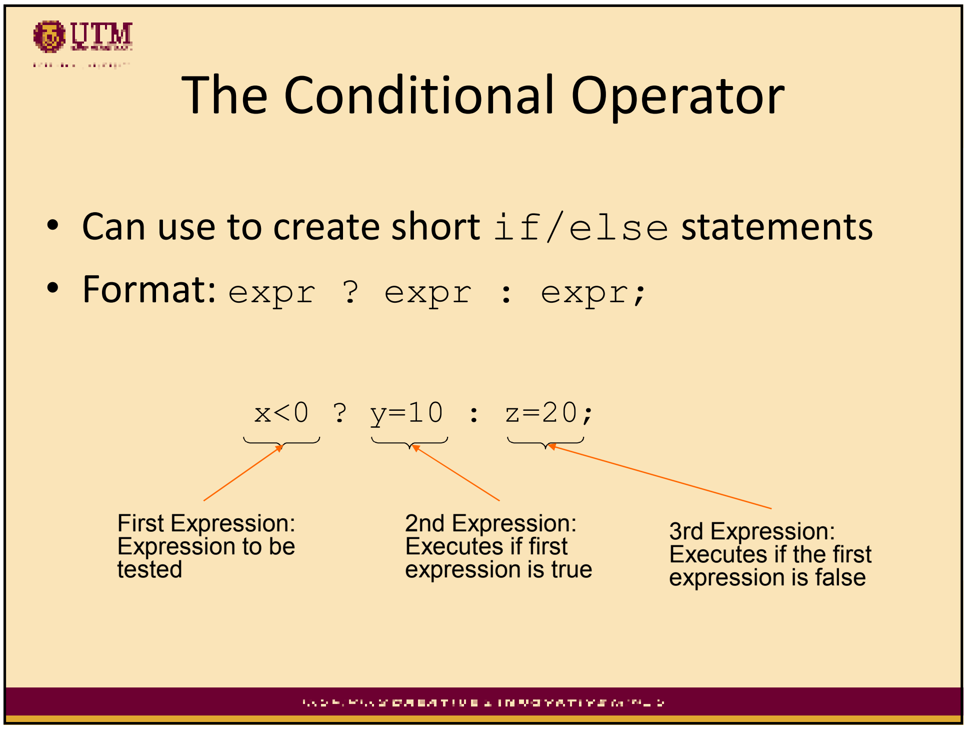


# The Conditional Operator

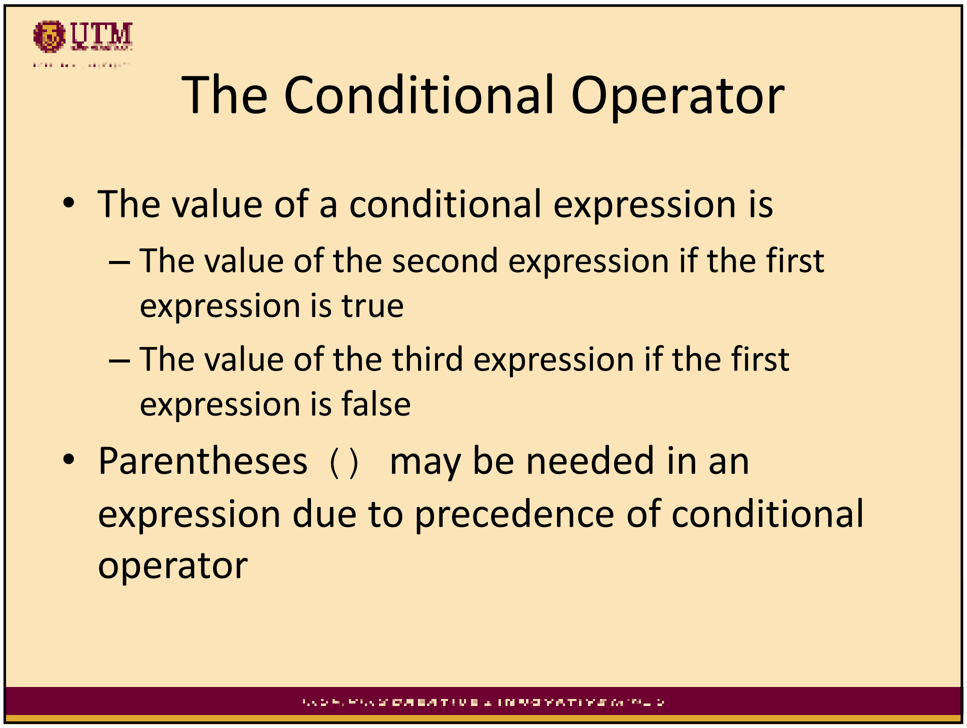
- Can use to create short `if/else` statements
- Format: `expr ? expr : expr;`

```
x<0 ? y=10 : z=20;
```

First Expression: Expression to be tested

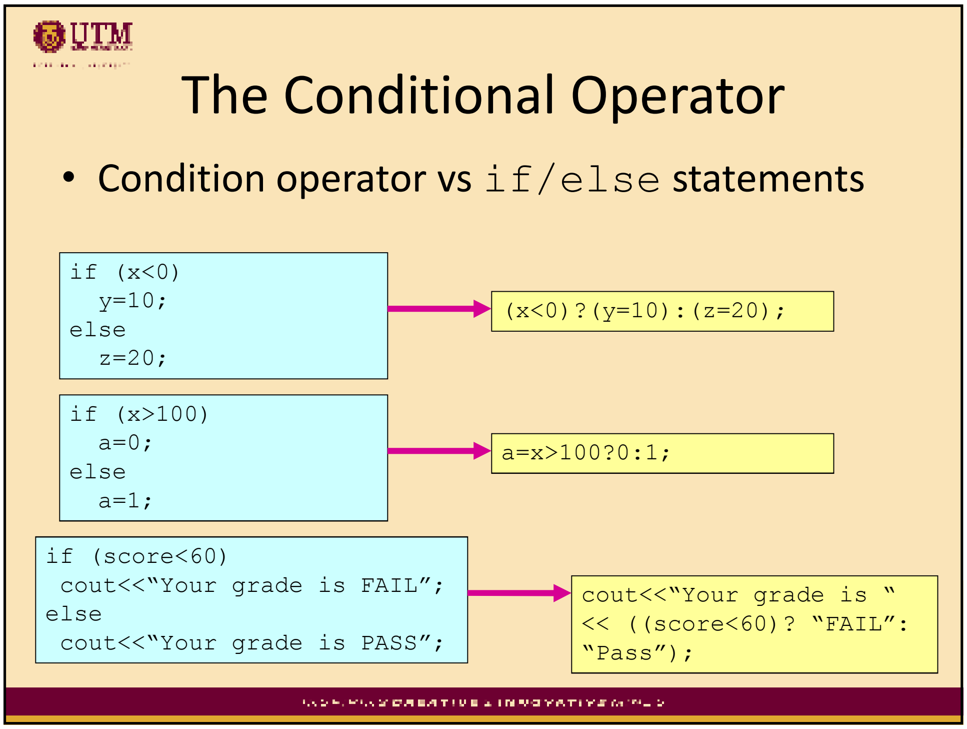
2nd Expression: Executes if first expression is true

3rd Expression: Executes if the first expression is false



# The Conditional Operator

- The value of a conditional expression is
  - The value of the second expression if the first expression is true
  - The value of the third expression if the first expression is false
- Parentheses `()` may be needed in an expression due to precedence of conditional operator



# The Conditional Operator

- Condition operator vs `if/else` statements

```
if (x<0)
  y=10;
else
  z=20;
```

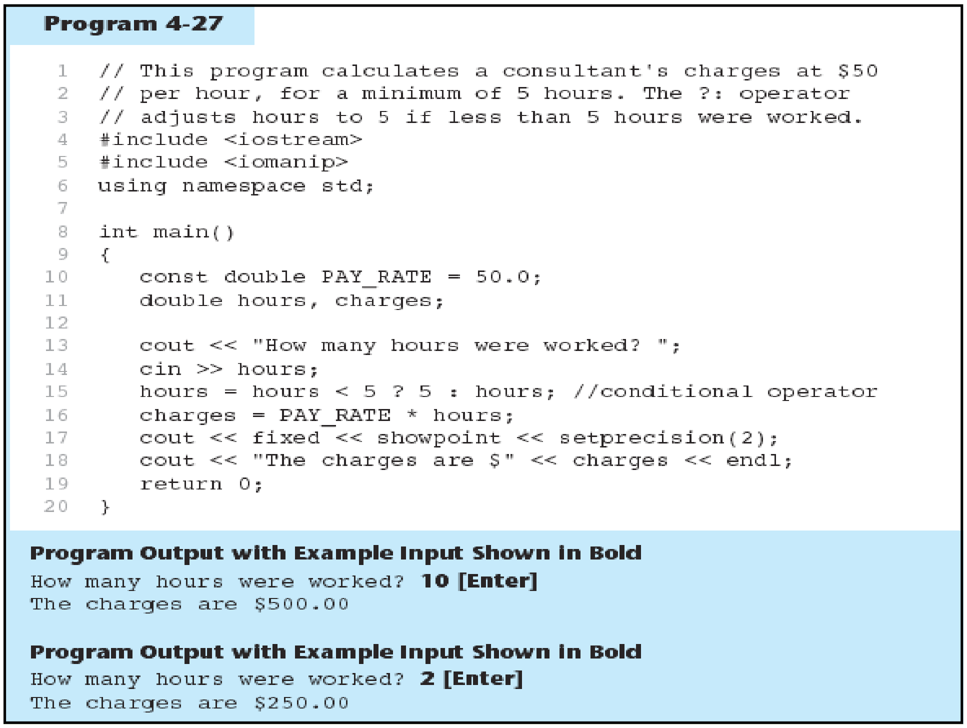
```
(x<0)?(y=10):(z=20);
```

```
if (x>100)
  a=0;
else
  a=1;
```

```
a=x>100?0:1;
```

```
if (score<60)
 cout<<"Your grade is FAIL";
else
 cout<<"Your grade is PASS";
```

```
cout<<"Your grade is "
<< ((score<60)? "FAIL":
"Pass");
```



**Program 4-27**

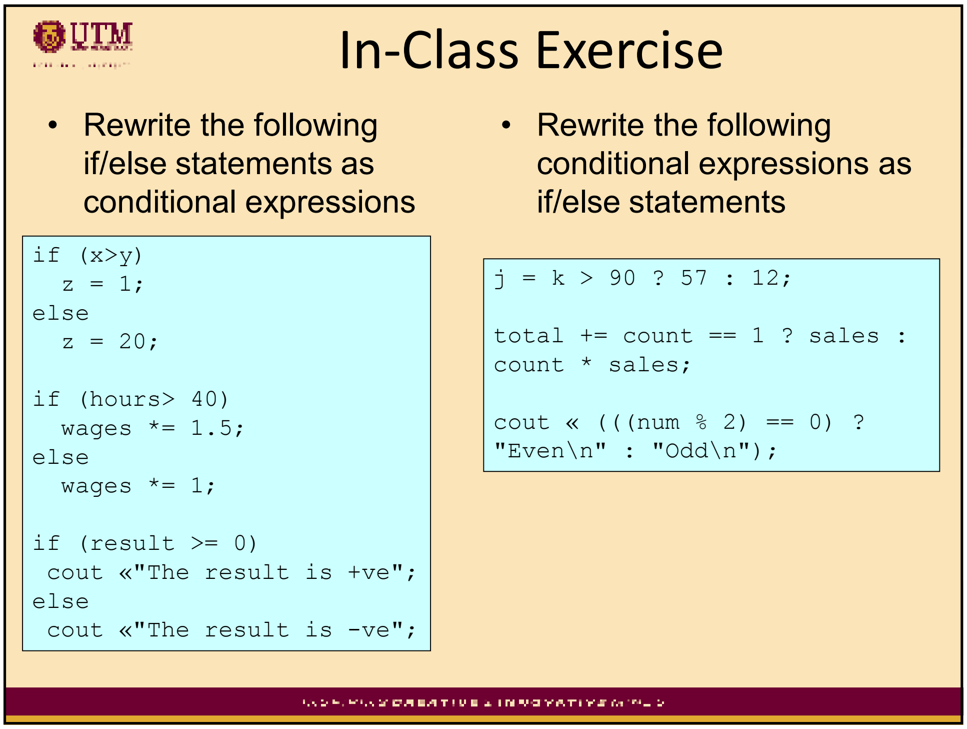
```
// This program calculates a consultant's charges at $50
// per hour, for a minimum of 5 hours. The ?: operator
// adjusts hours to 5 if less than 5 hours were worked.
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
   const double PAY_RATE = 50.0;
   double hours, charges;

   cout << "How many hours were worked? ";
   cin >> hours;
   hours = hours < 5 ? 5 : hours; //conditional operator
   charges = PAY_RATE * hours;
   cout << fixed << showpoint << setprecision(2);
   cout << "The charges are $" << charges << endl;
   return 0;
}
```

**Program Output with Example Input Shown in Bold**

```
How many hours were worked? 10 [Enter]
The charges are $500.00
```

**Program Output with Example Input Shown in Bold**

```
How many hours were worked? 2 [Enter]
The charges are $250.00
```



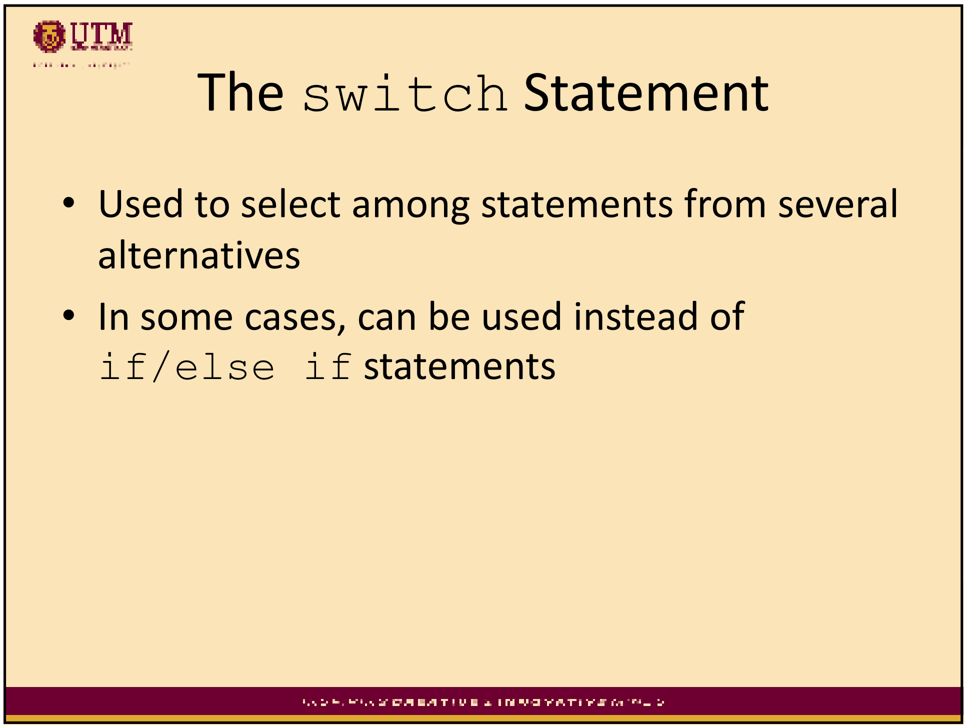
## In-Class Exercise

- Rewrite the following if/else statements as conditional expressions

```
if (x>y)
  z = 1;
else
  z = 20;

if (hours> 40)
  wages *= 1.5;
else
  wages *= 1;

if (result >= 0)
 cout «"The result is +ve";
else
 cout «"The result is -ve";
```

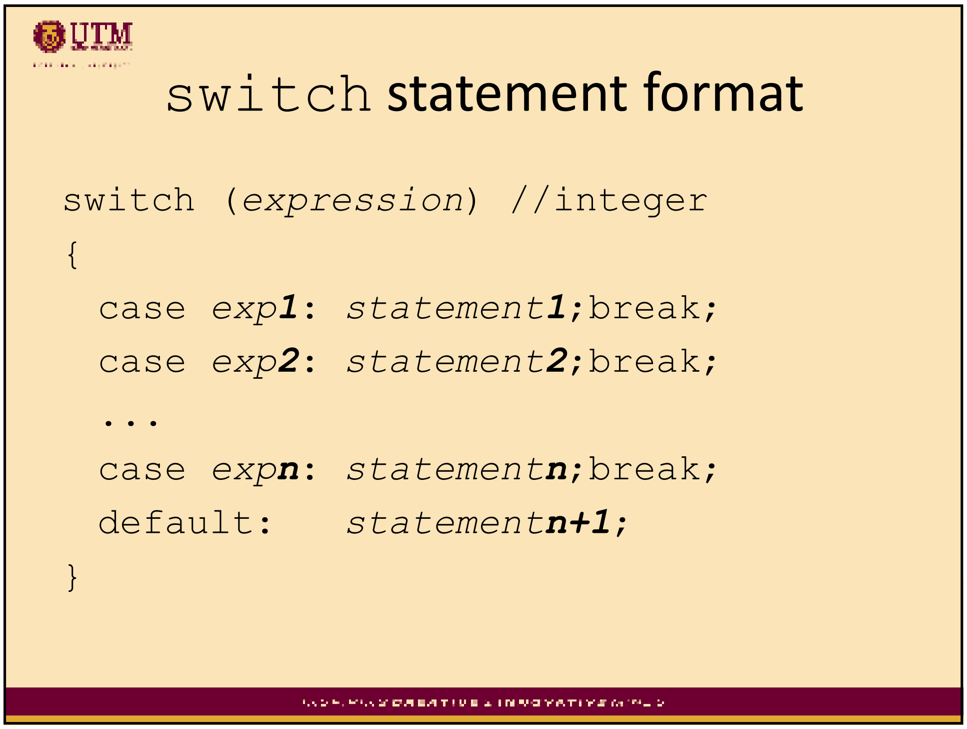
- Rewrite the following conditional expressions as if/else statements

```
j = k > 90 ? 57 : 12;

total += count == 1 ? sales :
count * sales;

cout « (((num % 2) == 0) ?
"Even\n" : "Odd\n");
```



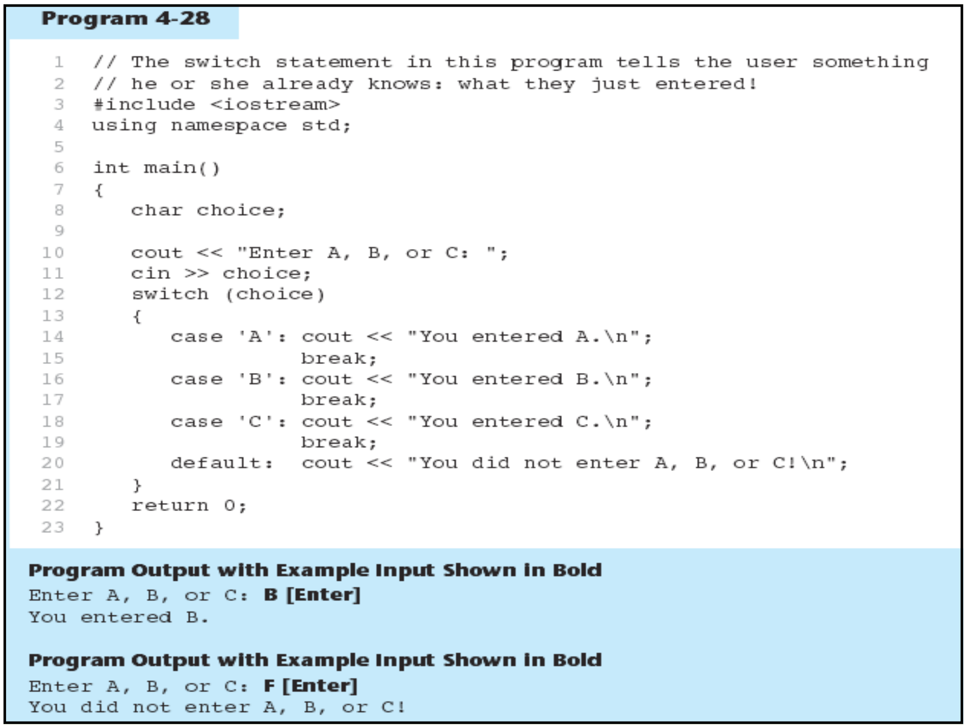
## The `switch` Statement

- Used to select among statements from several alternatives
- In some cases, can be used instead of `if/else if` statements



## `switch` statement format

```
switch (expression) //integer
{
  case exp1: statement1;break;
  case exp2: statement2;break;
  ...
  case expn: statementn;break;
  default:   statementn+1;
}
```



**Program 4-28**

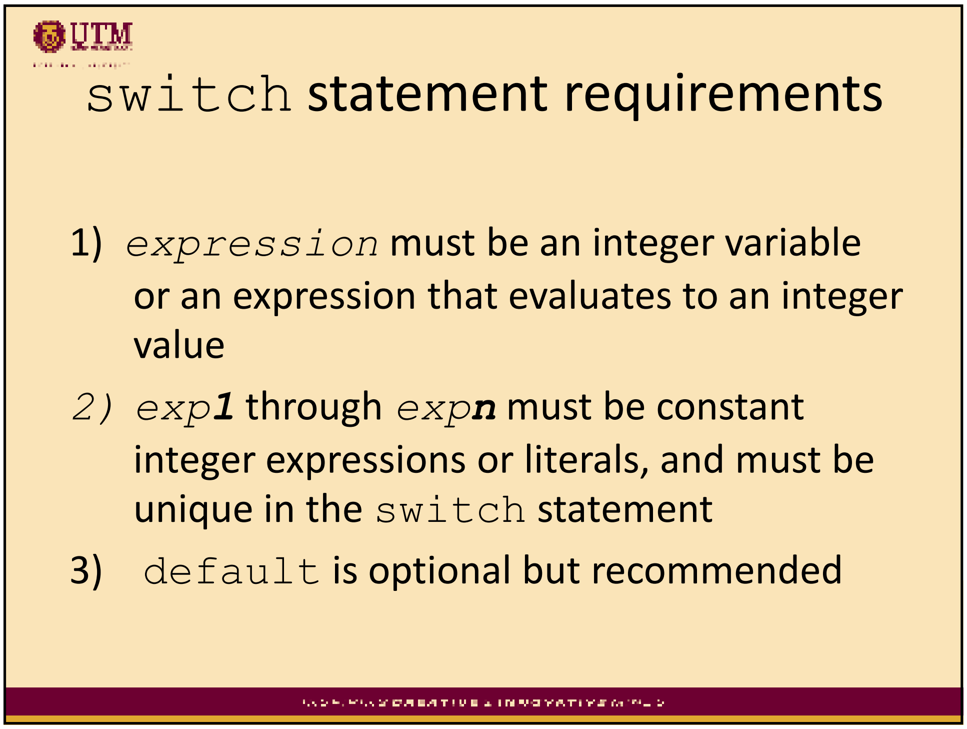
```
// The switch statement in this program tells the user something
// he or she already knows: what they just entered!
#include <iostream>
using namespace std;

int main()
{
   char choice;

   cout << "Enter A, B, or C: ";
   cin >> choice;
   switch (choice)
   {
      case 'A': cout << "You entered A.\n";
                break;
      case 'B': cout << "You entered B.\n";
                break;
      case 'C': cout << "You entered C.\n";
                break;
      default:  cout << "You did not enter A, B, or C!\n";
   }
   return 0;
}
```

**Program Output with Example Input Shown in Bold**

```
Enter A, B, or C: B [Enter]
You entered B.
```

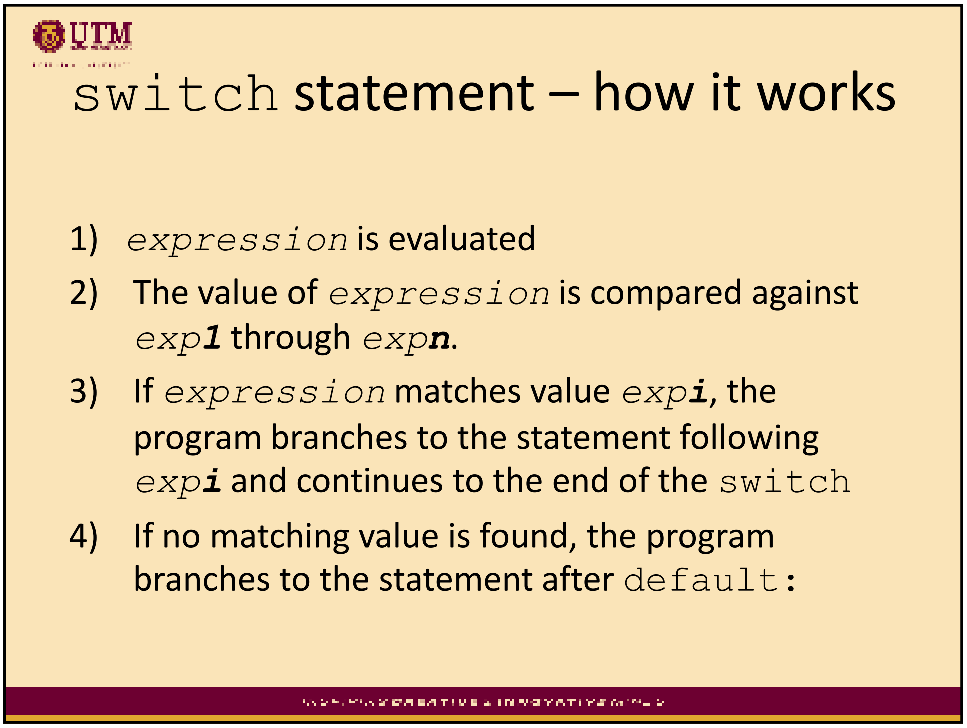
**Program Output with Example Input Shown in Bold**

```
Enter A, B, or C: F [Enter]
You did not enter A, B, or C!
```



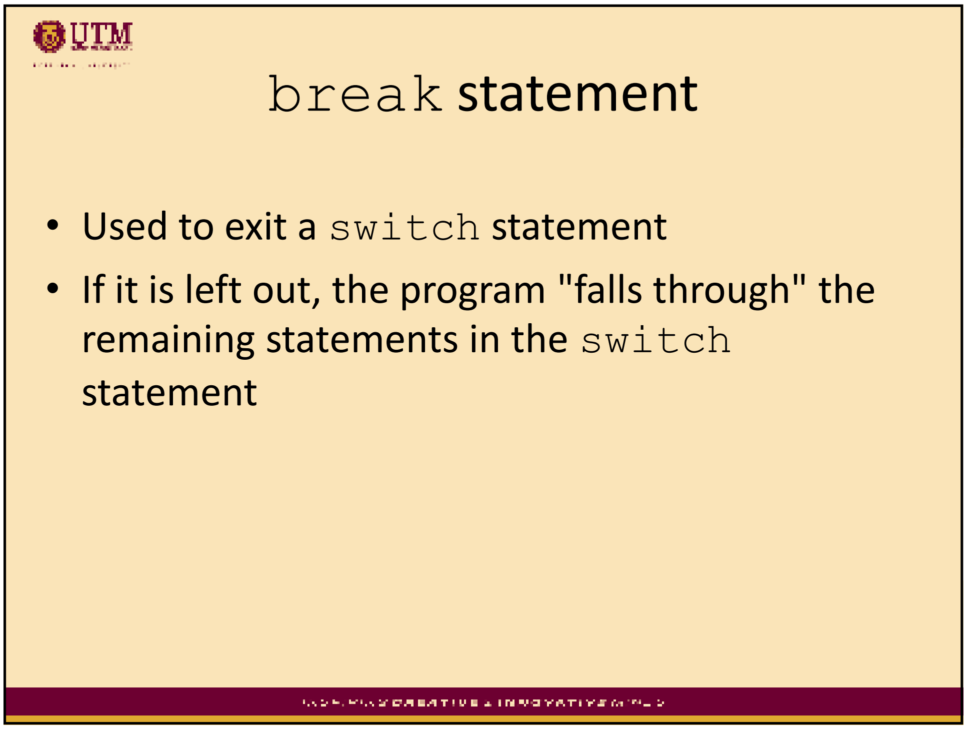
## switch statement requirements

1) *expression* must be an integer variable or an expression that evaluates to an integer value
*2)* *exp****1*** through *exp****n*** must be constant integer expressions or literals, and must be unique in the switch statement
3) default is optional but recommended



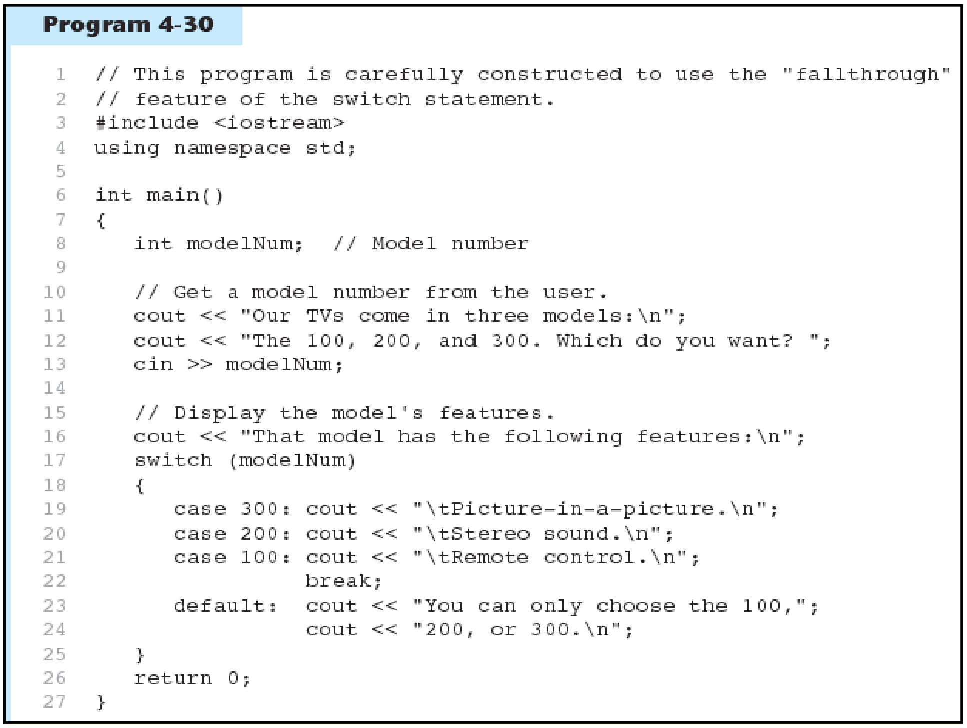
## switch statement – how it works

1) *expression* is evaluated
2) The value of *expression* is compared against *exp****1*** through *exp****n***.
3) If *expression* matches value *exp****i***, the program branches to the statement following *exp****i*** and continues to the end of the switch
4) If no matching value is found, the program branches to the statement after default:



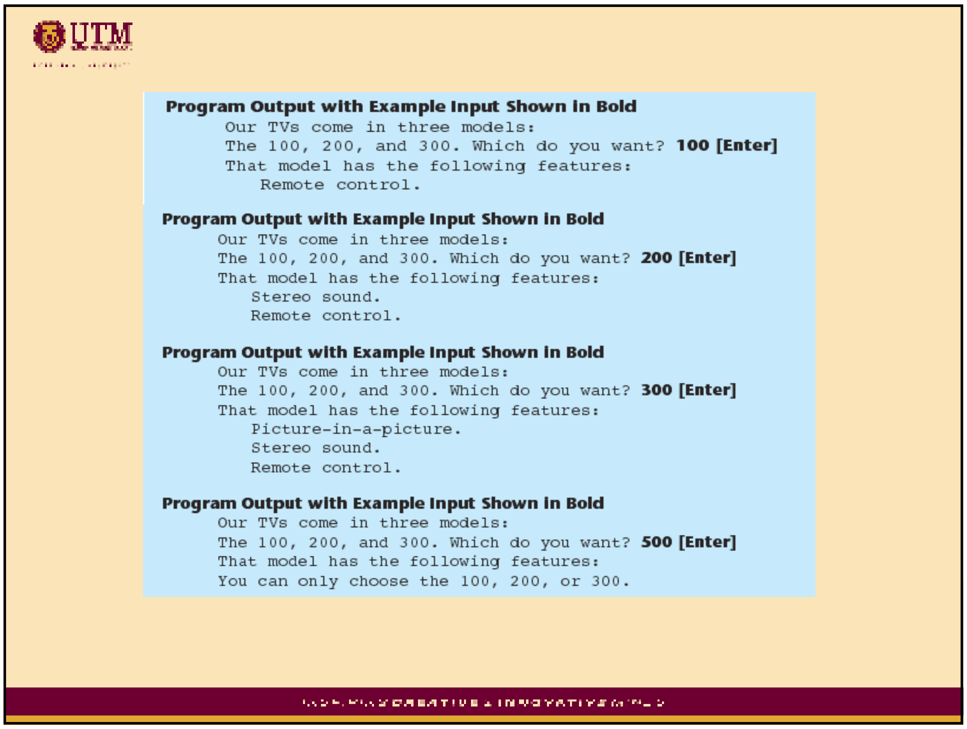
## break statement

- Used to exit a switch statement
- If it is left out, the program "falls through" the remaining statements in the switch statement



**Program 4-30**

```
// This program is carefully constructed to use the "fallthrough"
// feature of the switch statement.
#include <iostream>
using namespace std;

int main()
{
   int modelNum;   // Model number

   // Get a model number from the user.
   cout << "Our TVs come in three models:\n";
   cout << "The 100, 200, and 300. Which do you want? ";
   cin >> modelNum;

   // Display the model's features.
   cout << "That model has the following features:\n";
   switch (modelNum)
   {
      case 300: cout << "\tPicture-in-a-picture.\n";
      case 200: cout << "\tStereo sound.\n";
      case 100: cout << "\tRemote control.\n";
                break;
      default:  cout << "You can only choose the 100,";
                cout << "200, or 300.\n";
   }
   return 0;
}
```



**Program Output with Example Input Shown in Bold**

```
Our TVs come in three models:
The 100, 200, and 300. Which do you want? 100 [Enter]
That model has the following features:
    Remote control.
```

**Program Output with Example Input Shown in Bold**

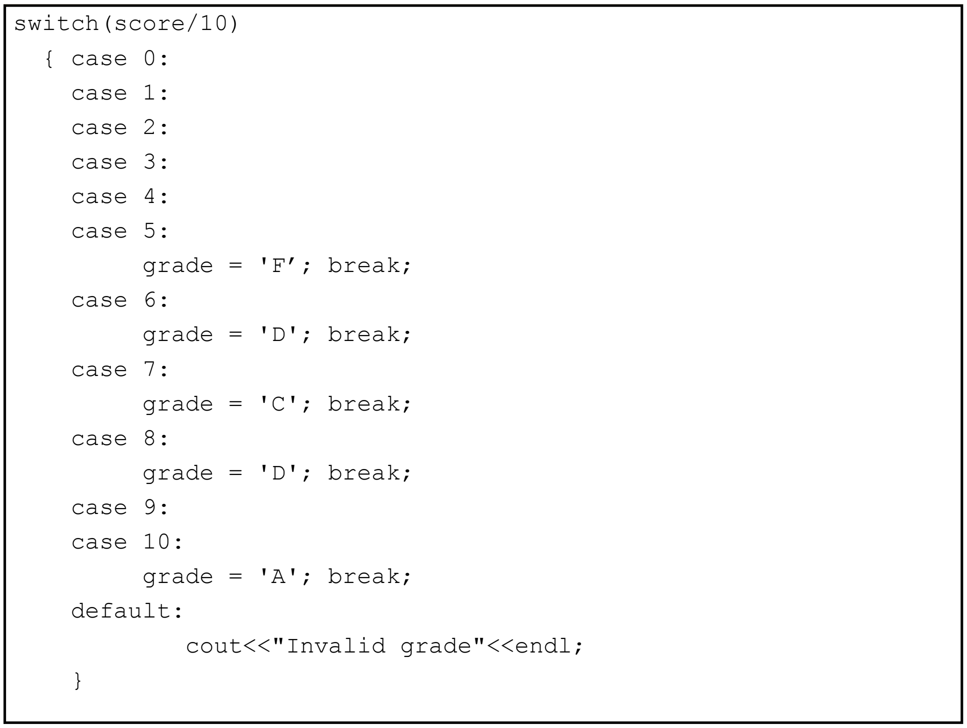
```
Our TVs come in three models:
The 100, 200, and 300. Which do you want? 200 [Enter]
That model has the following features:
    Stereo sound.
    Remote control.
```

**Program Output with Example Input Shown in Bold**

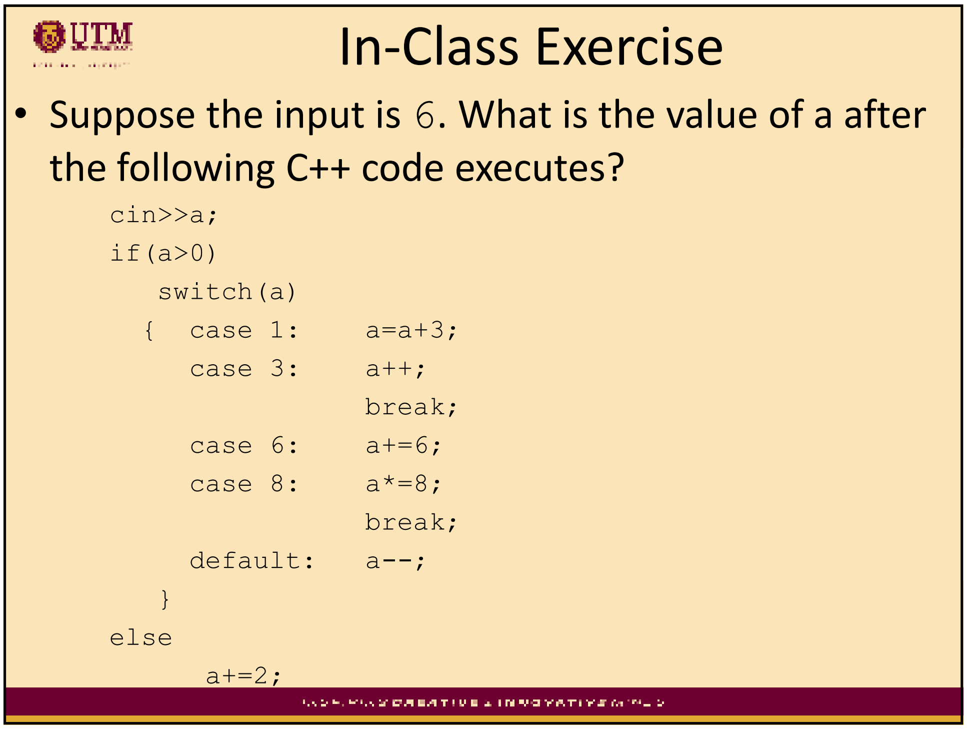
```
Our TVs come in three models:
The 100, 200, and 300. Which do you want? 300 [Enter]
That model has the following features:
    Picture-in-a-picture.
    Stereo sound.
    Remote control.
```

**Program Output with Example Input Shown in Bold**

```
Our TVs come in three models:
The 100, 200, and 300. Which do you want? 500 [Enter]
That model has the following features:
You can only choose the 100, 200, or 300.
```



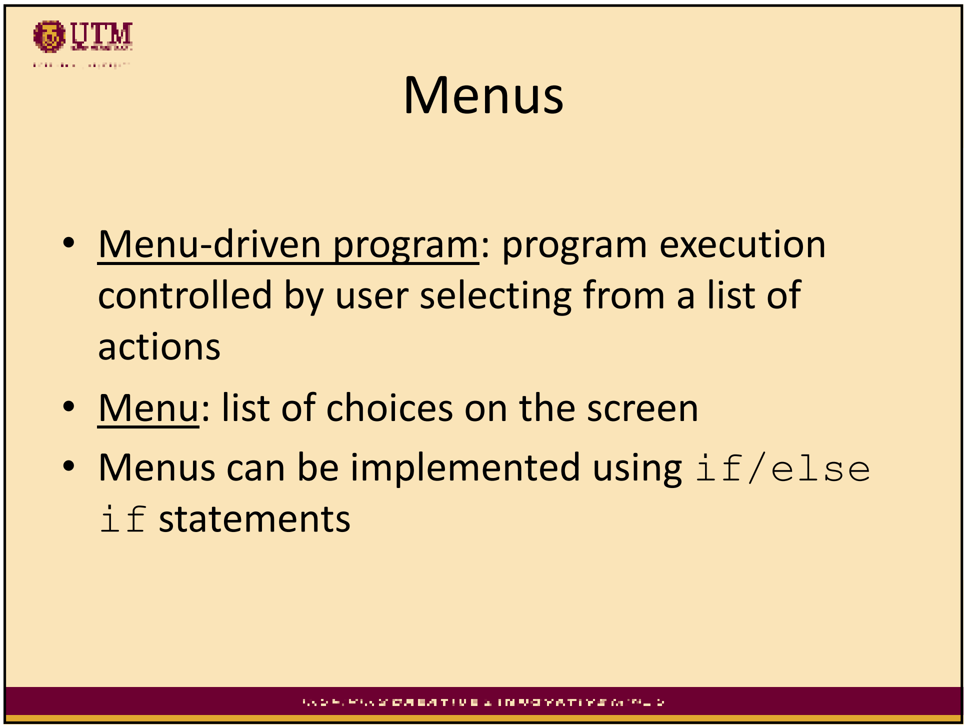
```
switch(score/10)
  { case 0:
    case 1:
    case 2:
    case 3:
    case 4:
    case 5:
         grade = 'F’; break;
    case 6:
         grade = 'D'; break;
    case 7:
         grade = 'C'; break;
    case 8:
         grade = 'D'; break;
    case 9:
    case 10:
         grade = 'A'; break;
    default:
            cout<<"Invalid grade"<<endl;
    }
```



# In-Class Exercise

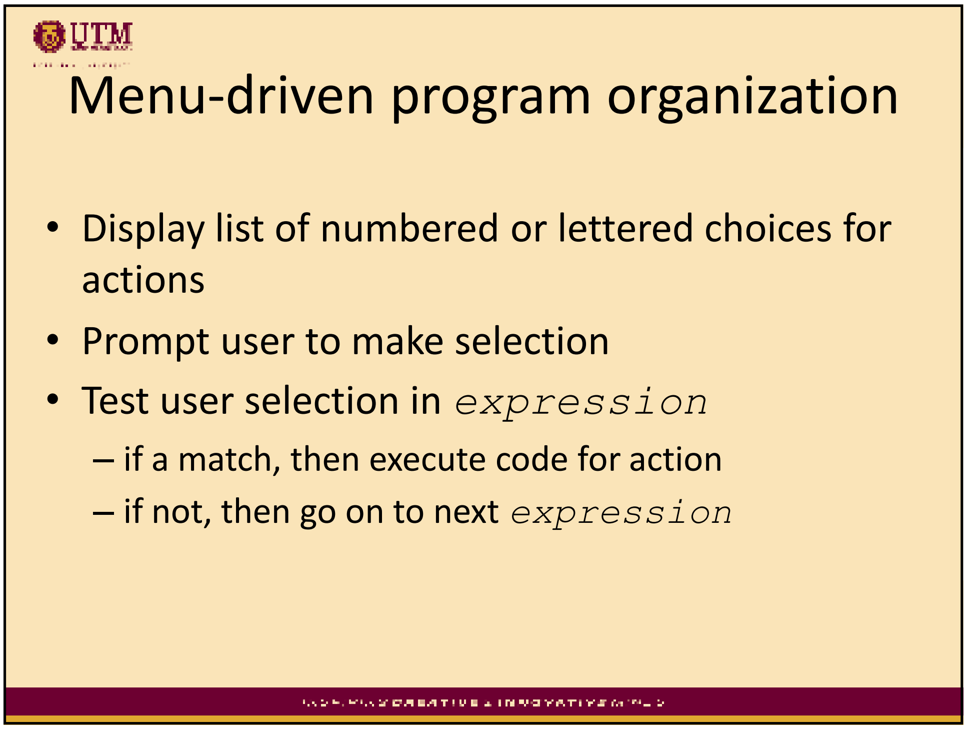
- Suppose the input is 6. What is the value of a after the following C++ code executes?

```
cin>>a;
if(a>0)
   switch(a)
 {   case 1:     a=a+3;
     case 3:     a++;
                 break;
     case 6:     a+=6;
     case 8:     a*=8;
                 break;
     default:    a--;
   }
else
      a+=2;
```



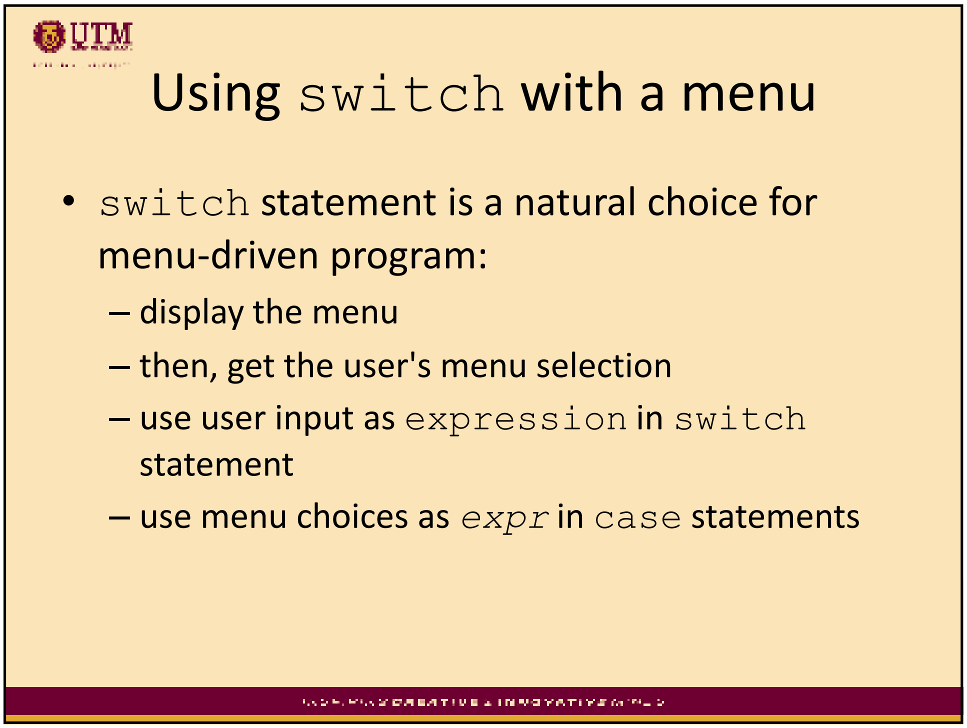
# Menus

- Menu-driven program: program execution controlled by user selecting from a list of actions
- Menu: list of choices on the screen
- Menus can be implemented using `if/else if` statements



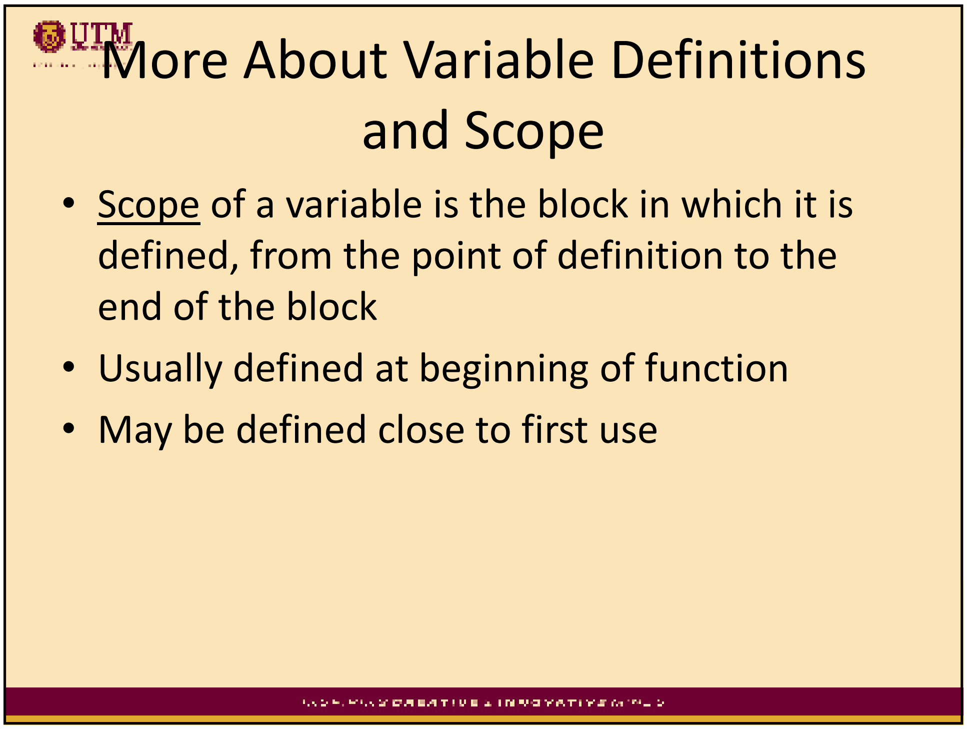
# Menu-driven program organization

- Display list of numbered or lettered choices for actions
- Prompt user to make selection
- Test user selection in *expression*
  - if a match, then execute code for action
  - if not, then go on to next *expression*



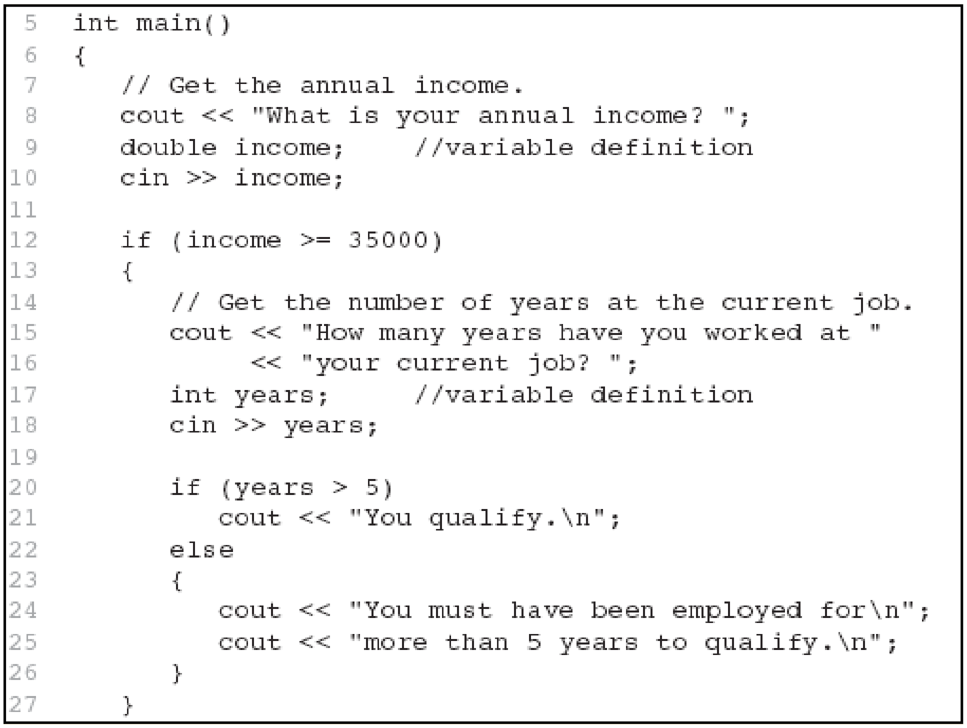
# Using `switch` with a menu

- `switch` statement is a natural choice for menu-driven program:
  - display the menu
  - then, get the user's menu selection
  - use user input as `expression` in `switch` statement
  - use menu choices as *expr* in `case` statements

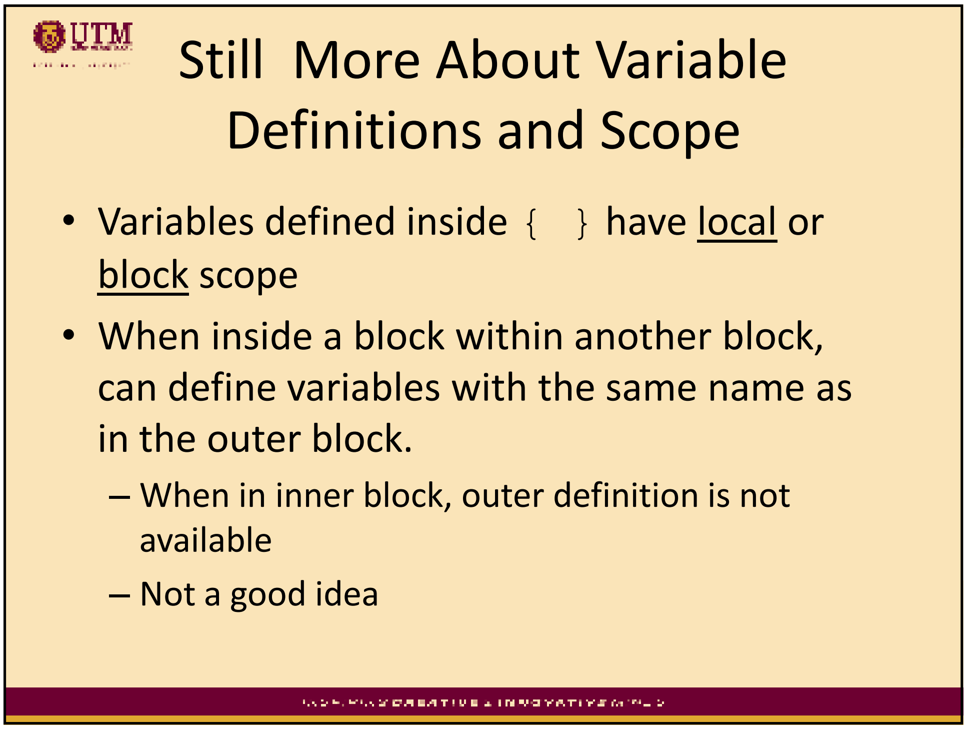


# More About Variable Definitions and Scope

- Scope of a variable is the block in which it is defined, from the point of definition to the end of the block
- Usually defined at beginning of function
- May be defined close to first use

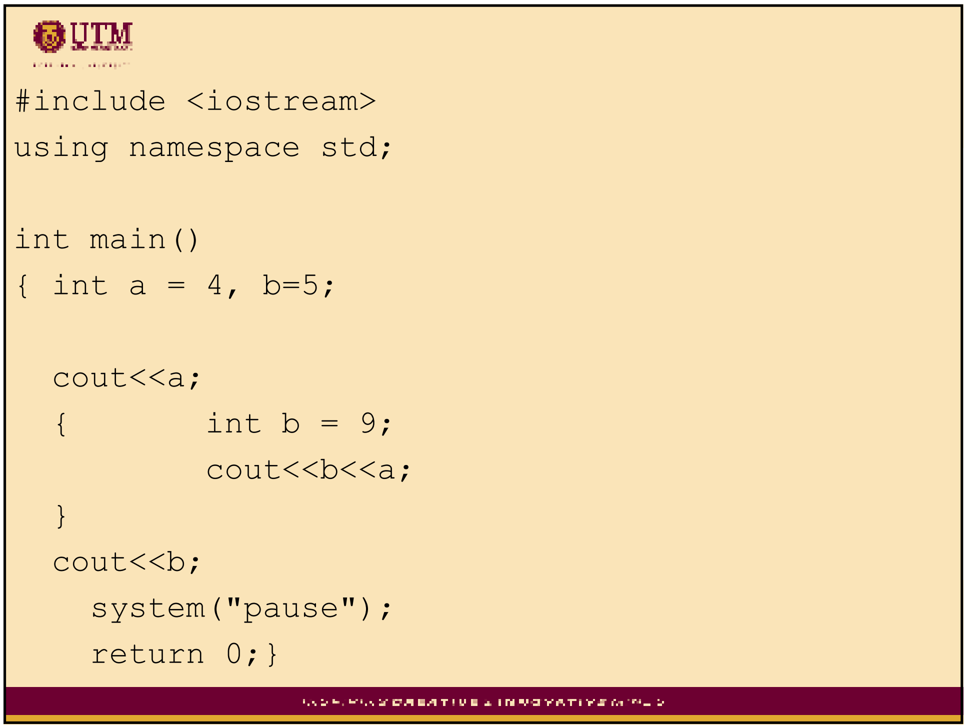


```
int main()
{
   // Get the annual income.
   cout << "What is your annual income? ";
   double income;     //variable definition
   cin >> income;

   if (income >= 35000)
   {
      // Get the number of years at the current job.
      cout << "How many years have you worked at "
           << "your current job? ";
      int years;     //variable definition
      cin >> years;

      if (years > 5)
         cout << "You qualify.\n";
      else
      {
         cout << "You must have been employed for\n";
         cout << "more than 5 years to qualify.\n";
      }
   }
```



## Still More About Variable Definitions and Scope

- Variables defined inside { } have local or block scope
- When inside a block within another block, can define variables with the same name as in the outer block.
  - When in inner block, outer definition is not available
  - Not a good idea



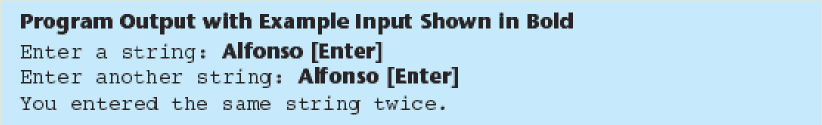
```
#include <iostream>
using namespace std;

int main()
{ int a = 4, b=5;

  cout<<a;
  {        int b = 9;
           cout<<b<<a;
  }
  cout<<b;
    system("pause");
    return 0;}
```

# In-Class Exercise

- What will the following program display if user enter `test1` 40 and `test2` 30?

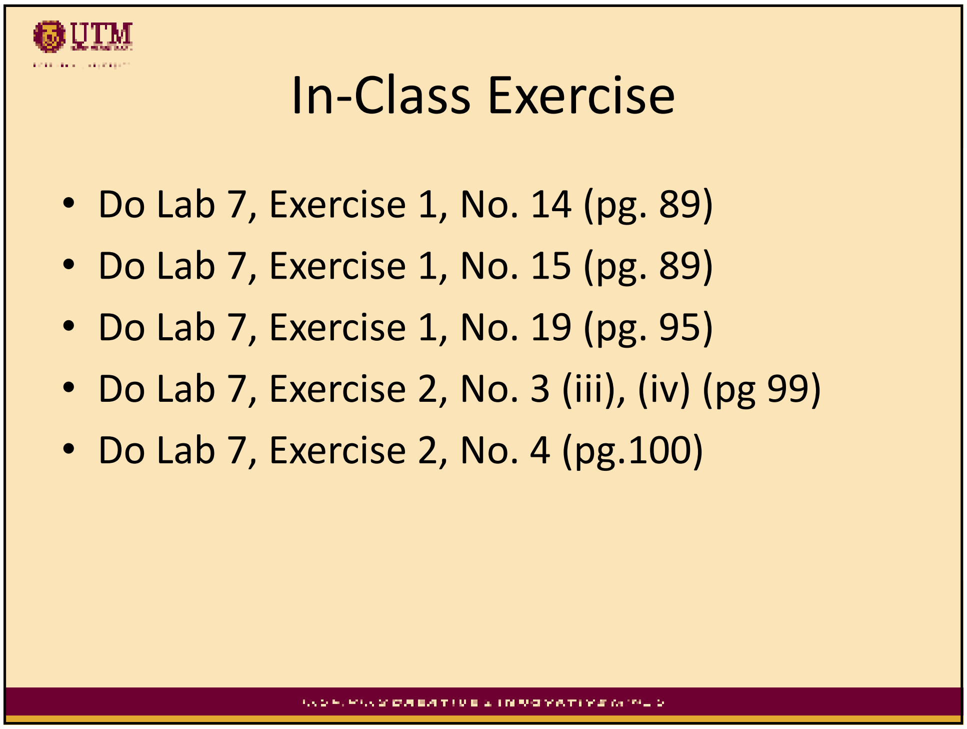
```
int main ()
{
 int test1;
 cout<<"Enter Test 1 score: ";
 cin>>test1;

 int test2;
 cout<<"Enter Test 2 score: ";
 cin>>test2;

 int sum=test1+test2;
```

```
if (sum>60)   {
 int bonus=10;
 test1+=bonus; test2+=bonus;
 int sum=test1+test2;
 cout<<"Test 1 with bonus:"
     <<test1<<endl;
 cout<<"Test 2 with bonus:"
     <<test2<<endl;
 cout<<"Sum with bonus:"
     <<sum<<endl;
}
cout<<"Test 1 : "
    <<test1<<endl;
cout<<"Test 2 : "
    <<test2<<endl;
cout<<"Sum : "<<sum<<endl;
return 0;
}
```



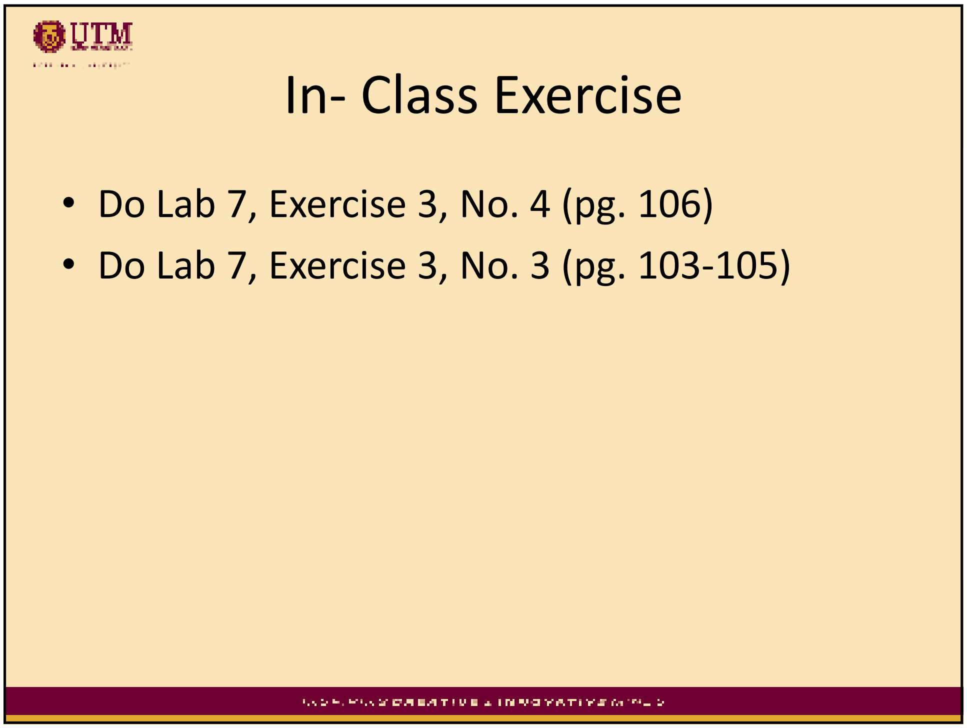
**Program Output with Example Input Shown in Bold**

```
Enter a string: Alfonso [Enter]
Enter another string: Alfonso [Enter]
You entered the same string twice.
```



# In-Class Exercise

- Do Lab 7, Exercise 1, No. 14 (pg. 89)
- Do Lab 7, Exercise 1, No. 15 (pg. 89)
- Do Lab 7, Exercise 1, No. 19 (pg. 95)
- Do Lab 7, Exercise 2, No. 3 (iii), (iv) (pg 99)
- Do Lab 7, Exercise 2, No. 4 (pg.100)



# In- Class Exercise

- Do Lab 7, Exercise 3, No. 4 (pg. 106)
- Do Lab 7, Exercise 3, No. 3 (pg. 103-105)