EEE146

SELECTION CONTROL STRUCTURE: IF AND SWITCH STATEMENTS

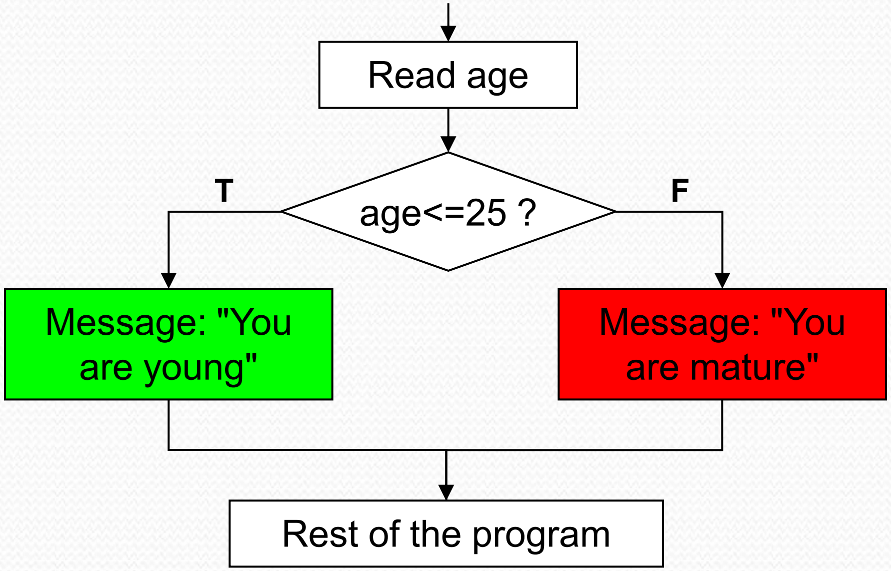
SELECTION CONSTRUCTS

- The C++ selection constructs are used to execute one of the possible alternatives, based on the outcome of a conditional expression.
- The following are some structured statements used for selection.

- if
- if/else
- nested if
- switch/case

If statement

- The "if statement" is used to break the sequential flow of execution.
- Enforces branching in execution according to the result of an expression.
  - There are two possible paths to take.
  - The expression determines which path should be taken.



Read age

age<=25 ?

T

F

Message: "You are young"

Message: "You are mature"

Rest of the program

If statement

- Syntax:

```
if (int_expr)
    stat_block$_1$
else
    stat_block$_2$
```

Text in green is optional

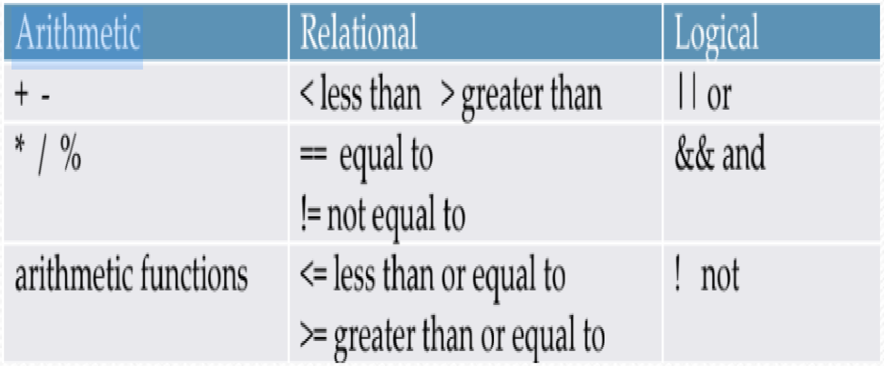
where **stat_block** is one of the following:

- a single statement `stat;`
- the null statement `;`
- a group of statements enclosed in braces

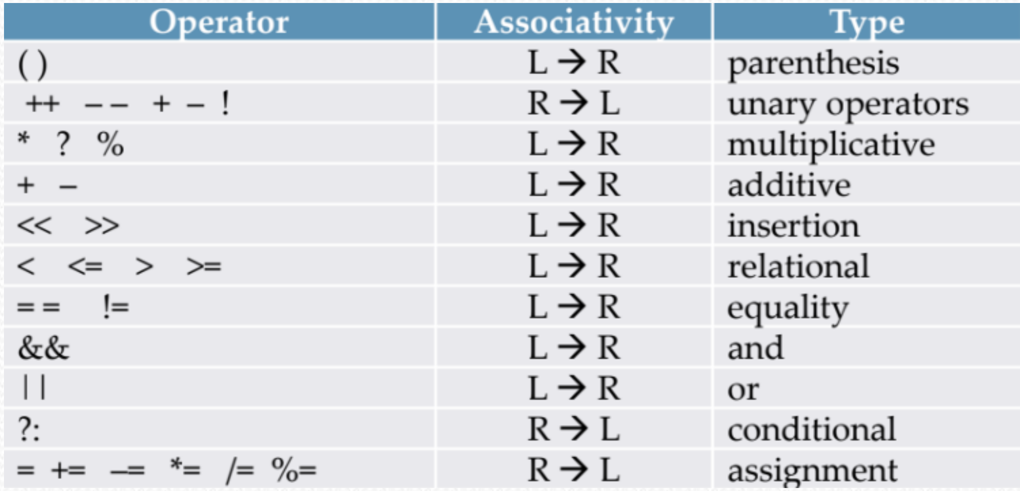
```
{   stat$_1$;
    ...
    stat$_n$;
}
```

THE IF STATEMENT

- A int_expression or boolean expression represents a condition that is either true or false. It is formed using operands (constants, variables) and operators (arithmetic operators, relational operators, logical operators).
- **Remember Operators and precedence:**



Arithmetic	Relational	Logical
+ -	< less than > greater than	| | or
* / %	== equal to != not equal to	&& and
arithmetic functions	<= less than or equal to >= greater than or equal to	! not



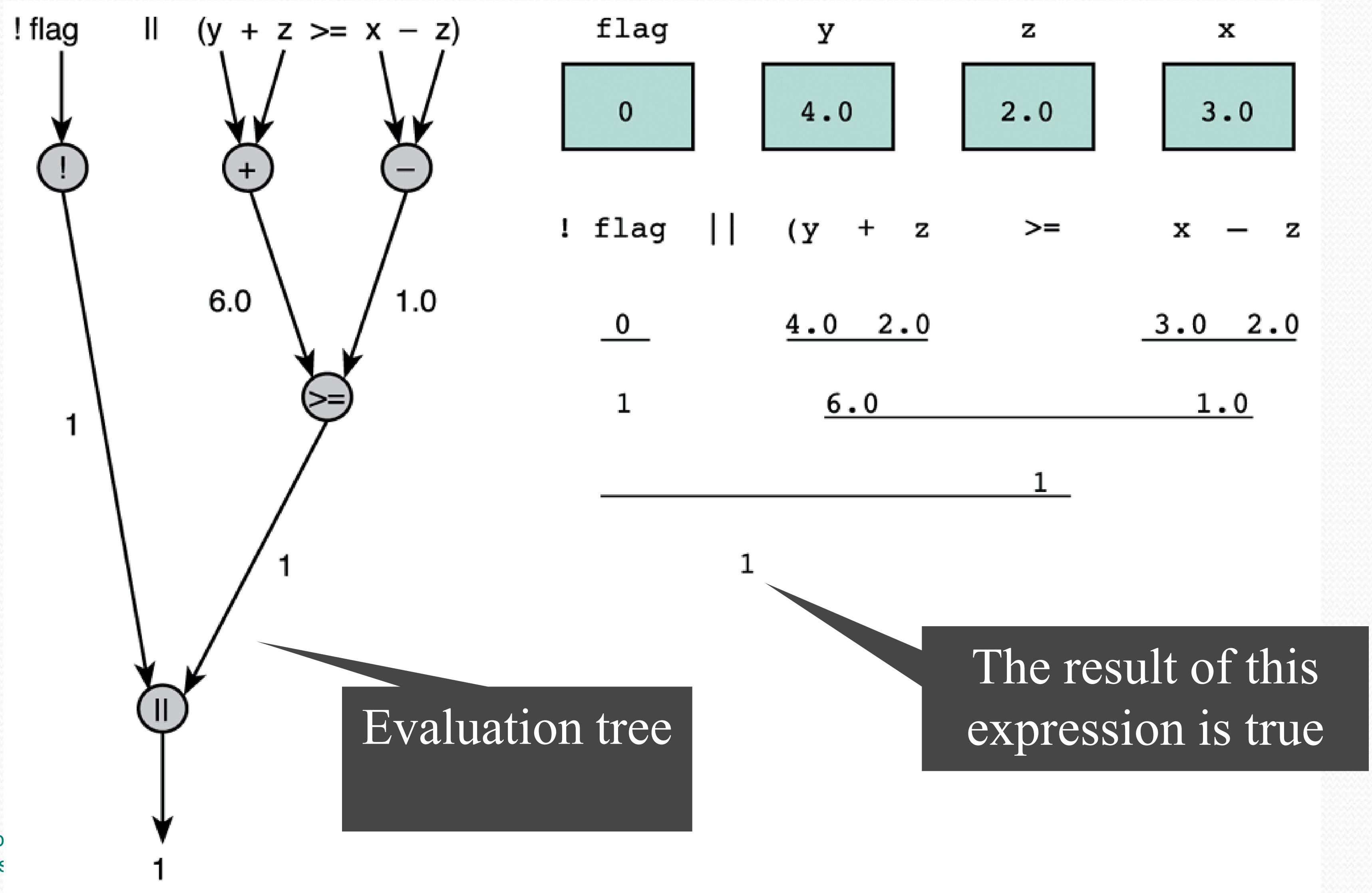
Operator	Associativity	Type
( )	L → R	parenthesis
++ –– + – !	R → L	unary operators
* ? %	L → R	multiplicative
+ –	L → R	additive
<< >>	L → R	insertion
< <= > >=	L → R	relational
= = !=	L → R	equality
&&	L → R	and
| |	L → R	or
?:	R → L	conditional
= += –= *= /= %=	R → L	assignment

For example,
if ((x > 0) && (a == b)) …

Example

- x: 3.0, y: 4.0, z: 2.0, flag: 0
- Expressions:
  - ! flag
  - x + y / z <= 3.5
  - ! flag || (y + z >= x – z)
  - ! (flag || (y + z >= x – z))

Evaluation for `!flag || (y + z  >= x - z)`



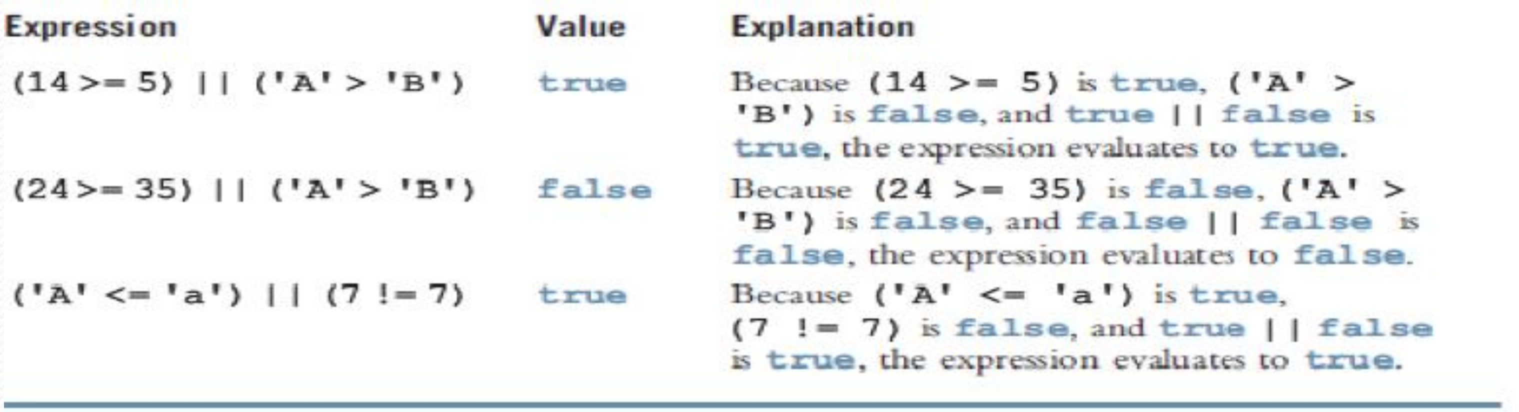
Evaluation tree

The result of this expression is true

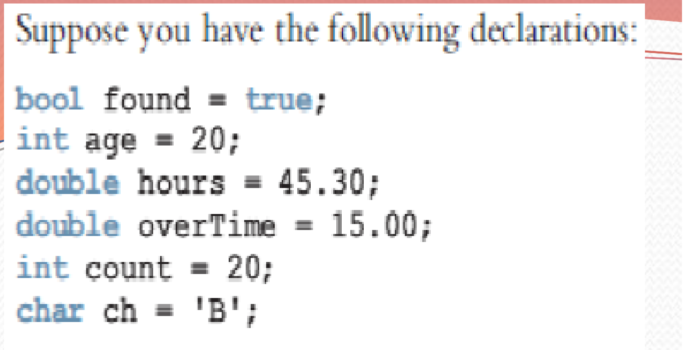
Comparing Characters

- We can also compare characters in C++ using the **relational** and **equality operators**.

Expression	Value
‘9’ >= ‘0’	1 (true)
‘a’ < ‘e’	1 (true)
‘Z’ == ‘z’	0 (false)
‘a’ <= ‘A’	0 (false)

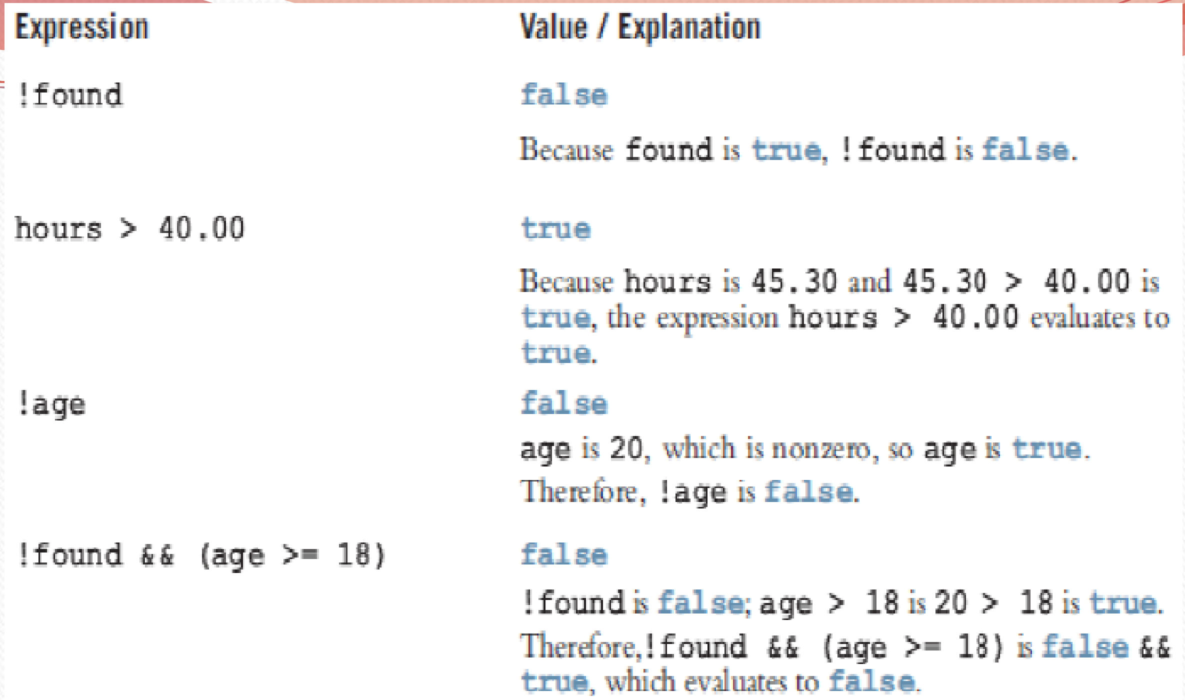


Expression	Value	Explanation
(14 >= 5) || ('A' > 'B')	true	Because (14 >= 5) is true, ('A' > 'B') is false, and true || false is true, the expression evaluates to true.
(24 >= 35) || ('A' > 'B')	false	Because (24 >= 35) is false, ('A' > 'B') is false, and false || false is false, the expression evaluates to false.
('A' <= 'a') || (7 != 7)	true	Because ('A' <= 'a') is true, (7 != 7) is false, and true || false is true, the expression evaluates to true.

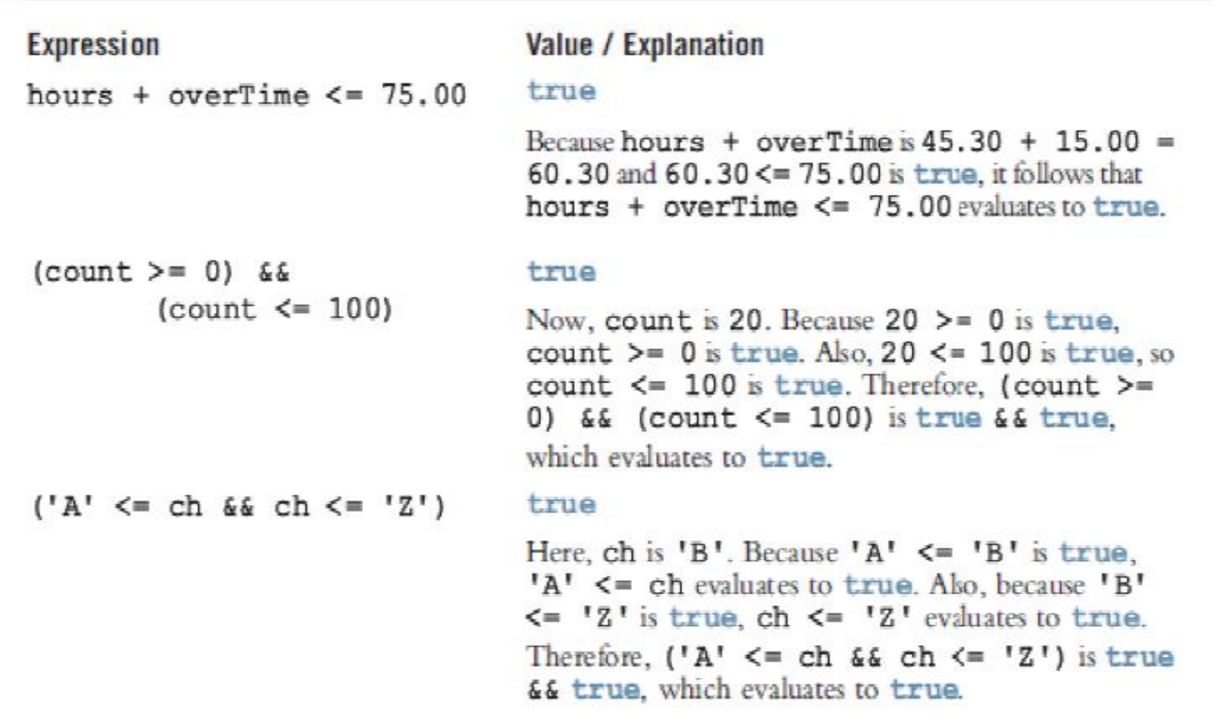


Suppose you have the following declarations:

```
bool found = true;
int age = 20;
double hours = 45.30;
double overTime = 15.00;
int count = 20;
char ch = 'B';
```



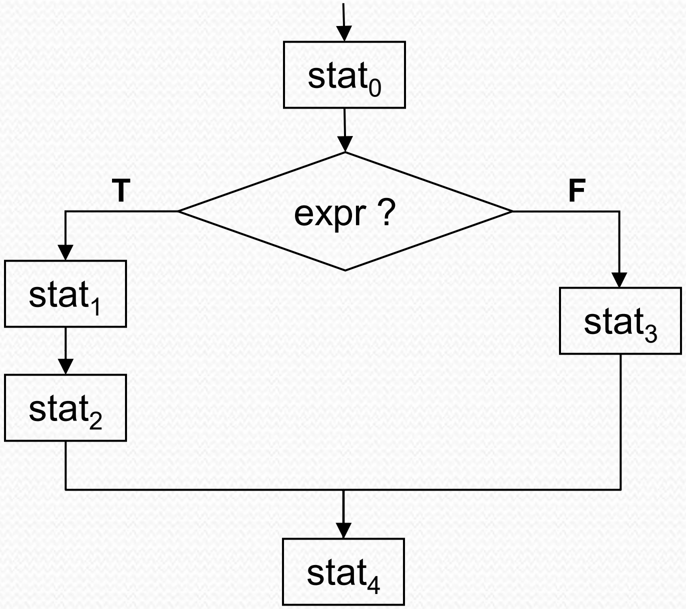
Expression	Value / Explanation
`!found`	false Because `found` is true, `!found` is false.
`hours > 40.00`	true Because `hours` is 45.30 and 45.30 > 40.00 is true, the expression `hours > 40.00` evaluates to true.
`!age`	false `age` is 20, which is nonzero, so `age` is true. Therefore, `!age` is false.
`!found && (age >= 18)`	false `!found` is false; `age > 18` is 20 > 18 is true. Therefore,`!found && (age >= 18)` is false && true, which evaluates to false.



Expression	Value / Explanation
`hours + overTime <= 75.00`	true Because `hours + overTime` is 45.30 + 15.00 = 60.30 and 60.30 <= 75.00 is true, it follows that `hours + overTime <= 75.00` evaluates to true.
`(count >= 0) && (count <= 100)`	true Now, `count` is 20. Because 20 >= 0 is true, `count >= 0` is true. Also, 20 <= 100 is true, so `count <= 100` is true. Therefore, `(count >= 0) && (count <= 100)` is true && true, which evaluates to true.
`('A' <= ch && ch <= 'Z')`	true Here, `ch` is 'B'. Because 'A' <= 'B' is true, `'A' <= ch` evaluates to true. Also, because 'B' <= 'Z' is true, `ch <= 'Z'` evaluates to true. Therefore, `('A' <= ch && ch <= 'Z')` is true && true, which evaluates to true.

If statement

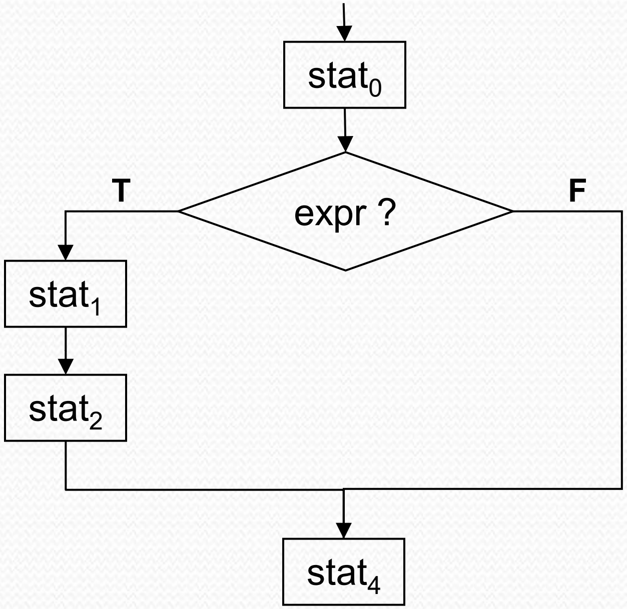
```
stat₀;
if (expr)
{   stat₁;
    stat₂;
}
else
    stat₃;
stat₄;
```



Notice the indentation.

If statement

```
stat0;
if (expr)
{   stat1;
    stat2;
}
stat4;
```



How would you move $stat_1$ and $stat_2$ to the "then" branch?

If statement

- Read one character as input; check if it is a digit; if so, convert it to an integer and display twice that number; o/w display an error message.

```
char ch;   int num;
cin>> ch;
if (('0'<=ch) && (ch<='9'))
{
   num=ch-'0';
   cout<<"Twice input is "<< 2*num<<endl;
}
else
    cout<<"Input is not a digit! \n";
```

If statement

- Read a number and state whether it is odd or even.

```
int num;
 cin>> num;
 cout<< num <<"is an ";

 if (num%2!=0)
    cout<<"odd ";
 else
     cout<<"even ";
 cout<<"number.\n ";
```

Nested-if statements

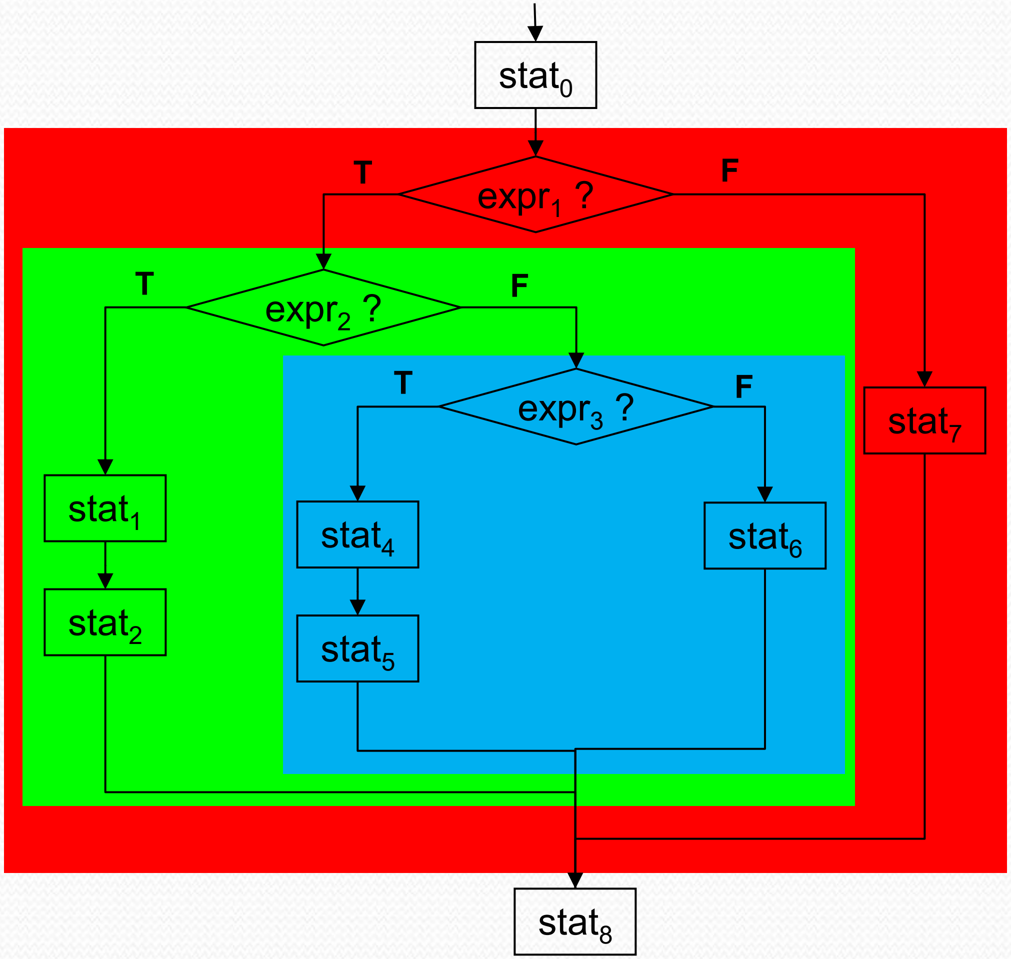
- Remember the syntax:

```
if (int_expr)
    stat_block$_1$
else
    stat_block$_2$
```

- Statement block contains statements.
  - "if" is also a statement.
  - So, you can "nest" one if statement in another.
    - This structure is called nested-if statements.
    - You can nest as much as you can.

Nested-if statements

```
stat0;
if (expr1)
    if (expr2)
    {   stat1;
        stat2;
    }
    else
        if (expr3)
        {   stat4;
            stat5;
        }
        else
            stat6;
else
    stat7;
stat8;
```

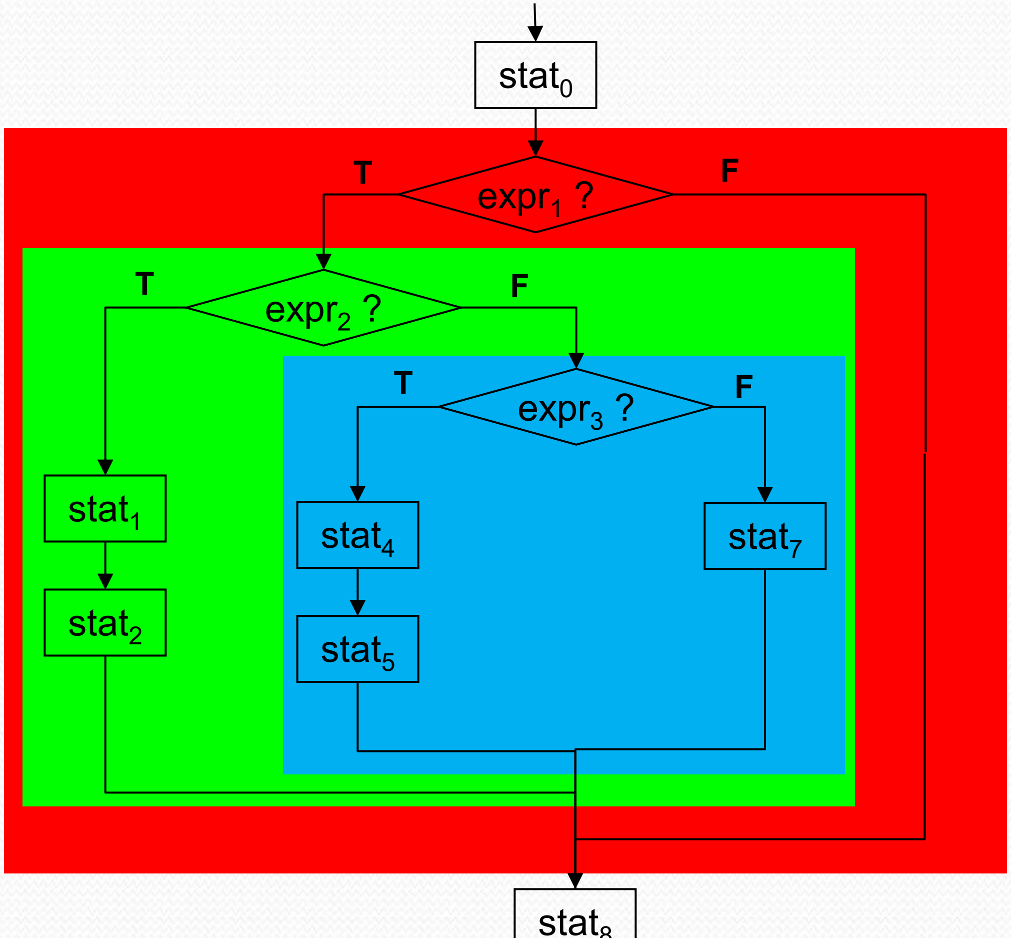


Nested-if statements

- Remember that the "else" part is optional.
  - Thus, some of the if statements may not have an "else" part.
  - Then, it is a little bit tricky to find to which "if" the "else" belongs.
    - Note that indentation is completely ignored by the compiler, so it does not help.
- The trick is the following:
  - "else" belongs to the <u>nearest incomplete</u> "if"

Nested-if statements

```
stat0;
if (expr1)
    if (expr2)
    {   stat1;
        stat2;
    }
    else
        if (expr3)
        {   stat4;
            stat5;
        }
        ~~else~~
            ~~stat6;~~
else
    stat7;
stat8;
```



IF INDENTATION IS NOT CORRECT, IT IS MISLEADING

Example:

```
 m= -1;
if (a>20)
 if (b<10)
if (a>=30) m =4;
else m=0;
else m=1;
else m=2;
cout<<m;
```

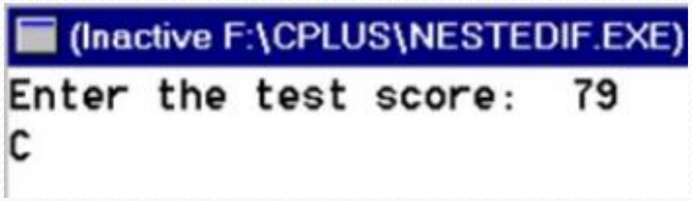
Else-if statements

- Else-if is a variation of nested-if.
- An inner if statement is executed iff all previous if statements have failed.
  - Thus, executing an inner if statement *implies* that all previous expressions were false.
- Syntax:

```
if (int_expr$_1$)
   stat_block$_1$
else if (int_expr$_2$)
   stat_block$_2$
...
else
   stat_block$_n$
```

Else-if statements

```
// This program converts a test score into a letter grade.
#include <iostream>
int main(){
int score;
cout << "Enter the test score: "; cin >> score;
if (score > 100) cout << "Error: score is out of range." ;
else if (score >= 90) cout << 'A';
        else if (score >= 80) cout << 'B';
                else if (score >= 70) cout << 'C';
                        else if (score >= 60) cout << 'D';
                                else if (score >= 0) cout << 'F';
                                        else
                                        cout <<"Error: score is out of range.";
return 0;
}
```



```
Enter the test score:  79
C
```

example

```
if (a < b)
if (a < c)
result = a;
else result = c
else if (b < c)
result = b;
else
result = c;
Cout<< " The smallest is << result;
```

Else-if statements

```
if (age<=1)
     cout<<"infant";
   else if (age<=3)
     cout<<"toddler";
   else if (age<=10)
     cout<<"child";
   else if (age<=18)
     cout<<"adolescent";
   else if (age<=25)
     cout<<"young";
   else if (age<=39)
     cout<<"adult";
   else if (age<=65)
     cout<<"middle-aged";
   else
    cout<<"elderly";
```

```
if (age<=1)
 cout<<"infant";
if (age<=3)
 cout<<"toddler";
if (age<=10)
 cout<<"child";
if (age<=18)
 cout<<"adolescent";
if (age<=25)
 cout<<"young";
if (age<=39)
 cout<<"adult";
if (age<=65)
cout<<"middle-aged";
```

These two codes gives different outputs

Else-if statements

- Alternative would be:

```
if (age<=1)
 cout<<"infant";
if ((1<age) && (age<=3))
 cout<<"toddler";
if ((3<age) && (age<=10))
 cout<<"child";
if ((10<age) && (age<=18))
 cout<<"adolescent";
if ((18<age) && (age<=25))
 cout<<"young";
if ((25<age) && (age<=39))
 cout<<"adult";
if ((39<age) && (age<=65))
 cout<<"middle-aged";
if (65<age)
 cout<<"elderly";
```

Example

- Given a person's salary, we want to calculate the tax due by adding the base tax to the product of the percentage times the excess salary over the minimum salary for that range.

Salary Range	Base tax	Percentage of Excess
0.00 – 14,999.99	0.00	15
15,000.00 – 29,999.99	2,250.00	18
30,000.00 – 49,999.99	5,400.00	22
50,000.00 – 79,999.99	11,000,00	27
80,000.00 – 150,000.00	21,600.00	33

```
/*tax=(salary - base_salary)*percantage_of_excess+base_tax*/
```

Common Programming Errors

- Consider the statement:
  if (0 <= x <= 4)
- This is always true!
  - First it does 0 <= x, which is true or false so it evaluates to 1 for true and 0 for false
  - Then it takes that value, 0 or 1, and does 1 <= 4 or 0 <= 4
  - Both are always true
- In order to check a range use (0 <= x && x <= 4).

- Consider the statement:
  if (x = 10)
- This is always true!
  - The = symbol assigns x the value of 10, so the conditional statement evaluates to 10
  - Since 10 is nonzero this is true.
  - You must use == for comparison

Ternary (conditional) operator

- Ternary operator is similar to the "if" statement. But it is an operator, not a statement.
- Syntax:

  int_expr ? value_1 : value_2

- Eg:

```
a = (b>c) ? b : c;
k = (n!=0) ? m/n : 0;
```

Ternary operator

- Note that it is not possible to know at compile time whether `value`$_1$ or `value`$_2$ will be used.
  - Therefore, the type of the expression is the type of the larger value.
- Eg: In the expression below, if the value of `b` is `9` and if a if float variable;

```
    a = b / ((b%2)?2:3.0);
```

  the value of `a` is `4.5` (not `4`), because we perform a float division (not integer division)

Switch statement

- If you have multiple cases depending on different values of the same integer expression, switch is easier to use.
- Syntax:

```
switch (int_expr)
{   case constant_int_value$_1$: stat(s);
    case constant_int_value$_2$: stat(s);
    ...
    default: stat(s);
}
```

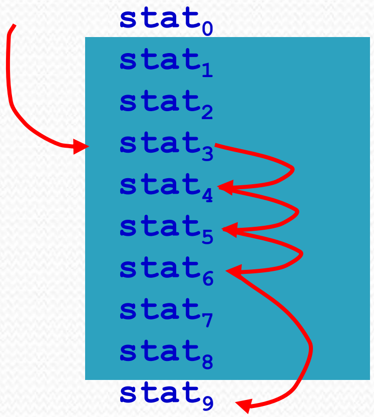
- You may have zero or more statements in each case.

Break statement

- Switch statement actually gathers many statements of several cases.
  - The case labels denote the specific statement from which the execution of this group of statements begins.
  - All statements till the end of the group are executed sequentially.
- To separate the cases, **`break`** statement is used.
  - **`break`** breaks the sequential execution of the statements and immediately jumps to the end of the switch statement.

Break statement

```
stat$_0$;
switch (expr)
{   case value$_1$:
        stat$_1$;
        stat$_2$;
    case value$_2$:
        stat$_3$;
        stat$_4$;
        stat$_5$;
    case value$_3$:
        stat$_6$;
        break;
    case value$_4$:
        stat$_7$;
        stat$_8$;
}
stat$_9$;
```



stat$_0$
stat$_1$
stat$_2$
stat$_3$
stat$_4$
stat$_5$
stat$_6$
stat$_7$
stat$_8$
stat$_9$

If **expr** happens to be **value$_2$**

Switch statement

- Define the days of the week as an enumerated type and display them as strings.

```
enum day_type {MON=1,TUE,WED,THU,FRI,SAT,SUN} day;

cin>>day;
switch (day)
{   case SUN: cout<<"Sunday\n"; break;
    case WED: cout<< "Wednesday\n"; break;
    case TUE: cout<< "Tuesday\n"; break;
    case THU: cout<< "Thursday\n"; break;
    case FRI: cout<< "Friday\n"; break;
    case SAT: cout<< "Saturday\n"; break;
    case MON: cout<< "Monday\n"; break;
    default: cout<< "Incorrect day\n"; break;
}
```

Switch statement

- Note that without the "break" statement, execution traverses all cases until the end of the switch statement.
- This allows implementation of OR (if you use it properly.
- Eg:

```
switch (number)
{   case 1:
    case 3:
    case 5: cout<<"Odd number \n"; break;
    case 0:
    case 2:
    case 4: cout<<"Even number \n"; break;
}
```

Example:

```
// This program also converts a test score into a letter grade
#include <iostream>
int main()
{ int score;
cout << "Enter the test score: ";
cin >> score;
switch (score/10)
{
case 10:
case 9: cout << 'A' << endl;
break;
 case 8: cout << 'B' << endl;
 break;
case 7: cout << 'C' << endl;
break;
case 6: cout << 'D' << endl;
break;
case 5:
 case 4:
case 3:
case 2:
case 1:
 case 0:
cout << 'F' << endl;
break;
default: cout << "Error: score is out of range.\n";
} return 0;
}
```

Switch statement

- As long as the cases are separated with "break"s, their order is not relevant.
- "default" is optional. If the default case is not specified and none of the cases holds, no statement is executed; this is not an error.
- It is a good practice to put a break even after the last case.

Example 1

- Write a code segment that detects whether a number is divisible by 6.

```
if ((num%2==0) && (num%3==0))
   cout<<num<<" is divisible by 6 \n";
```

Example 2

- Write a code segment that detects whether a number is divisible by 3 or 6.

```
if (num%3==0)
   if (num%2==0)
      cout<<" is divisible by 6 \n“;
   else
       cout<<" is divisible by 3 \n";
```

Example 3

- Write a program that reads two real numbers and checks if they are equal.

  (Assume first number is smaller.)

```
#include <iostream>
using namespace std;
#define EPSILON 0.000000001
float r1, r2;

int main()
{
    cin>>r1>>r2;
   if ((r2-r1)<=EPSILON)
      cout<<"The numbers are (almost) equivalent";
   return 0;
}
```

Example 4

- Write a program that reads a 3-digit number from the input char-by-char, and displays its square.

```
#include <iostream>
using namespace std;
char c1, c2, c3;
int num;

int main()
{
   cin>>c1>>c2>>c3;
   num  = (c1-'0')*100;
   num += (c2-'0')*10;
   num += c3-'0';
   cout<<num<<endl<<num*num<<endl;
   return 0;
}
```

Example 5

- Write a program that reads a character ('a', 'p', or 'v') and radius R. It displays the area or perimeter of a circle with radius R, or the volume of a sphere.

```
#include <iostream>
using namespace std;
#define PI ~~3.14~~   3.141592654
char ch;   float R;

int main()
{
   cin>>ch>>R;
   switch(ch)
   {
       case 'a': cout<<"Area of circle = "<< PI*R*R <<endl;
                 break;
      case 'p': cout<<"Perimeter of circle = "<< 2*PI*R<< endl;
                break;
      case 'v': cout<<"Volume of sphere = "<< (4/3)*(PI*R*R*R)<<endl;
                break;
      default:  cout<<"Invalid input"<<endl;  (4.0/3)
                break;}
   return 0;
}
```

Homework

Write a program to compute the gross salary for an employee given the number of hours worked and the hourly rate. If the number of hours worked is greater than 40, the hourly rate shall be 1.5 times the normal hourly rate for all overtime hours. The program should print the overtime hours, the regular salary, the overtime salary, and the gross salary for an employee.

Homework

- Write a program to determine whether three lengths a, b, and c form a triangle.
- Conditions: |a-b|<c and   c< a+b
- Modify the program so that it also determines whether the triangle is isosceles or equilateral.