3.2 The bool Data Type

Wednesday, January 25, 2023 9:12 AM

- True or false
- compare values
Operator	Mathematics Symbol	Name	Example (radius is 5)	Result
<	<	less than	`radius < 0`	`false`
<=	≤	less than or equal to	`radius <= 0`	`false`
>	>	greater than	`radius > 0`	`true`
>=	≥	greater than or equal to	`radius >= 0`	`true`
==	=	equal to	`radius == 0`	`false`
!=	≠	not equal to	`radius != 0`	`true`
- Result is boolean value, true/false
- They are literals, keywords, cannot be used as identifiers in ur pgrm
- C++ uses 1 for true, 0 for false

3.3 if Statements

Wednesday, January 25, 2023 9:22 AM

- Alternative paths
  - One way if statements
  - Two war if else statements
  - Nested if statements
  - Switch statements
  - Conditional expression
- 1 way if
  - "if and only if condition is true"
  - Written as:
    - if (radius >= 0){
    - area = r * r * PI;
    - cout<< "Area is" <<area;
    - }
  - Boolean expressions are in the parentheses

3.4 Two Way if-else Statements

Wednesday, January 25, 2023 2:13 PM

- One-way if does action only if condition is true
- 2-way if-else statement does false as well
- Ex:
  - if(radius>=0) {
  - area=radius*radius*PI;
  - cout<<"The area for the circle of radius"<< radius<<"is"<<area; }
  - else {
  - cout<<"Negative radius";
  - }
- Same as with if, if function inside if statement is 1 line, don't have to use brackets

3.5 Nested if & Multi-Way if-else Statements

Wednesday, January 25, 2023 2:30 PM

- Statement in if-else statement can be anything, like other if-else
- Yeah, just do it

3.6 Common Errors and Pitfalls

Wednesday, January 25, 2023 2:57 PM

1. Forgetting Necessary braces
    a. If statement in if-else, if, or else statement is longer than 1 line, needs braces
2. Wrong semicolon at if line
    a. Don't put semicolon on if statement
3. = instead of ==
    a. = is assignment, == is comparator
4. Redundant testing of Boolean values
    a. If need to check if the boolean is true, just add that var to the condition w/out ==true
    b. So Not:
        i. if (even == true)
        ii. cout << "It is even.";
    c. This is better:
        i. if (even)
        ii. cout << "It is even.";
5. Dangling else Ambiguity
    a. Make sure to use braces, otherwise nested if statements could have the wrong else assigned to them
6. Equality Test of 2 floating pt vals
    a. Floating pts do round off errors, so don't use them as == comparators, be broad
    b. Can use keyword EPSILON for very small values
7. Simplifying Boolean Var Assignment
    a. I didn't know this, but not error, don't write this:
        i. if (number % 2 == 0)
        ii. even = true;
        iii. else
        iv. even = false;
    b. Write this:
        i. bool even = number % 2 == 0;
8. Don't duplicate code in diff cases
    a. Just combo them out in one place
    b. Like outside a for loop, instead of in the for loop

3.7 Case Study: Computing Body Mass Index

Friday, January 27, 2023 9:38 AM

BMI	Interpretation
BMI < 18.5	Underweight
18.5 ≤ BMI < 25.0	Normal
25.0 ≤ BMI < 30.0	Overweight
30.0 ≤ BMI	Obese
- Pgrm that gets user's weight in lbs, height in inches, then displays BMI
- 1 lbs = .45359237 kg
- 1 in = .0254 meters
- ```
  const double KILOGRAMS_PER_POUND = 0.45359237; // Constant
  const double METERS_PER_INCH = 0.0254; // Constant

  // Compute BMI
  double weightInKilograms = weight * KILOGRAMS_PER_POUND;
  double heightInMeters = height * METERS_PER_INCH;
  double bmi = weightInKilograms /
    (heightInMeters * heightInMeters);
  ```
  ```

3.8 Case Study: Computing Taxes

Friday, January 27, 2023 9:44 AM

- Based on filing status & taxable income
- 4 filing statuses
  - Single filers
  - Married filing jointly/qualified widow(er)
  - Married filing separately
  - Head of household
-

Marginal Tax Rate	Single	Jointly or Qualifying Widow(er)	Married Filing Separately	Head of Household
10%	$0 – $8,350	$0 – $16,700	$0 – $8,350	$0 – $11,950
15%	$8,351– $33,950	$16,701 – $67,900	$8,351 – $33,950	$11,951 – $45,500
25%	$33,951 – $82,250	$67,901 – $137,050	$33,951 – $68,525	$45,501 – $117,450
28%	$82,251 – $171,550	$137,051 – $208,850	$68,526 – $104,425	$117,451 – $190,200
33%	$171,551 – $372,950	$208,851 – $372,950	$104,426 – $186,475	$190,201 – $372,950
35%	$372,951+	$372,951+	$186,476+	$372,951+

- Pgrm compute personal income tax
- Filing status
  - 0 = single filers
  - 1 = married/widow(er)
  - 2 = married filing separately
  - 3 = head of household

3.9 Generating Random Numbers

Friday, January 27, 2023 9:51 AM

- rand() in cstdlib header file
- This returns random integer btwn 0 and RAND_MAX (platform dependant constant)
- rand() makes numbers that are pseudorandom, it makes same sequence of numbers
- This bc seed value used to ctrl it, so if change seed val => change rand numbs
- Change seed val by srand(seed) function in cstdlib header file
- To Make sure seed val diff each time ran, use time(0) which puts in current time (always changing)
- For random int btwn 0 and 9, use: rand() % 10;
- #include <ctime>
- #include <cstdlib>
- Using namespace std;

3.10 Logical Operators

Friday, January 27, 2023 11:32 AM

- Combo of many conditions to do statement, use logical ops (aka bool ops), operate on bool values & make bool vals
- ! Negates the thing, T-> F, F-> T
- && of 2 bool operands, T if (and only if) both operands are true
- || is or, T if one (or both) of the operands is T
-

Operator	Name	Description
!	not	logical negation
&&	and	logical conjunction
||	or	logical disjunction

-

p	!p	Example (assume age = 24, weight = 140)
true	false	!(age > 18) is false, because (age > 18) is true.
false	true	!(weight == 150) is true, because (weight 150) is false.

-

p	p2	p && p2	Example (assume age = 24, weight = 140)
false	false	false	(age > 28) && (weight < 140) is false, because
false	true	false	(age > 28) and (weight < 140) are both false.
true	false	false	(age > 18) && (weight >= 140) is true, because
true	true	true	both (age > 18) and (weight >= 140) are true.

-

-

p	p2	p || p2	Example (assume age = 24, weight = 140)
false	false	false	(age > 34) || (weight < 140) is false, because both (age > 34) and (weight < 140) are false.
false	true	true	
true	false	true	(age > 18) || (weight >= 150) is true, because (age > 18) is true.
true	true	true	

- Don't do dumb
  - 28 <= x <= 31 wrong
  - (28 <= x) && (x <= 31) correct
- De Morgan's Law- simplifies bool expressions
  - !(x && y) = !x || !y
  - !(x||y) = !x && !y
- && and || called lazy operators bc don't always check both terms
- Bitwise AND (&) and OR (|) operators
- True = 1, false = 0
- Can assign bool val to be used as int
- Ex:
  - Incorrect:
  - `if (!amount <= 50)`
  - `  cout << "Amount is more than 50";`
  - **(a) (!amount <= 50) is incorrect**
  -
  - Correct code:
  - `if (!(amount <= 50))`
  -
    `  cout << "Amount is more than 50";`
  - **(b) !(amount <= 50) is correct**
- bruh

3.11 Case Study: Determining Leap Year

Friday, January 27, 2023 1:36 PM

- Leap year if it's divisible by 4 but not 100, or if it's divisible by 400
- Leap year has 366 days, feb in leap year has 29 days
-

```
// A leap year is divisible by 4
bool isLeapYear = (year % 4 == 0);

// A leap year is divisible by 4 but not by 100
isLeapYear = isLeapYear && (year % 100 != 0);

// A leap year is divisible by 4 but not by 100 or divisible by 400
isLeapYear = isLeapYear || (year % 400 == 0);
```

-
- or you can combine all these expressions into one:
-

```
isLeapYear = (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
```

3.12 Case Study: Lottery

Friday, January 27, 2023 6:49 PM

- Pgrm plays lottery, random make 2 digit numb, get's user enter 2 numbs, find out if user won according to rule:
  - If user input matches lottery number in exact order, win $10,000
  - If all digits in input match lottery numb, award is $3,000
  - If 1 digit in input matches digit in lottery numb, win $1,000
- Digit of 2 digits can be 0, less than 10 means starts w/ 0 (like 09 is nine)
- Get first digit from user by guess/10 then get second digit by guess%10
- Get random number [1-99] by rand() % 100

3.13 switch Statements

Friday, January 27, 2023 7:47 PM

- Don't do bunch of if-else statements, use this
- switch (status) {
  - case 0: words;
    - break
  - case 1: words;
    - break
  - default: words;
- }
- Takes in the status, check if it's the right case, if it is, does it & then breaks (goes to end, doesn't run anything in the switch), if does none of them, then does default
- Can leave off break if want to do fall thru- last case that has break has condition that all others need
-

```
switch (day)
{
  case 1: // Fall to through to the next case
  case 2: // Fall to through to the next case
  case 3: // Fall to through to the next case
  case 4: // Fall to through to the next case
  case 5: cout << "Weekday"; break;
  case 0: // Fall to through to the next case
  case 6: cout << "Weekend";
```

3.14 Conditional Operators

Friday, January 27, 2023 8:12 PM

- Assign val to vars to restrict conditions, or can do conditional operator
- If(x>0) y = 1; else y = -1;
- y= x > 0 ? 1 : -1;
- ? and : make conditional operators (aka ternary operator)
- boolean-expression ? expression1 : expression2;
- If boolean-expression is true, does expression 1, else does expression 2
- ```
  cout << (num % 2 == 0 ? "num is even" : "num is odd") << endl;
  ```
-
  ```

3.15 Operator Precedence & Associativity

Friday, January 27, 2023 9:14 PM

- **Operator Precedence Chart (precedence from high to low)**

  var++ and **var--** (Postfix)

  +, **-** (Unary plus and minus), **++var**, and **--var** (Prefix)

  `static_cast<type>(v)`, `(type)v` (Casting)

  ! (Not)

  *****, **/**, **%** (Multiplication, division, and remainder)

  +, **-** (Binary addition and subtraction)

  <, **<=**, **>**, **>=** (Relational)

  ==, **!=** (Equality)

  && (AND)

  || (OR)

  ?: (Ternary conditional operator)

  =, **+=**, **-=**, ***=**, **/=**, **%=** (Assignment and augumented operator)

- PEMDAS- on roids

3.16 Debugging

Friday, January 27, 2023 9:20 PM

- Logic errors are bugs
- Execute single statement at a time
- Tracing into/ stepping over a function
- Setting breakpoints
- Displaying values in variables
- Displaying call stacks
- Modifying variable
-

Chapter Summary

Friday, January 27, 2023 9:25 PM

1. A bool type variable can store a true or false value.
2. Internally, C++ uses 1 to represent true and 0 for false.
3. If you display a bool value to the console, 1 is displayed if the value is true and 0 if the value is false.
4. In C++, you can assign a numeric value to a bool variable. Any nonzero value evaluates to true and zero value evaluates to false.
5. The relational operators (<, <=, ==, !=, >, >=) yield a Boolean value.
6. The equality testing operator is two equal signs (==), not a single equal sign (=). The latter symbol is for assignment.
7. Selection statements are used for programming with alternative courses of actions. There are several types of selection statements: if statements, two-way if-else statements, nested if statements, multi-way if-else statements, switch statements, and conditional expressions.
8. The various if statements all make control decisions based on a Boolean expression. Based on the true or false evaluation of the expression, these statements take one of two possible courses.
9. The Boolean operators &&, ||, and ! operate with Boolean values and variables.
10. When evaluating p1 && p2, C++ first evaluates p1 and then evaluates p2 if p1 is true; if p1 is false, it does not evaluate p2. When evaluating p1 || p2, C++ first evaluates p1 and then evaluates p2 if p1 is false; if p1 is true, it does not evaluate p2. Therefore, && is referred to as the short-circuit AND operator, and || is referred to as the short-circuit OR operator.
11. The switch statement makes control decisions based on a switch expression.
12. The keyword break is optional in a switch statement, but it is normally used at the end of each case in order to skip the remainder of the switch statement. If the break statement is not present, the next case statement will be executed.
13. The conditional operators can be used to simplify coding.
14. The operators in expressions are evaluated in the order determined by the rules of parentheses, operator precedence, and operator associativity.
15. Parentheses can be used to force the order of evaluation to occur in any sequence.
16. Operators with higher precedence are evaluated earlier. For operators of the same precedence, their associativity determines the order of evaluation.
17. All binary operators except assignment operators are left-associative; assignment operators are right-associative.