Conditional statement

Arijit Mondal
Dept. of Computer Science & Engineering
Indian Institute of Technology Patna
`arijit@iitp.ac.in`

Conditional statements

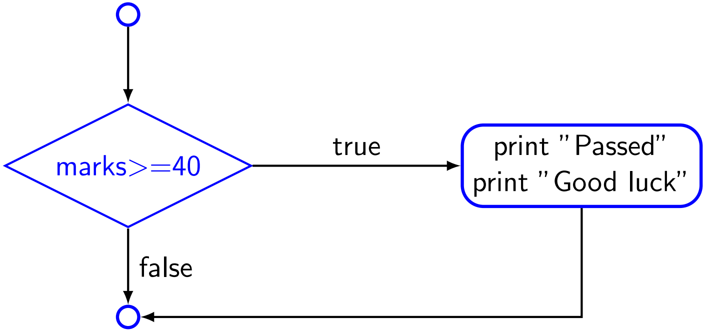
- Allow different sets of instructions to be executed depending on truth or falsity of a logical condition
- Also called Branching
- How do we specify conditions?
    - Using expressions
        - non-zero value means condition is true
        - value 0 means condition is false
    - Usually logical expressions, but can be any expression
        - The value of the expression will be used

Branching: if statement

```
if(expression)
  statement;

if(expression){
  Block of statements;
}
```

Branching



Branching: if-else statement

```
if(expression){
  Block of statements;
}
else{
  Block of statements;
}
```

```
if(expression){
  Block of statements;
}
else if(expression){
  Block of statements;
}
else{
  Block of statements;
}
```

Example: grade computation

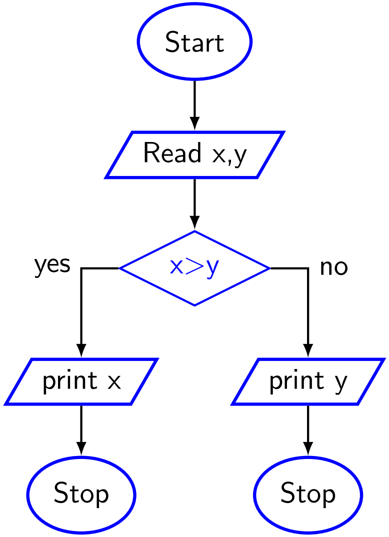
```
void main(){
  int marks;
  scanf("%d",&marks);
  if(marks>=80)
    printf("A");
  else if(marks>=70)
    printf("B");
  else if(marks>=60)
    printf("C");
  else printf("failed");
}
```

Example: grade computation

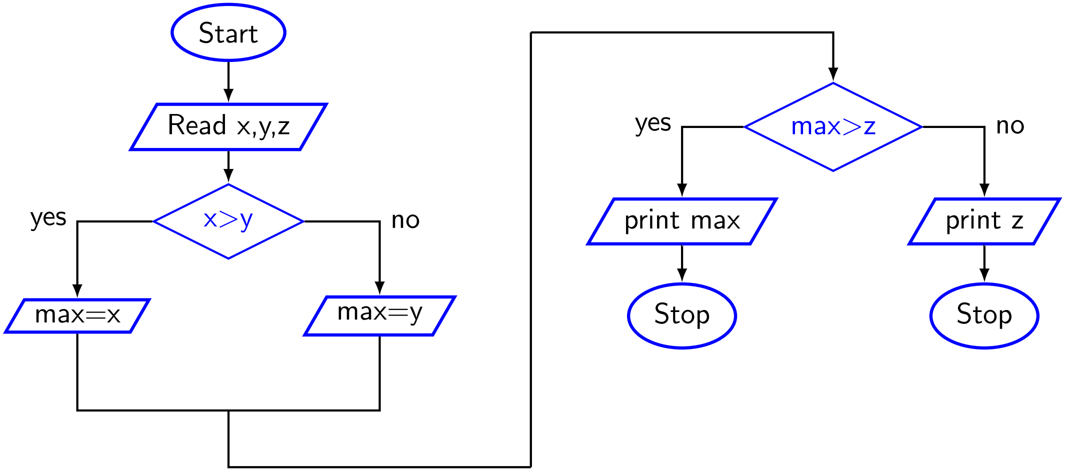
```
void main(){
  int marks;
  scanf("%d",&marks);
  if(marks>=80){
    printf("A");
    printf("Great Job");
  } else if(marks>=70)
    printf("B");
  else{
    printf("failed");
    printf("study more");
  }
}
```

Maximum of two numbers

```
#include <stdio.h>
void main()
{
  int x,y;
  scanf("%d%d",&x,&y);
  if(x>y) printf("Largest is %d\n",x);
  else printf("Largest is %d\n",y);
}
```



Max of three numbers



Maximum of three numbers

```
#include <stdio.h>
void main()
{
  int x,y,z,max;
  scanf("%d%d%d",&x,&y,&z);
  if(x>y) max=x;
  else max=y;
  if(max>z) printf("%d",max);
  else printf("%d",z);
}
```

Maximum of three numbers

```
void main(){
  int x,y,z;
  scanf("%d%d%d",&x,&y,&z);
  if((x>=y)&&(x>=z))
    printf("%d",x);
  if((y>=x)&&(y>=z))
    printf("%d",y);
  if((z>=y)&&(z>=x))
    printf("%d",x);
}
```

Equality (==) vs Assignment (=) operators

- Dangerous error
  - Does not ordinarily cause syntax errors
  - Any expression that produces a value can be used in control structures
  - Nonzero values are true, zero values are false
- Example:

```
if ( payCode = 4 )
  printf( "You get a bonus!\n" );
```

Equality (==) vs Assignment (=) operators

- Dangerous error
  - Does not ordinarily cause syntax errors
  - Any expression that produces a value can be used in control structures
  - Nonzero values are true, zero values are false

- Example:

```
if ( payCode = 4 )
  printf( "You get a bonus!\n" );
```

  - **Will always print the line**

Nesting of if-else structures

- It is possible to nest if-else statements, one within another
- All "if" statements may not be having the "else" part
  - Confusion??
- Rule to be remembered:
  - An "else" clause is associated with the closest preceding unmatched "if"

Dangling else problem

- `if(exp1) if(exp2) stamta else stmtb`

```
if(exp1){
  if(exp2)
    stmta
  else
    stmtb
}
```

```
if(exp1){
  if(exp2)
    stmta
}
  else
    stmtb
```

More examples

```
if e1 s1
else if e2 s2
```

More examples

```
if e1 s1
else if e2 s2
```

```
if e1 s1
else { if e2 s2}
```

More examples

```
if e1 s1
else if e2 s2
```

```
if e1 s1
else { if e2 s2}
```

```
if e1 s1
else if e2 s2
else s3
```

More examples

```
if e1 s1
else if e2 s2
```

```
if e1 s1
else { if e2 s2}
```

```
if e1 s1
else if e2 s2
else s3
```

```
if e1 s1
else { if e2 s2 else s3}
```

More examples

```
if e1 s1
else if e2 s2
```

```
if e1 s1
else { if e2 s2}
```

```
if e1 s1
else if e2 s2
else s3
```

```
if e1 s1
else { if e2 s2 else s3}
```

```
if e1 if e2 s1
else s2
else s3
```

More examples

```
if e1 s1
else if e2 s2
```

```
if e1 s1
else { if e2 s2}
```

```
if e1 s1
else if e2 s2
else s3
```

```
if e1 s1
else { if e2 s2 else s3}
```

```
if e1 if e2 s1
else s2
else s3
```

```
if e1 {if e2 s1 else s2}
else s3
```

Conditional operator ?:

- This makes use of an expression that is either non-0 or 0. An appropriate value is selected, depending on the value of the expression
- Example: instead of writing

```
if (balance > 5000)
  interest = balance * 0.2;
else interest = balance * 0.1;

interest = (balance > 5000) ?  balance * 0.2 :  balance * 0.1;
```

More examples

-
  ```
  if (((a >10) && (b < 5))
     x = a + b;
  else x = 0;
  ```

More examples

-
  ```
  if (((a >10) && (b < 5))
     x = a + b;
  else x = 0;

  x = ((a > 10) && (b < 5)) ?  a + b :  0
  ```

-
  ```
  if (marks >= 60)
      printf("Passed \n");
  else printf("Failed \n");
  ```

More examples

-
  ```
  if (((a >10) && (b < 5))
    x = a + b;
  else x = 0;

  x = ((a > 10) && (b < 5)) ?  a + b :  0
  ```

-
  ```
  if (marks >= 60)
     printf("Passed \n");
  else printf("Failed \n");

  (marks >= 60) ?  printf("Passed \n") :  printf("Failed \n");
  ```

switch statement

- An alternative to writing lots of if-else in some special cases
- This causes a particular group of statements to be chosen from several available groups based on equality tests only
- Uses switch statement and case labels

switch statement (contd.)

- Syntax

```
switch (expression) {
  case const-expr-1:  s-1
  case const-expr-2:  s-2
  ⋮
  default:  s
}
```

- `expression` is any integer-valued expression
- `const-expr-1, const-expr-2,...` are any constant integer valued expressions
  - Values must be distinct
- `s-1, s-2, ...,s-m, s` are statements/compound statements
- Default is optional, and can come anywhere (not necessarily at the end as shown)

Behavior of switch

- `expression` is first evaluated
- It is then compared with `const-expr-1, const-expr-2,...` for equality in order
- If it matches any one, all statements from that point till the end of the switch are executed (including statements for default, if present)
  - Use `break` statements if you do not want this
- Statements corresponding to `default`, if present, are executed if no other expression matches

Example

```
int x;
scanf("%d",&x);
switch (x){
  case 1:  printf("one \n");
  case 2:  printf("two \n");
  default:  printf("Not one or two \n");
}
```

- Let entered value is 1, output will be

Example

```
int x;
scanf("%d",&x);
switch (x){
  case 1:  printf("one \n");
  case 2:  printf("two \n");
  default:  printf("Not one or two \n");
}
```

- Let entered value is 1, output will be

```
One
Two
Not one or two
```

Example: correct version

```
int x;
scanf("%d",&x);
switch (x){
  case 1:  printf("one \n");
           break ;
  case 2:  printf("two \n");
           break ;
  default:  printf("Not one or two \n");
}
```

- Let entered value is 1, output will be

  One

Rounding a digit

```
switch (digit){
  case 0:
  case 1:
  case 2:
  case 3:
  case 4:  result=0; printf("Round down\n");break ;
  case 5:
  case 6:
  case 7:
  case 8:
  case 9:  result=10; printf("Round up\n");break ;
}
```

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
- Can be used with other statements also