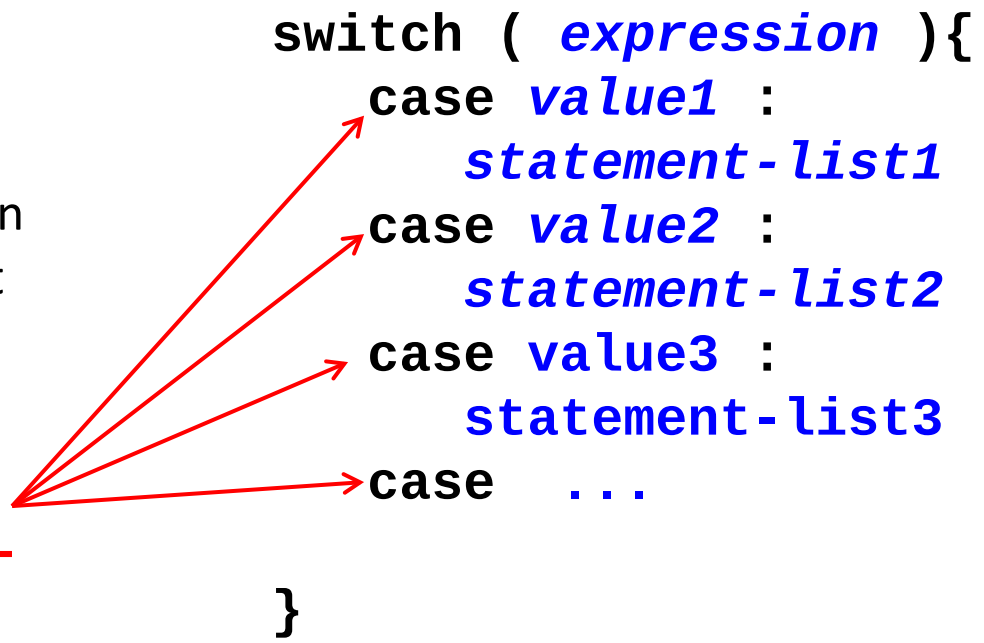
Switch Statements

Comparing Exact Values

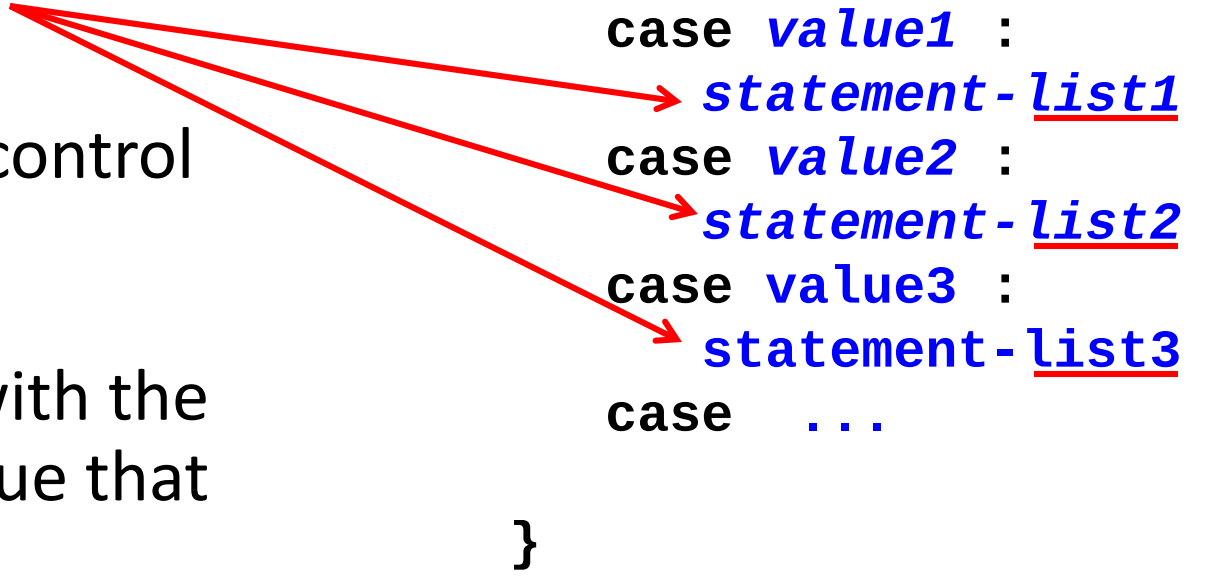
The Switch Statement: *Syntax*

- The *switch statement* provides another way to decide which statement to execute next
- The `switch` statement evaluates an expression, then attempts to match the result to one of several possible *cases*
- The match must be an exact match.



The Switch Statement

- Each **case** contains a value and a list of statements
- The flow of control transfers to statement associated with the first case value that matches



Switch Syntax

- The general syntax of a `switch` statement is:

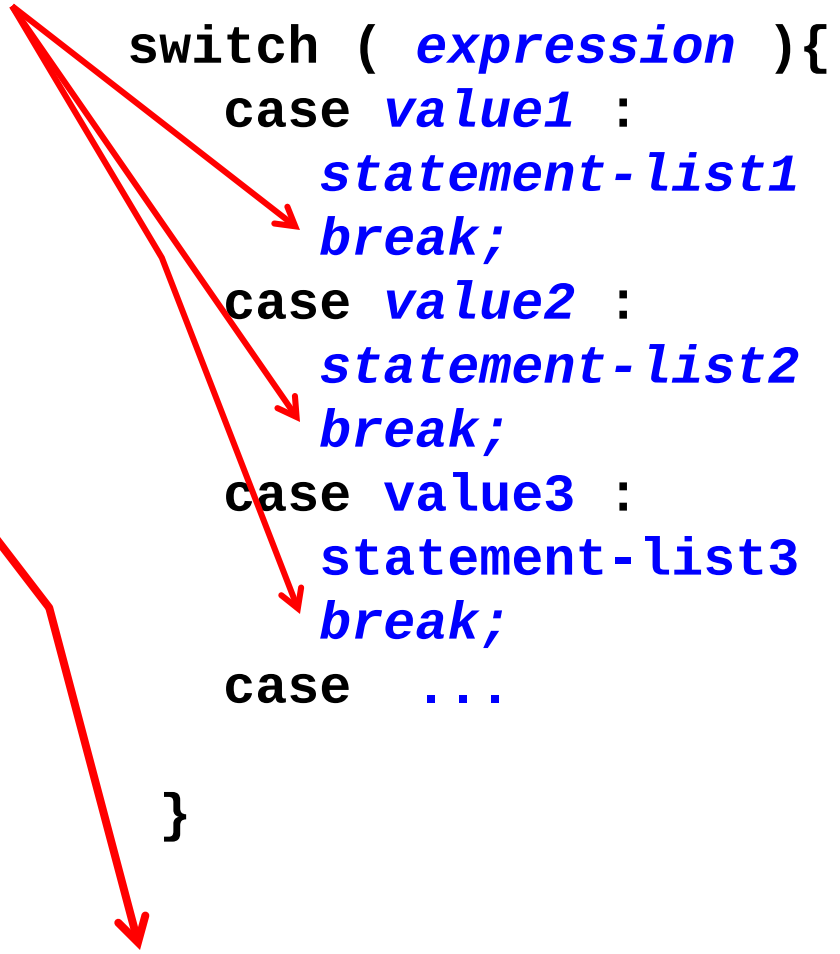
switch and case are reserved words

```
switch ( expression ){
   case value1 :
      statement-list1
   case value2 :
      statement-list2
   case value3 :
      statement-list3
   case  ...

}
```

If *expression* matches *value3*, control jumps to here

The Switch Statement

- The *break statement* can be used as the last statement in each case's statement list
- A `break` statement causes control to transfer to the end of the `switch` statement
- *If* a `break` statement is not used, the flow of control will continue into the next case

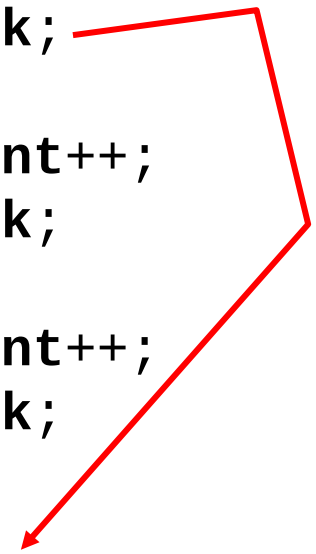


Switch Example

- Example of the **switch** statement:

If '**1**' is true, then **aCount** is incremented by 1, then *breaks* to the bottom and *outside* of the switch

```
switch (test){
    case 1:
        aCount++;
        break;
    case 2:
        bCount++;
        break;
    case 3:
        cCount++;
        break;
}
```



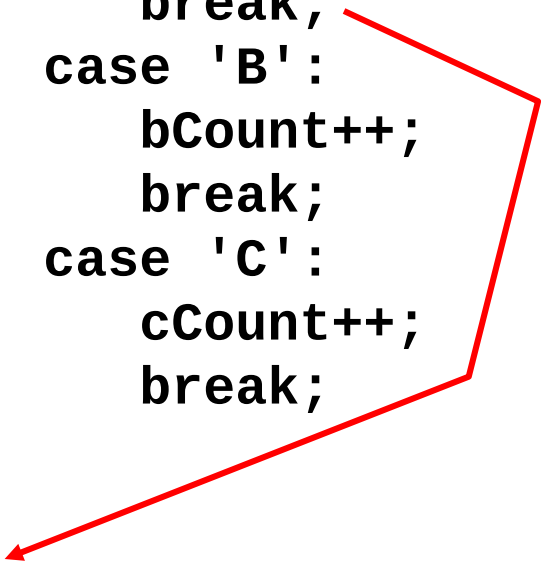
```
int test = 0;

int aCount = 1;
int bCount = 2;
int cCount = 3;

test = aCount;
```

Switch: No breaks!!!

- Another Example:

If there is no *break* statement, then the switch will move down through every **case** and **do** all the statements until done with the switch.

```
switch (option){
   case 'A':
      aCount++;
      break;
   case 'B':
      bCount++;
      break;
   case 'C':
      cCount++;
      break;
}
```



```
switch (option){
   case 'A':
      aCount++;
   case 'B':
      bCount++;
   case 'C':
      cCount++;
}
```

Switch - Default

- A `switch` statement can have an optional *default case*
- The default case has no associated value and simply uses the reserved word `default`
- If the default case is present, control will transfer to it if *no other case value matches*
- If there is no default case, and no other value matches, control falls through to the statement after the switch

The switch Statement

- Switch with default case:

```
switch (option){
   case 'A':
      aCount++;
      break;
   case 'B':
      bCount++;
      break;
   case 'C':
      cCount++;
      break;
   default:
      otherCount++;
      break;
}
```

To Switch or Not to Switch

- The expression of a `switch` statement must result in an ***integral type***, meaning an integer (`byte`, `short`, `int`, `long`) or a `char`
- It **cannot** be a `boolean` value or a floating point value (`float` or `double`)
- The implicit boolean condition in a `switch` statement is **equality: A == B**
- **You cannot perform relational checks with a `switch` statement**

To switch or not to switch….

…that is the question!