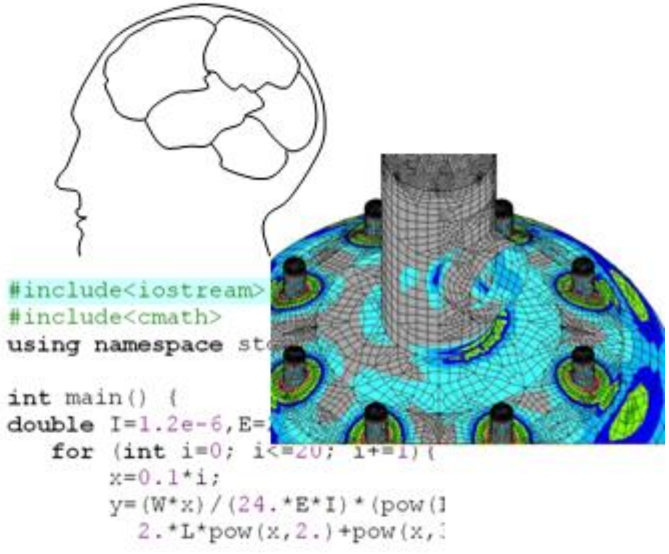
ME 110 Computation for Mechanical Engineering



```
#include<iostream>
#include<cmath>
using namespace st

int main() {
double I=1.2e-6,E=
   for (int i=0; i<=20; i+=1){
        x=0.1*i;
        y=(W*x)/(24.*E*I)*(pow(1
          2.*L*pow(x,2.)+pow(x,
```

Week 5

Control structures: Selection

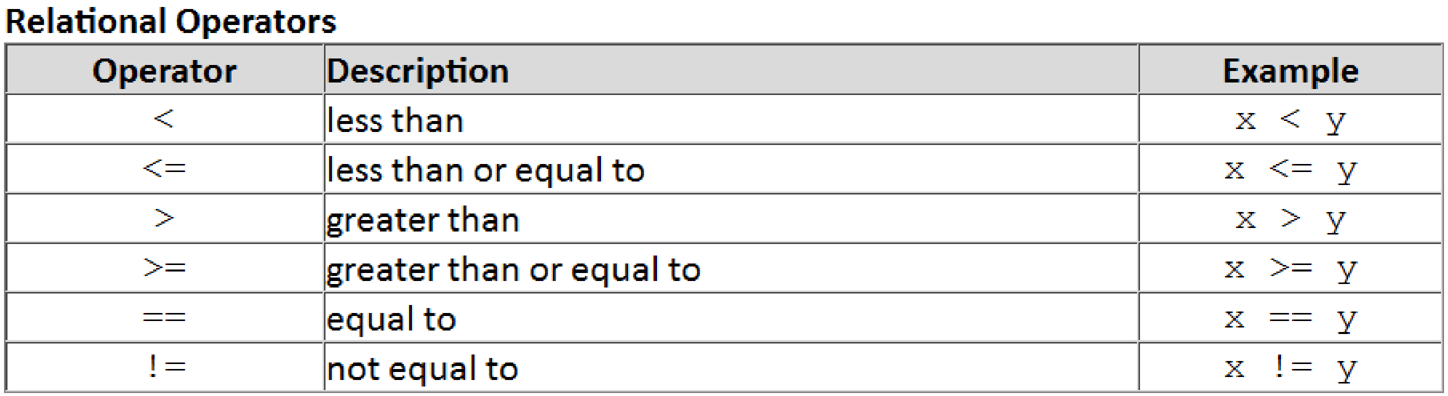
Content:

- **Relational and logical operators**
- **Boolean Expressions**
- **The `if` and `if ... else` structures**
- **The `?` Operator**
- **The `Switch` structure**
- **Nested `if` structures**
- **Example solved problems**

Relational Operators

Control statements use *relation operators* in order to compare two objects.

In C++ there are six relational operators as follows:



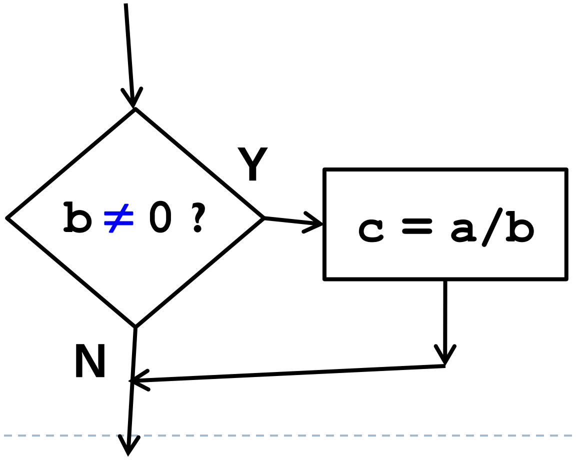
Relational Operators

Operator	Description	Example
<	less than	x < y
<=	less than or equal to	x <= y
>	greater than	x > y
>=	greater than or equal to	x >= y
==	equal to	x == y
!=	not equal to	x != y

Example:

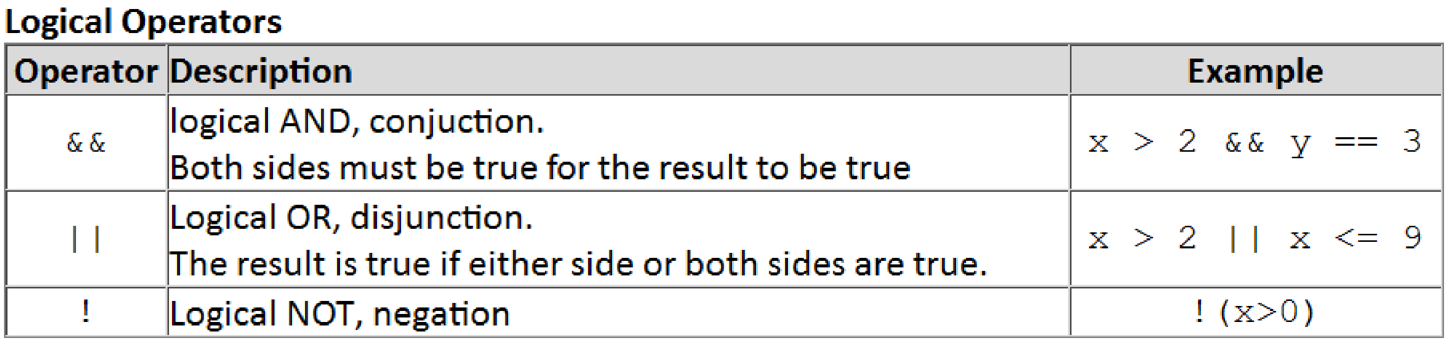
```
if ( b != 0 ) c = a/b;
```

control structure using
a relational operator



Logical Operators

Compound relation expressions can be formed using the *logical operators*:



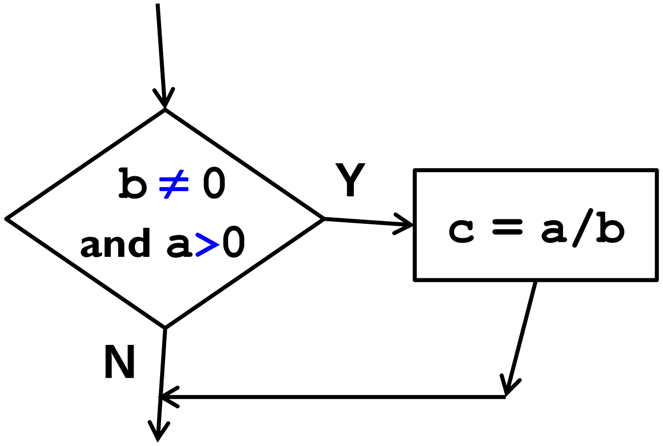
Logical Operators

Operator	Description	Example
&&	logical AND, conjuction. Both sides must be true for the result to be true	`x > 2 && y == 3`
||	Logical OR, disjunction. The result is true if either side or both sides are true.	`x > 2 || x <= 9`
!	Logical NOT, negation	`!(x>0)`

Example:

```
if ( b != 0 && a > 0 ) c = a/b;
```

control structure using a compound relational operator



b ≠ 0 and a>0

Y

c = a/b

N

The result of a relational operation such as `b != 0` is either **true** or **false**; the assignment of `c` in the selection structure `if ( b != 0 ) c = a/b;` occurs only if `b != 0` is **true**.

Example control statements and their results

```
1   double x=1.3, y=2.7, c=0.;

2   if ( x > y) cout << "x is greater than y." << endl;

3   if ( y > 0. ) cout << x/y << endl;

4   if ( x+y != 0. ) c = 1/(x+y);

5   cout << "c = " << c << endl;
```

Output

```
0.481481
c = 0.25
```

Note that there is no output from line 2 because the relation `( x > y)` is ***false***.

Boolean Expressions

Expressions that evaluate to **true** or **false** are called *Boolean Expressions.*

We can form Boolean expressions inside control statements (previous page) or in the form of assignments as follows:

```
int x=1, y=2, s;
bool u, z = true, t, w;
s = 2 > 1;
u = x > 3;
z = x <= y && y > 0;
t = y <= 0 || z;
w = !s;
```

Note that variables **u**, **z**, **t**, and **w** are declared as type **bool** and so can represent the states **true** and **false**.

Also *literal constants* **true** and **false** can be used in assignments and relational operations.

Results	s = **true**	since **2>1** (always).
	u = **false**	since **1>3** is false.
	z = **true**	since **1<=2** and **2>0** are both true.
	t = **true**	since **z** is true.
	w = **false**	since s is true, therefore its negation is false.

Example

```
bool ok;
ok = y != 0.;
if ( ok ) c = x/y;
```

variable **ok** can be **true** or **false**.

ok is assigned **true** if **y**≠0 or **false** if **y**=0.

c is assigned the result **x/y** if **ok** is **true**.

Integer represented of bools

```
 bool good = true;
 bool bad = false;
 cout << true  << endl;
 cout << false << endl;
 cout << good  << endl;
 cout <<  bad  << endl;
 cout << (good || bad) << endl;
 cout << (good && bad) << endl;
```

integer 0 represents **false**

non-zero represents **true**

```
1
0
1
0
1
0
```

Output

The `if` structure

The **`if`** statement allows conditional execution; the general form is:

```
if  (condition)  {

    statements
    .
    .

}
```

If condition is *true* then the block defined by the braces **{ . . . }** is executed.

```
if ( x+y != 0. ) {

   c = 1/(x+y);
   cout << "c = " << c << endl;

}
```

If *statements* is a single statement then the braces can be omitted:

```
if  (condition)

    statement ;
```

```
if ( x+y != 0. )
   c = 1/(x+y);

cout << "c = " << c << endl;
```

`c` is assigned only if the condition is *true*. But, the output statement will be executed in <u>any case</u>.

The if .. else structure

The **if..else** structure allows <u>both</u> outcomes of a selection to be defined.

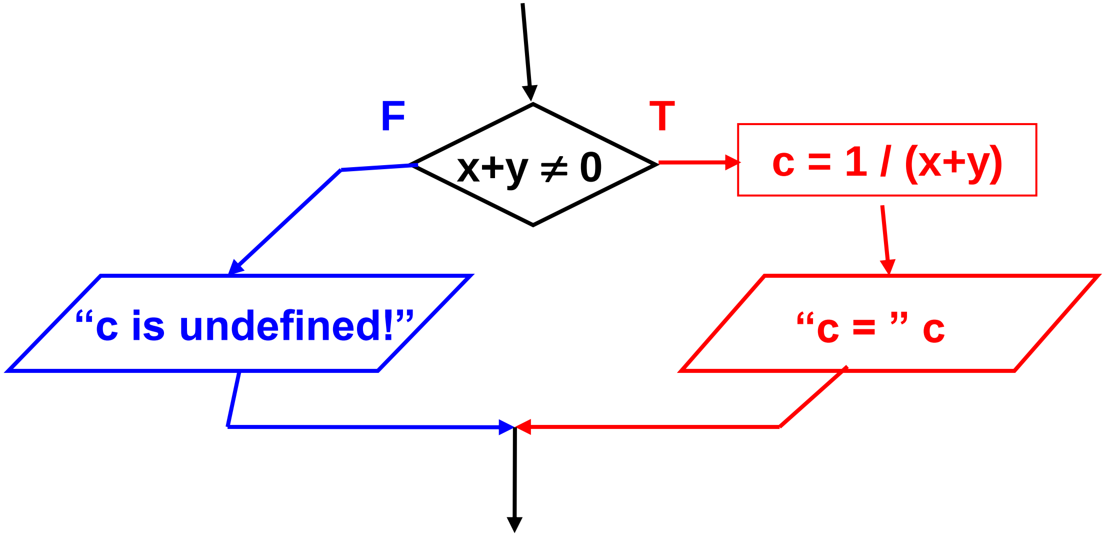
The general form is:

```
if (condition) {
   statements1
   .
   .
} else {
   statements2
   .
   .
}
```

If condition is *true* then the first block is executed, otherwise (false) the second block is executed.

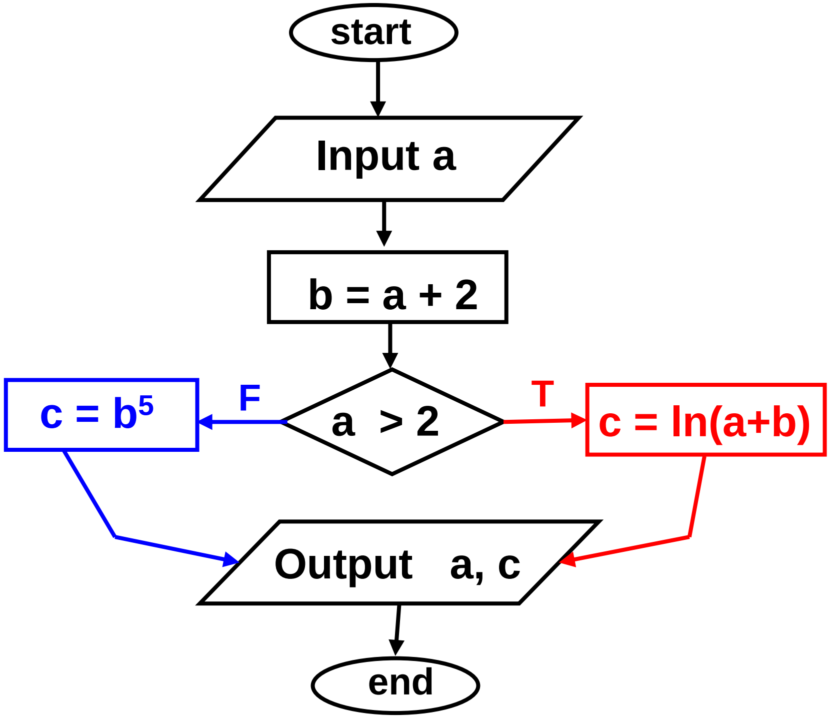
```
if ( x+y != 0. ) {
   c = 1/(x+y);
   cout << "c = " << c << endl;
} else {
   cout << "c is undefined! " << endl;
}
```

```
if ( x+y != 0. ) {

   c = 1/(x+y);
   cout << "c = " << c << endl;

} else {

   cout << "c is undefined! " << endl;

}
```



A complete example program using the `if else` structure.

```
#include <iostream>
#include <cmath>
using namespace std;

int main() {

  double a, b, c;
  cin >> a;
  b = a + 2.0;

  if ( a > 2.0 ) {
    c = log(a+b);
  } else {
    c = pow(b,5.0);
  }

  cout << a << " " << c << endl;

}
```



The if .. else if .. else structure

More levels of selection can be added with the **else if** statement.

The general form is:

```
if (condition1) {
   statements1
   .
}else if(condition2){
   statements2
   .
} else {
   statements3
   .
}
```

Example program section

```
int classCode;
cout << "Enter the class code: ";
cin >> classCode;

if (classCode==1)
   cout << "Freshman" << endl;

else if (classCode==2)
   cout << "Sophmore" << endl;

else if (classCode==3)
   cout << "Junior" << endl;

else if (classCode==4)
   cout << "Graduate" << endl;

else
   cout << "Illegal class code." << endl;
```

The ? Operator

The **?** Operator (*conditional expression operator*) provides a concise form of the `if else` structure.

The general form is:

```
( condition ) ? expression1 : expression2;
```

The value produced by this operation is either ***expression1*** or ***expression2*** depending on ***condition*** being ***true*** or ***false*** respectively.

Example:

```
max = ( x > y ) ? x : y;
```

is equivalent to

```
if ( x > y )
   max = x;
else
   max = y;
```

The switch *Statement*

- The **switch** statement is C++'s multiway branch statement.
- It is used to route execution one of several different ways.
- The general form of the statement is

```
 switch (expression) {
      case constant 1: statement sequence 1;
            break;
      case constant 2: statement sequence 2;
            break;
  .   .   .
      case constant N: statement sequence N;
            break;
      default: default statements;
}
```

Example:

```
#include <iostream>
using namespace std;
int main(){
int classCode;
 cout << "Enter the class code: ";
 cin >> classCode;

 switch (classCode){
    case(1):cout << "Freshman" << endl;
    break;
    case(2):cout << "Sophmore" << endl;
    break;
    case(3):cout << "Junior" << endl;
    break;
    case(4):cout << "Graduate" << endl;
    break;
    default:cout << "Illegal class code." << endl;
}
system("pause");
}
```

Nested `if` structures

An **`if..else`** structure can be placed in another **`if..else`** structure.

```
if (condition1) {
   if (condition2) {
      …
   } else {
      …
   }
   ...
}
```

```
if (condition1) {
      if (condition2) {
         …
      } else {
         …
      }
      ...
} else {
      if (condition3) {
         …
      } else {
         …
      }
      ...
}
```

Example:

Calculating the following function for arbitrary x values using nested **if** structures.

$$f(x)=\begin{cases} x>5 & \begin{cases} x<10 & x^2 \\ x>=10 & x+90 \end{cases} \\ x<=5 & \dfrac{125}{\mathrm{x}} \end{cases}$$

```
#include <iostream>
using namespace std;

int main() {

  double x;
  cout<<"input x\n";
  cin >> x;

  if ( x > 5.0 ) {
      if(x<10)cout<<"fx="<<x*x<<endl;
      else cout<<"fx="<<x+90<<endl;
  } else {
      if(x!=0)cout<<"fx="<<125./x<<endl;
      else cout<<"function is infinite!\n";
  }
system("pause");
}
```

…Solved problems