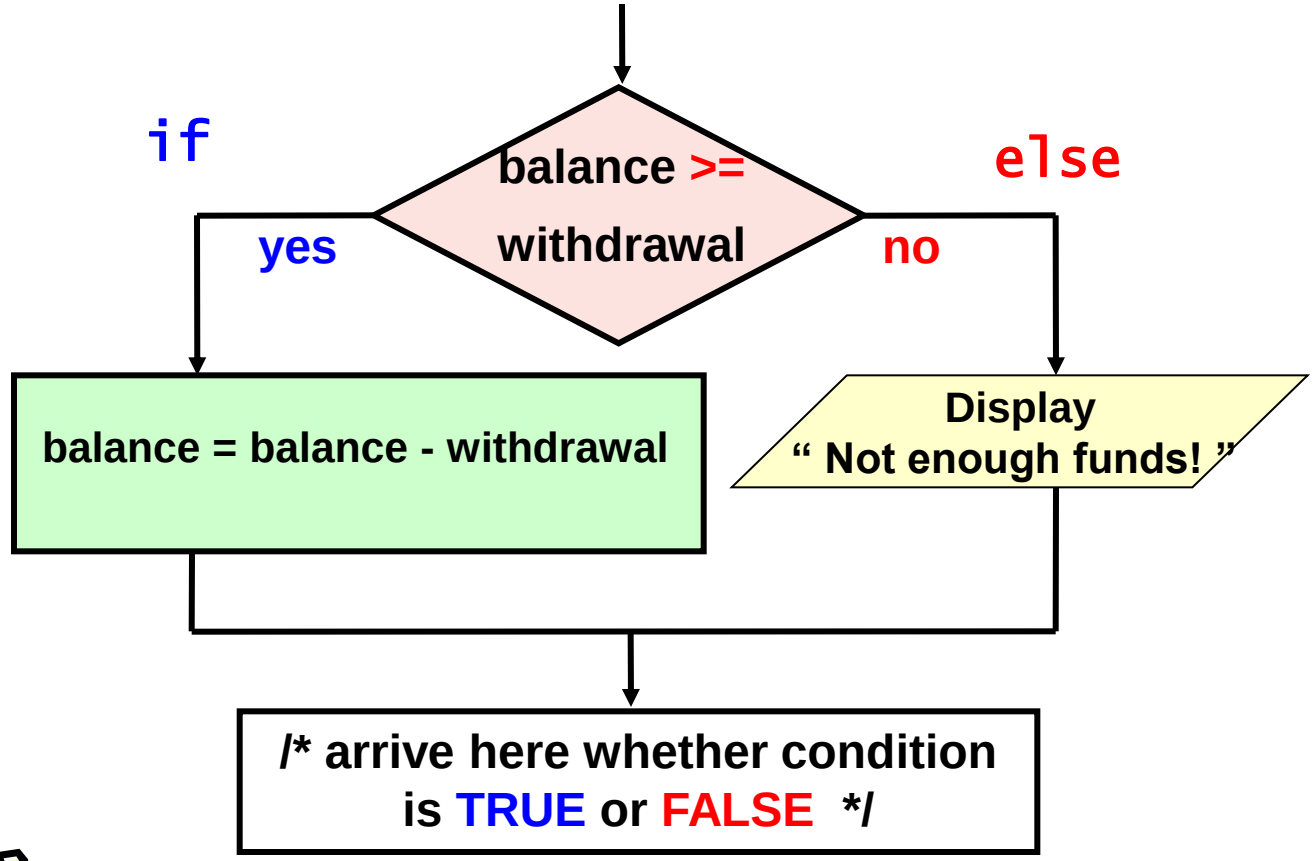
Flow Chart



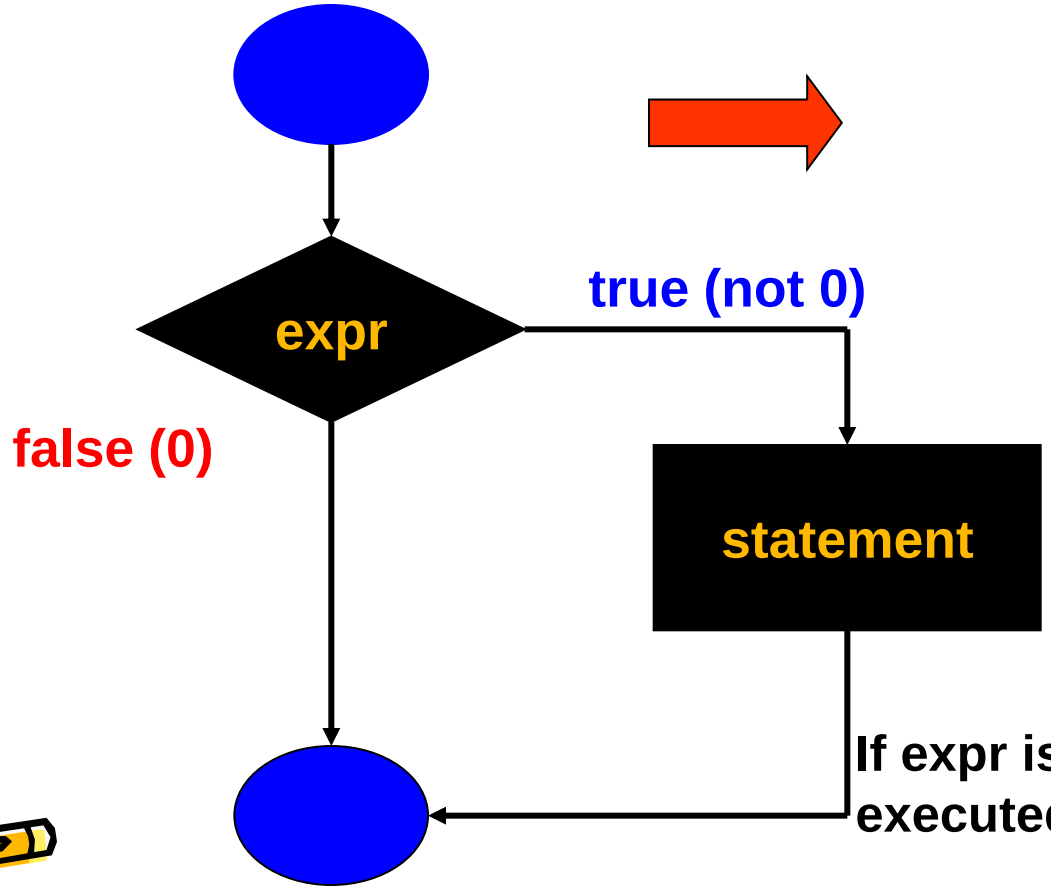
if..else (Example)

```
if ( balance >= withdrawal )
        {
                balance = balance - withdrawal;
        }
else
        {
                printf ( “Not enough funds!\n ” ) ;
        }
```

if..else (Example)



Optional (else)




```
if(expr)
 statement ;
```

If expr is nonzero, statement is executed, otherwise it is skipped.

Nested if-else Statements

- The **if** or **else** clause of an if-else statement itself may be an if-else statement. These are known as **nested if-else** statements

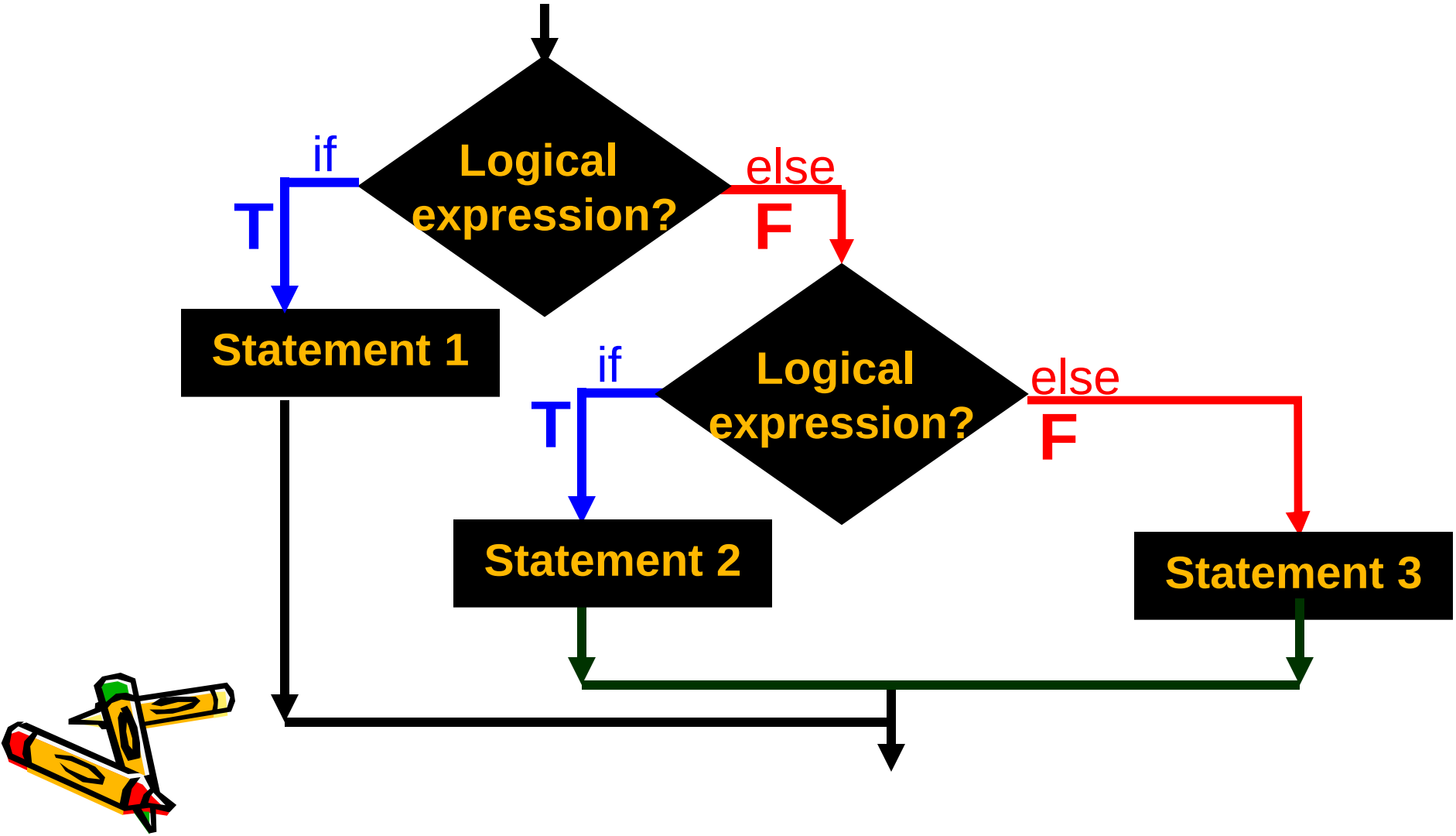
BASIC SYNTAX

```
if (logexpr1)
            statement1;
else if (logexpr2)
            statement2;
else if (logexpr3)
            statement3;
else if (logexpr4)
            statement4;
else
      default_statement;
```

GOOD PROGRAMMING PRATICE

Ladder Structure

Flow Chart


Logical expression?
if
T
else
F
Statement 1
Logical expression?
if
T
else
F
Statement 2
Statement 3

Association else

```
if (logexpr1)
    if(logexpr2)
        statement1;
    else
        statement3;
```

whether the else is associated with the first if or the second if?

The rule here is that an else is associated with the closest previous unmatched if.

Association else

```
if (logexpr1)
    {
    if(logexpr2)
        statement1;
    else
        statement3;
    }
```

Association else

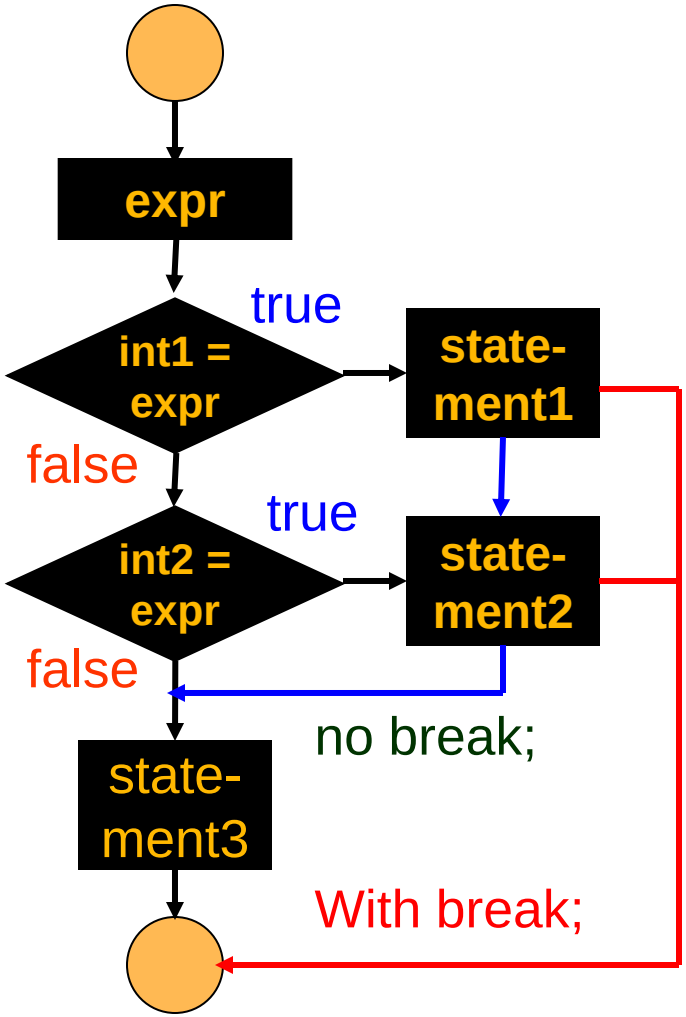
```
if (logexpr1)
    {
    if(logexpr2)
        statement1;
    }
else
    statement3;
```

switch - statement

(Generalization of if – else)

```
switch(expr){
 case int1: statement1; break;
 case int2: statement2; break;
 default: statement3;
}
```

integral expression



without the break statement, execution falls through to subsequent case!

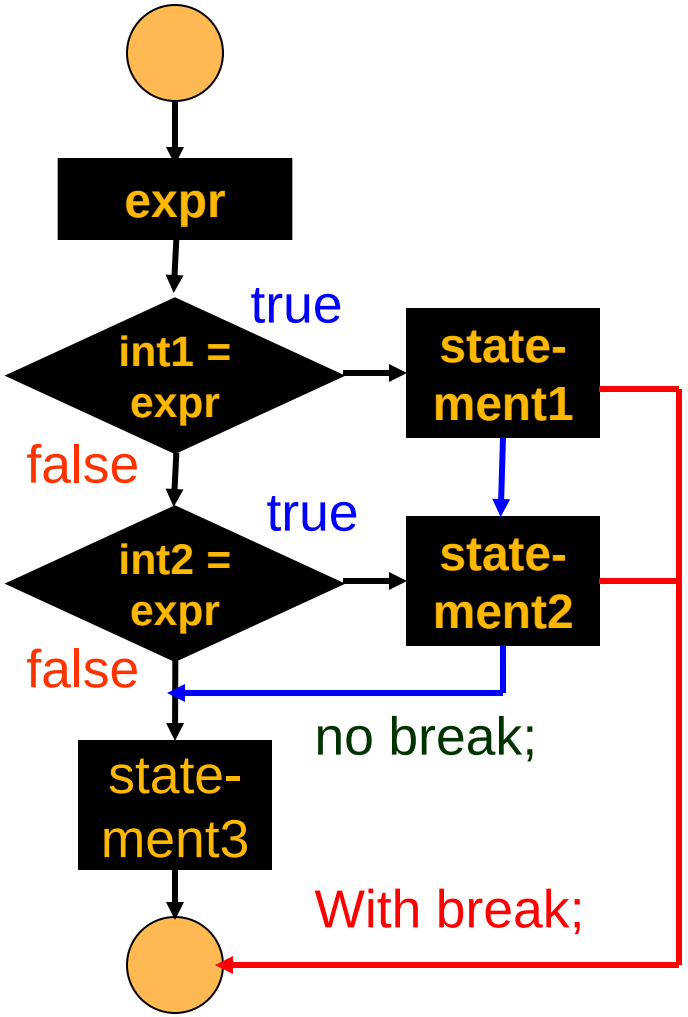
switch - statement

(Generalization of if – else)

```
switch(expr){
 case int1: statement1; break;
 case int2: statement2; break;
 default: statement3;
}
```

integral expression

without the break statement, execution falls through to subsequent case!



FALLTHRU.C

```
#include <stdio.h>
int main()
{
    int j;
    printf("\nEnter an integer value : ");
    scanf("%d",&j);
    switch(j)
        {
        case 0: printf("\ncase 0");
        default: printf("\ndefault case");
        case 1: printf("\ncase 1");
        case 2: printf("\ncase 2");
        }
    return 0;
}
```

OUTPUT:

```
Enter an integer value : 0
case 0
default case
case 1
case 2
```

Most Common Form

```
switch(e)
  {
  case c1 : statement1;break;
  case c2 : statement2;break;
  case c3 : statement3;break;
  default : default statement;
  }
```

Program Indentation

- Proper use of indentation in the text of a program improves the readability of a program and facilitates the finding of bugs in the program.
- It also clarifies which statements go with which control flow statements

Program Indentation Style

- Whitesmith Style
- Indentation style used in K & R
- Allman style (named after Eric Allman)
- GNU style (Free Software Foundation's GNU writing style)

Whitesmith Style

```
if( x < 0.0)
    {
    printf("\n Whitesmith Style");
    exit(1);
    }
```

Indentation style used in K & R

```
if( x < 0.0){
    printf("\n K & R Style.");
    exit(1);
 }
```

Allman style

```
if( x < 0.0)
  {
     printf("\n Allman Style.");
     exit(1);
  }
```

GNU style

```
if( x < 0.0)
    {
        printf("\n GNU Style.");
        exit(1);
    }
```

goto Statement

The goto statement is used to "jump" to another part of the program marked with a label. It has the syntax

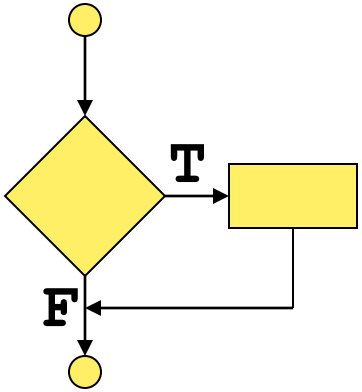
```
goto label;

. . .

. . .

label: . . .
```

any valid identifier

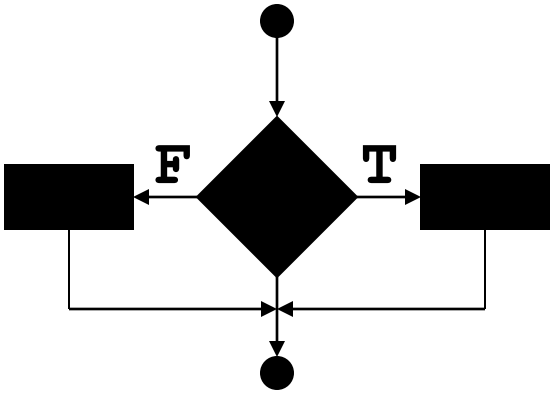
Use of goto statement should be avoided

Selection Structures

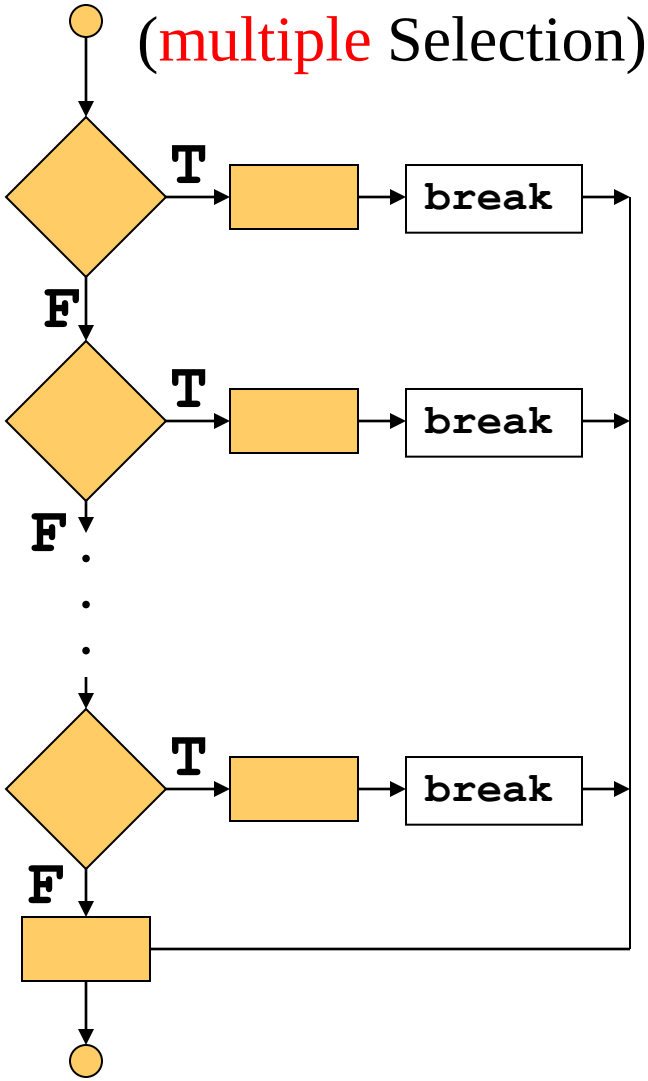
`if` statement
(single Selection)



`if` statement
(double Selection)



`switch` statement
(multiple Selection)



Points to Remember

- The if-else provides the simplest mechanism for branching.
- We must be careful to avoid unintended fall-through in a switch statement.
- As far as possible, we must avoid the use of the goto keyword as it leads to non-readable code.

THANK YOU