3.11 Boolean data type

Boolean data type

Boolean refers to a quantity that has only two possible values, true or false.The language has the built-in data type ***bool*** for representing Boolean quantities.

A Boolean variable may be set using true or false keywords. Ex: `isMale = true;` assigns isMale with the Boolean value true. A Boolean variable may also be set to the result of a logical expression. Ex: `isOverweight = (userBmi >= 25);` assigns isOverweight with the result of the expression userBmi >= 25.

Figure 3.11.1: Variables of bool data type: Life expectancy calculator.

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

int main() {
   double lifeExpectancy;
   double userBmi;
   char userChoice;
   bool isOverweight;
   bool isMale;

   // Get user's sex
   cout << "Female (f) or male (m): ";
   cin >> userChoice;
   if (userChoice == 'm') {
      isMale = true;
   }
   else {
      isMale = false;
   }

   // Get user's BMI
   cout << "Enter body mass index (BMI): ";
   cin  >> userBmi;
   isOverweight = (userBmi >= 25);

   // Determine life expectancy based on sex and BMI
   if (isMale && !isOverweight ) {
      lifeExpectancy = 79.4;
   }
   else if (!isMale && !isOverweight ) {
      lifeExpectancy = 83.5;
   }
   else if (isMale && isOverweight) {
      lifeExpectancy = 77.3;
   }
   else {
      lifeExpectancy = 81.4;
   }

   cout << "Life expectancy is " << lifeExpectancy
        << " years." << endl;

   return 0;
}
```

```
Female (f) or male (m): f
Enter body mass index (BMI): 24.1
Life expectancy is 83.5 years.

...

Female (f) or male (m): m
Enter body mass index (BMI): 28.2
Life expectancy is 77.3 years.
```

Source: Life expectancy calculator (Bankrate.com, 2017)

Feedback?

PARTICIPATION ACTIVITY | 3.11.1: Boolean variables.

1) Write a statement that declares a Boolean variable named hasHighBP.

Check Show answer

Answer

bool hasHighBP;

bool indicates the variable can hold Boolean values of true or false.

2) Write a statement that assigns hasHighBP with false.

Check Show answer

Answer

hasHighBP = false;

A bool variable can be assigned with the Boolean values of true or false.

3) isTall, isRich, and isFamous are Boolean variables. What is isFamous after executing the following statements? Type true or false.

```
isTall = false;
isRich = true;
isFamous = false;

if (isTall && isRich) {
   isFamous = true;
}
```

Check Show answer

Answer

false

Because isTall is false, the if expression is false, and thus the value of isFamous is not changed from its initial false value.

isFamous is set to true only when isTall and isRich are both true.

Feedback?

Uses of Boolean data types

A programmer can use a Boolean variable to simplify a complex expression. An expression that combines logical and relational operators can be simplified by assigning bool variables with the result of the expression using relational operators. The if-else expression can then consist of only logical operations using those variables.

The following program assigns bool variables isHot, isReallyHot, and isHumid with the results of expressions comparing currentTemp, desiredTemp, and currentHumidity. The if-else statement then uses isHot and isHumid in the if-else's expressions.

Figure 3.11.2: Using Boolean variables to simplify expressions.

```
isHot = (currentTemp > desiredTemp);
isReallyHot = (currentTemp > (desiredTemp + 5.0));
isHumid = (currentHumidity > 0.50);

if (isReallyHot) {
   // Use A/C and evaporative cooler
   acOn = true;
   evapCoolerOn = true;
}
if (isHot && isHumid) {
   // Use A/C
   acOn = true;
   evapCoolerOn = false;
}
else if (isHot && !isHumid) {
   // Use evaporative cooler
   acOn = false;
   evapCoolerOn = true;
}
else {
   acOn = false;
   evapCoolerOn = false;
}
```

Feedback?

PARTICIPATION ACTIVITY 3.11.2: Simplifying expressions.

Given the following if-else statement:

```
if ( (userAge > 13) && (userAge < 21) && studentGpa >= 3.5 ) {
   studentDiscount = 7.5;
}
else if ( (userAge > 13) && (userAge < 21) && studentGpa >= 2.75 ) {
  studentDiscount = 5.0;
}
else {
  studentDiscount = 0.0;
}
```

1) Write a statement that assigns the variable veryGoodGpa with an expression that evaluates to true if studentGpa is greater than or equal to 3.5.

```
veryGoodGpa = (
                          );
```

Check Show answer

Answer

studentGpa >= 3.5

veryGoodGpa is assigned the result of the relational operator, which is true or false.

2) Write a statement that assigns the variable goodGpa with an

Answer

expression that evaluates to true if studentGpa is greater than 2.75.

```
goodGpa = (
                       );
```

Check Show answer

studentGpa > 2.75

If the expression studentGpa >2.75 evaluates to true, goodGpa is assigned with true. Otherwise, goodGpa is assigned with false.

3) Write a statement that assigns the variable isInAgeRange with an expression that evaluates to true if userAge is greater than 13 and less than 21.

```
isInAgeRange = (

);
```

Check Show answer

Answer

(userAge > 13) && (userAge < 21)

isInAgeRange is assigned with an expression that detects a specific age range, and thus combines relational and logical operators to detect that range.

4) Revise the if expression above to use the variables isInAgeRange and veryGoodGpa.

```
if (

) {
   studentDiscount = 7.5;
}
```

Check Show answer

Feedback?

CHALLENGE ACTIVITY 3.11.1: Using bool.

Assign isTeenager with true if kidAge is 13 to 19 inclusive. Otherwise, assign isTeenager with false.

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

int main() {
   bool isTeenager;
   int kidAge;

   cin >> kidAge;

   /* Your solution goes here  */
   if(kidAge >=13 && kidAge <=19){
   isTeenager = true;
   }else{
      isTeenager = false;
   }
   if (isTeenager) {
      cout << "Teen" << endl;
   }
   else {
      cout << "Not teen" << endl;
```

Run ✓ All tests passed

✓ Testing value of isTeenager for kidAge = 13

Your output
```
Teen
```

✓ Testing value of isTeenager for kidAge = 1

Your output
```
Not teen
```

✓ Testing value of isTeenager for kidAge = 19

Your output
```
Teen
```

✓ Testing value of isTeenager for kidAge = 24

Your output
```
Not teen
```

✓ Testing value of isTeenager for kidAge = -4

Your output
```
Not teen
```

Feedback?

CHALLENGE ACTIVITY | 3.11.2: Bool in branching statements. ✓

Write an if-else statement to describe an object. Print "Balloon" if isBalloon is true and isRed is false. Print "Red balloon" if isBalloon and isRed are both true. Print "Not a balloon" otherwise. End with newline. (Notes)

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

int main() {
   bool isRed;
   bool isBalloon;

   cin >> isRed;
   cin >> isBalloon;

   /* Your solution goes here  */
   if (isBalloon&& !isRed){
      cout<<  "Balloon" << endl;
   } else if (isBalloon&& isRed){
      cout<<  "Red balloon" << endl;
   } else{
   cout<<  "Not a balloon" << endl;
   }

   return 0;
}
```

Run ✓ All tests passed

✓ Testing isBalloon = false, isRed = false

Your output
```
Not a balloon
```

✓ Testing isBalloon = true, isRed = false

Your output
```
Balloon
```

✓ Testing isBalloon = false, isRed = true

Your output
```
Not a balloon
```

✓ Testing isBalloon = true, isRed = true

Your output
```
Red balloon
```

Feedback?