2.2 Writing a Simple Program

Monday, January 23, 2023 9:49 AM

- Problem: area of circle
- First make algorithm- describes how problem solved by listing action must be taken & order of execution, they can help plan b4 writing, usually written in pseudocode
  - Algorithm for area of circle
  1. Read in circle's radius
  2. Compute area using formula: area = r * r * pi
  3. Display result
- 2 new questions:
  1. Reading the radius
  2. Storing the radius – use variable, choose descriptive name, also specify data type aka declaring a variable
     i. Primitive types: integer, floating-point, characters, and Boolean
  - We use double for this:
    - double radius;
    - double area;
- We'll read the radius later, but for now, plug into formula
  - area= radius * radius * 3.1415926;
  - And then print it out:
  - cout<< "The area is "<< area << endl;
  - return 0;
- Each var has mem location, can cause errors if delete where intialized

2.3 Reading Input from Keyboard

Monday, January 23, 2023 11:55 AM

- The cin object reads input from keyboard, assign it to a variable
    - Ex:
    - double radius;
    - cout<< "Enter a radius: ";
    - cin>> radius;
- Usually u gotta prompt the user to enter something
- cin is console input, waits till data is entered and Enter key is pressed
- >> is stream extraction operator, said as "get from"
- Following it is variable
- cin can be used to read in many vals
    - cin>> x1>>x2>>x3;
-

2.4 Identifiers

Monday, January 23, 2023 12:11 PM

- Identifiers are:
    - String w/ letter, digits, and _ (underscores)
    - Start w/ letter/_, not w/ digit
    - Cannot be a reserved word
    - Any length, can have restrictions
- All keywords in C++ are lowercase

2.5 Variables

Monday, January 23, 2023 12:16 PM

- They store values, can be changed
- They represent data of certain type
- To use var, declare it by telling name and type
- Varibale declaration tells compiler to keep mem space for var based on data type
  - Synatax is:
  - datay
- If var of same type, can be declared together
  - Datatype var1, var2, var3;
- Usually they have initial vals, and they can also be dun together
  - int I =1, j = 2;
- Scope of variable is where var can be referenced

2.6 Assignment Statements and Assignment Expressions

Monday, January 23, 2023 12:32 PM

- = is assignment operator
  - Variable = expression;
- Expression reps computation w/ values, vars, and operator
- Can use variables itself inside function, uses old val to update to new val
  - x= x+1;
- Assignment expression, and chained assignment
  - cout << x=1;   is the same as:
  - x=1;
  - cout << x;
  - Also, chained:
  - i= j = k = 1;
  - Can't do:
  - int i = j = k =1;
  - bc j & k aren’t declared

2.7 Named Constants

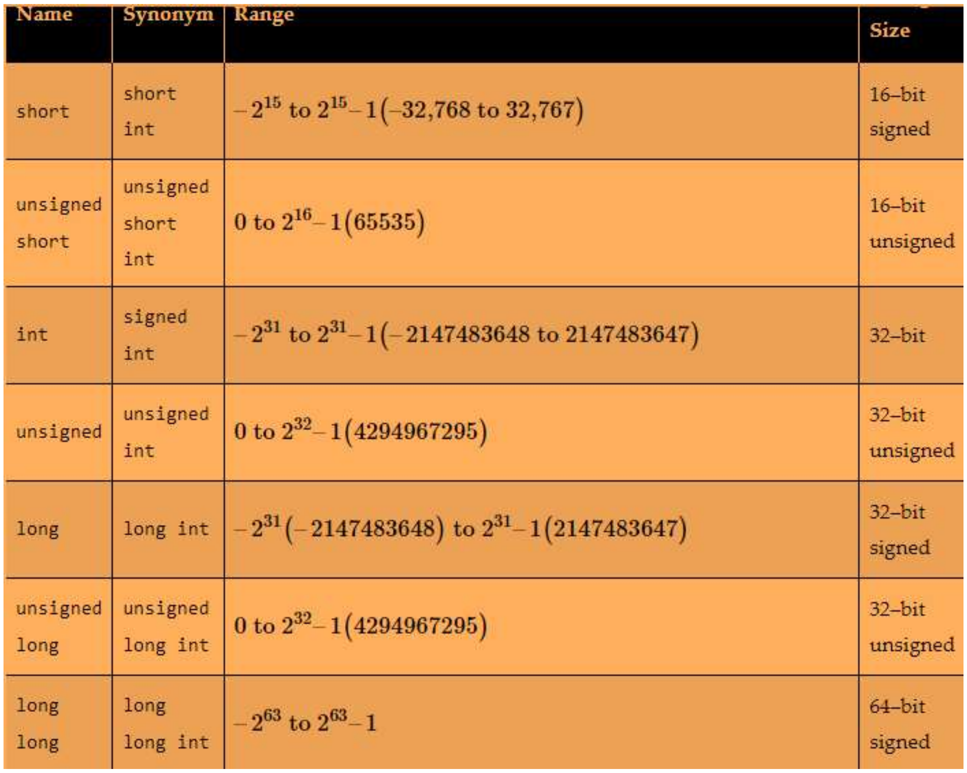
Monday, January 23, 2023 12:42 PM

- It's permanent data, not changed
- Syntax:
    - const datatype CONSTANTNAME = value;
- Constant must be declared and initialized @ same statement
- By convention, constants are named in uppercase
    - Ex:
    - const double PI = 3.1415926;

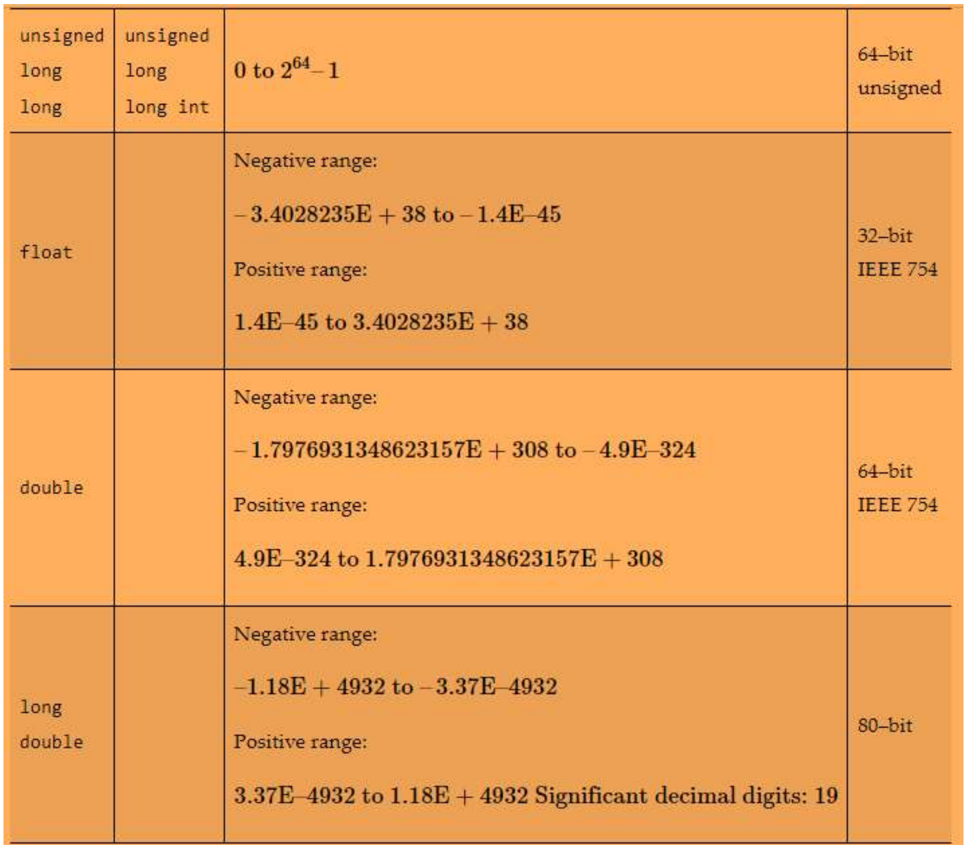
2.8 Numeric Data Types & Operations

Monday, January 23, 2023 12:50 PM

- Data types have range of values, space in mem
  -



Name	Synonym	Range	Size
short	short int	-2^{15} to $2^{15}-1(-32{,}768$ to $32{,}767)$	16–bit signed
unsigned short	unsigned short int	0 to $2^{16}-1(65535)$	16–bit unsigned
int	signed int	-2^{31} to $2^{31}-1(-2147483648$ to $2147483647)$	32–bit
unsigned	unsigned int	0 to $2^{32}-1(4294967295)$	32–bit unsigned
long	long int	$-2^{31}(-2147483648)$ to $2^{31}-1(2147483647)$	32–bit signed
unsigned long	unsigned long int	0 to $2^{32}-1(4294967295)$	32–bit unsigned
long long	long long int	-2^{63} to $2^{63}-1$	64–bit signed



unsigned long long	unsigned long long int	0 to $2^{64}-1$	64–bit unsigned
float		Negative range: $-3.4028235\text{E}+38$ to $-1.4\text{E}{-}45$ Positive range: $1.4\text{E}{-}45$ to $3.4028235\text{E}+38$	32–bit IEEE 754
double		Negative range: $-1.7976931348623157\text{E}+308$ to $-4.9\text{E}{-}324$ Positive range: $4.9\text{E}{-}324$ to $1.7976931348623157\text{E}+308$	64–bit IEEE 754
long double		Negative range: $-1.18\text{E}+4932$ to $-3.37\text{E}{-}4932$ Positive range: $3.37\text{E}{-}4932$ to $1.18\text{E}+4932$ Significant decimal digits: 19	80–bit

- Half of numbs repped by signed int are neg, other are pos
- All numbs repped in unsigned int are pos
  - This means that u can store 2x big largest positive int in unsigned than in signed
- 3 types of integers: short, int and long, can be either signed or unsigned
- 3 types of floating-point numbers: float, double, and long double (2x float)
- Can use sizeof(datatype) to find the bytes stored for that datatype
  - Ex: sizeof(int)

2.8.1 Numeric Literals

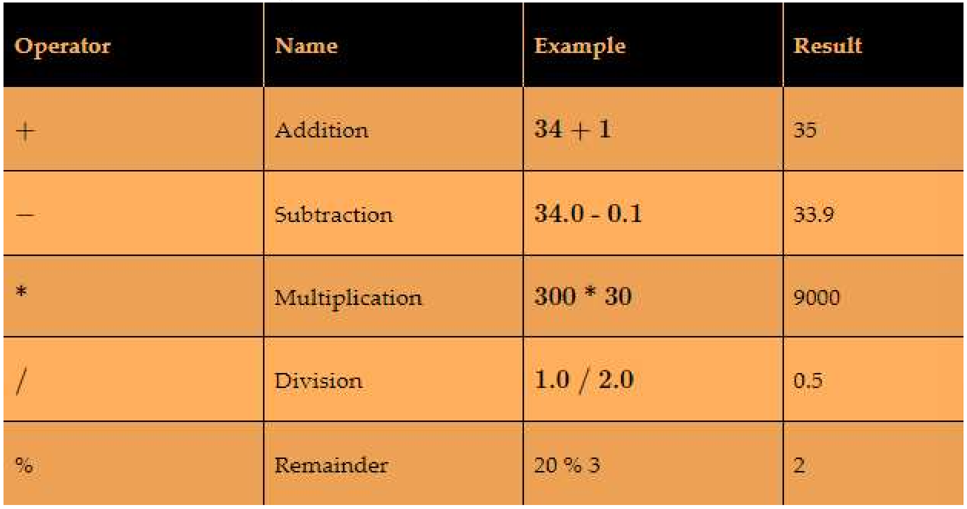
Monday, January 23, 2023 2:12 PM

- Literal is constant val, direct in pgrm
- By default, integer literal is deciaml integer numb
  - for binary integer literal, use leading 0b or 0B
  - For octal integer literal, use leading 0
  - For hexadecimal integer literal, use leading 0x or 0X
- Floating point literals write in scientific notation, form x10^{b} , E (or e) reps exponent
  - Ex:
  - 1.23456 x 10^{2}
  - 1.23456E2 or 1.23456E+2
- Can use single quotes ' and underscore _ as digit separators btwn 2 digits in number literal
  - int amount = 2'245'451;
  - Int amount = 2_245_451;
  - Which is the same as 2,245,451 that we normally read

2.8.2 Numeric Operators

Monday, January 23, 2023 2:31 PM

- Operators for numeric data types, main operators: +, -, *, /, %(remainder)



Operator	Name	Example	Result
+	Addition	34 + 1	35
–	Subtraction	34.0 - 0.1	33.9
*	Multiplication	300 * 30	9000
/	Division	1.0 / 2.0	0.5
%	Remainder	20 % 3	2

- Div of ints only give int
- % only works with integer operands

2.8.3 Exponent Operations

Monday, January 23, 2023 2:40 PM

- Function of pow(a,b) does a^b , its in the cmath library
  - #include <cmath>
- Sometimes it required a &/or b be decimal val
  - 2.0 instead of 2

2.9 Evaluating Expressions & Operator Precedence

Monday, January 23, 2023 3:00 PM

- Translate math expressions to code
- Lots of parentheses, nested
- Pemdas, also be careful w/ fractions, 5/9 = 0 , 5.0/9 = .55556

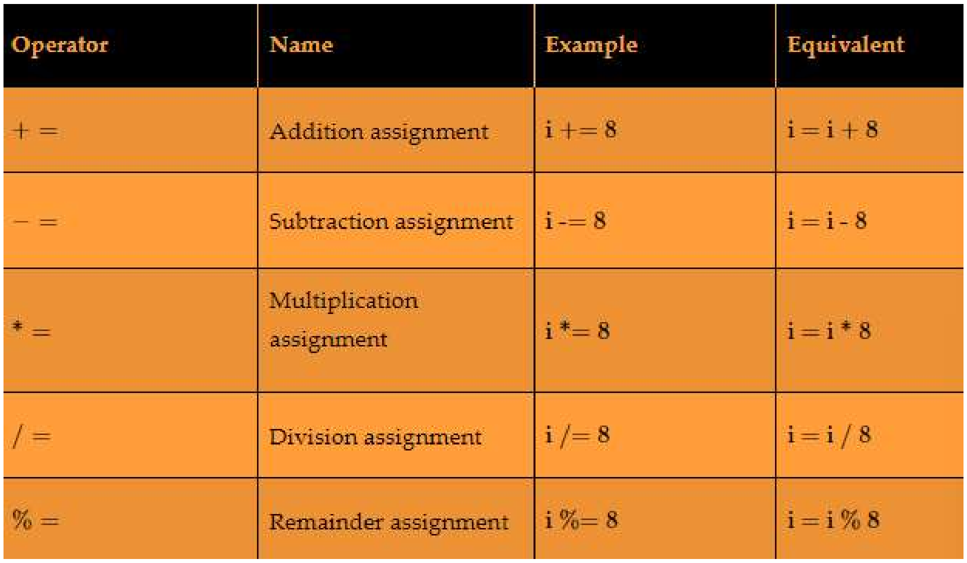
2.10 Case Study: Displaying the Current Time

Monday, January 23, 2023 5:46 PM

- Current time in GMT (Greenwich Mean Time), format of hour:minute:second
- Now include ctime
  - #include <ctime>
- totalSeconds since january 1, 1970 (when it all began) to now using time(0)
- Current second from totalSeconds % 60
- totalMinutes by totalSecods/60
- Current minuite from totalMinutes % 60
- totalHours by totalMinutes/60
- Current hour from totalHours % 24

2.11 Augmented Assignment Operators

Monday, January 23, 2023 5:55 PM

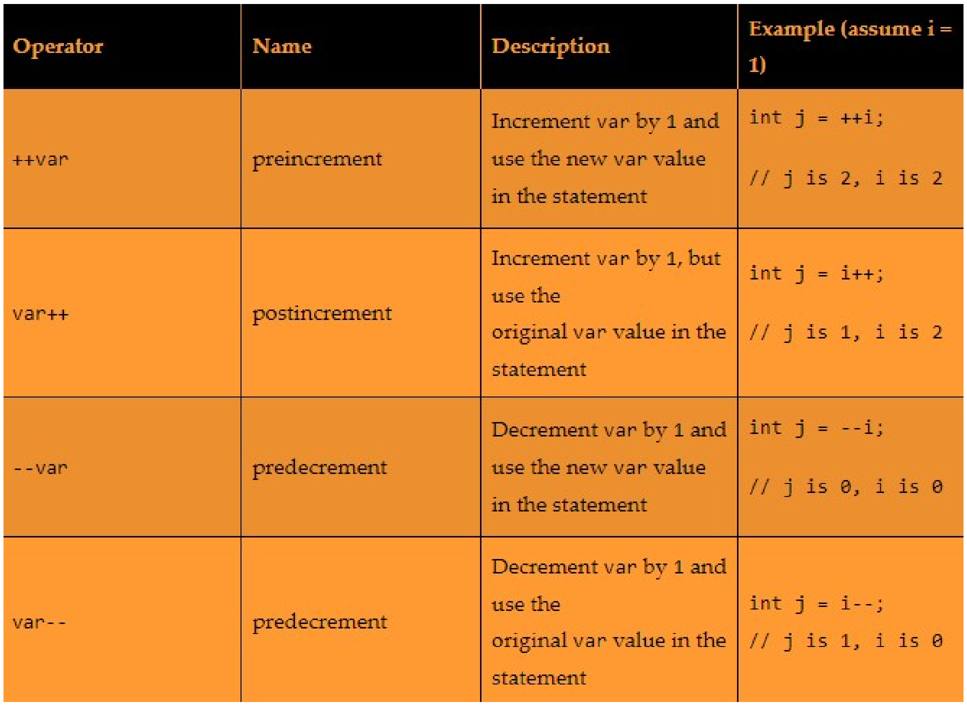
- Usually modify var, and set it equal to itself
  - count = count + 1;
- Can also combo them
  - count += 1;
- They called augmented assignment operator
  -
  - 

Operator	Name	Example	Equivalent
+ =	Addition assignment	i += 8	i = i + 8
− =	Subtraction assignment	i -= 8	i = i - 8
* =	Multiplication assignment	i *= 8	i = i * 8
/ =	Division assignment	i /= 8	i = i / 8
% =	Remainder assignment	i %= 8	i = i % 8

  - The augmented assignment is dun last in expression, like:
  - x/= 4 + 5.5 * 1.5;
  - Is the same as:
  - x= x / (4 + 5.5 * 1.5);

2.12 Increment and Decrement Operators

Monday, January 23, 2023 6:06 PM

- ++ (increment) and -- (decrement) both by 1.
- If i++ (or i--), it's postincrement (postdecrement), and they change the val after the function
    - Easier to see here:
    - These two have the same effect
        - int i = 10;
        - int newNum = 10 * i++;
    - &
        - int newNum = 10 * i;
        - i= i + 1;
    - Both result in newNum = 100, i = 11.
- If ++i (or --i), it's preincrement (predecrement), and they change the val b4 the function
    - Easier to see here:
    - These two have the same effect
        - int i = 10;
        - int newNum = 10 * ++i;
    - &
        - i= i + 1;
        - int newNum = 10 * i;
    - Both result in newNum = 110, i = 11.
-
- 

Operator	Name	Description	Example (assume i = 1)
++var	preincrement	Increment var by 1 and use the new var value in the statement	int j = ++i; // j is 2, i is 2
var++	postincrement	Increment var by 1, but use the original var value in the statement	int j = i++; // j is 1, i is 2
--var	predecrement	Decrement var by 1 and use the new var value in the statement	int j = --i; // j is 0, i is 0
var--	predecrement	Decrement var by 1 and use the original var value in the statement	int j = i--; // j is 1, i is 0

- Don't write code that depends on operand evaluation order
    - ++I + I
    - If I is 1, the evaluation could end as 4 (2 + 2) or 3 (2 + 1)
- Ex from questions:
    - `int i = 1;`

- `int j = ++i;`

- `cout << "i is " << i;`

- `cout << " and j is " << j;`
- `What is i & j?`
- `A: i is 2, j is 2`

2.13 Numeric Type Conversions

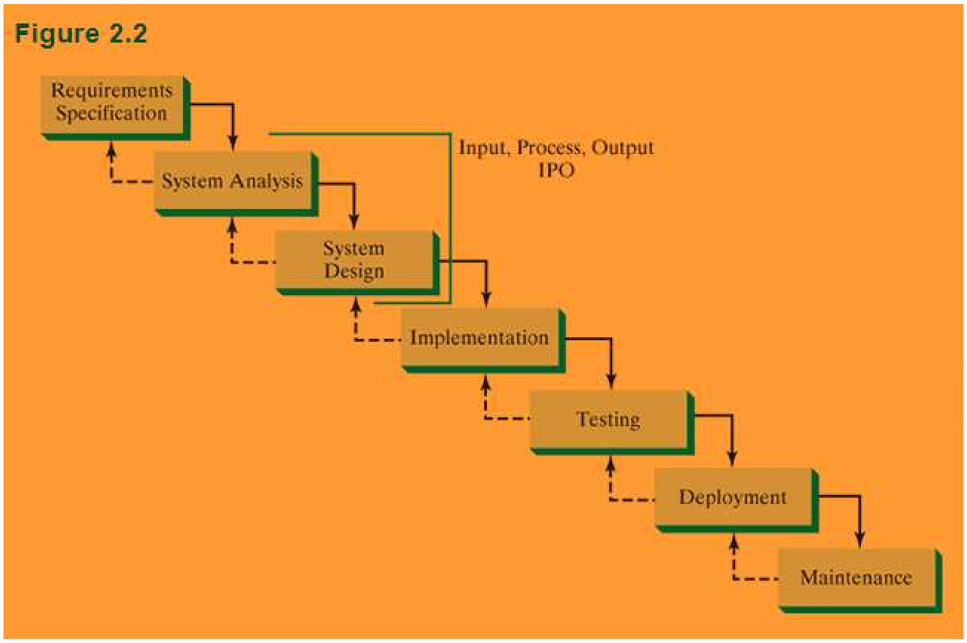
Monday, January 23, 2023 6:23 PM

- When converting stuff, probably chop stuff off (round down) when converting to more limited type, but stay same when converting to less limited type
- Operations w. diff types is good, converts to less limited one
- Can also cast from one type to another, casting operator
    - Syntax is static_cast<type>(value);
    - Value can be a var, literal, or expression
    - Type is what you want to convert value to
- When going from var of type w/ small range to var of type w/ larger range, it called widening a type
- Opp is narrowing a type, can lose precision, can lead to inaccurate results

2.14 Software Development Process

Monday, January 23, 2023 6:41 PM

- Life cycle of product design
  -
  - Figure 2.2

- Req specification is process to understand problem software will fix
  - Close talk btw devs and users
- Sys analysis- analyze data flow, id sys input & output
- Sys design- getting output from input
  - Many levels, breakdown into manageable parts
  - Subsystems, input process output (IPO)
- Implementation – translate sys design to pgrms
  - Sep pgrms for each pt, work together
  - Includes coding, self-testing, and debugging
- Testing- code meets reqs
- Deployment- software available for use
- Maintenance- update & improve product, evolving environment, bug fixes

2.15 Case Study: Counting Monetary Units

Tuesday, January 24, 2023 9:35 PM

- Pgrm for smaller monetary units, double for dollar and cents
- Steps:
    1. Prompt user to enter amount as decimal
    2. Convert to cents by *100
    3. Divide cents by 100, find dollars, remainder cents
    4. Remaining cents divide by 25 to find quarters
    5. Remaining cents divide by 10 to find dimes
    6. Remaining cents divide by 5 to find nickels
    7. Left are pennies
    8. Display
- Keep changing remainingAmount after each function
- 10.03 won't work bc 10.03 * 100 = 1002.9999999999 bc double to int loses precision, so int val now is 10 dollars and 2 cents

2.16 Common Errors

Tuesday, January 24, 2023 9:48 PM

1. Undeclared/uninitialized variables and unused variables
    a. Watch how u type stuff
2. Integer Overflow
    a. Size of types, if too large, causes overflow
    b. Also, when float is stored too close to 0, it can cause underflow
3. Round Off errors
    a. Aka rounding error
    b. Floating pt numbs are approximated
4. Unintended Integer Division
    a. /, 2 ints doing / will give int
    b. To force to make float, one of ints have to be float
5. Forgetting Header Files
    a. cmath
    b. ctime
    c. include <iostream>

Ch summary

Tuesday, January 24, 2023 9:56 PM

1. The cin object along with the stream extraction operator (>>) can be used to read an input from the console.
2. Identifiers are names for naming elements in a program. An identifier is a string that consists of letters, digits, and underscores (_). An identifier must start with a letter or an underscore. It cannot start with a digit. An identifier cannot be a reserved word.
3. Choosing descriptive identifiers can make programs easy to read.
4. Declaring a variable tells the compiler what type of data a variable can hold.
5. In C++, the equal sign (=) is used as the assignment operator.
6. A variable declared in a function must be assigned a value. Otherwise, the variable is called uninitialized and its value is unpredictable.
7. A named constant or simply constant represents permanent data that never changes.
8. A named constant is declared by using the keyword const.
9. By convention, constants are named in uppercase.
10. C++ provides integer types (short, int, long, unsigned short, unsigned int, and unsigned long) that represent signed and unsigned integers of various sizes.
11. Unsigned integers are nonnegative integers.
12. C++ provides floating-point types (float, double, and long double) that represent floating-point numbers of various precisions.
13. C++ provides operators that perform numeric operations: + (addition), – (subtraction), * (multiplication), / (division), and % (modulus).
14. Integer arithmetic (/) yields an integer result.
15. In C++, the % operator is for integers only.
16. The numeric operators in a C++ expression are applied the same way as in an arithmetic expression.
17. The increment operator (++) and the decrement operator (—) increment or decrement a variable by 1.
18. C++ provides augmented operators += (addition assignment), –= (subtraction assignment), *= (multiplication assignment), /= (division assignment), and %= (modulus assignment).
19. When evaluating an expression with values of mixed types, C++ automatically casts the operands to appropriate types.
20. You can explicitly cast a value from one type to the other using the <static_cast> (type) notation or the legacy c-style (type) notation.
21. In computer science, midnight of January 1, 1970 is known as the UNIX epoch.

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

int main(){
   double x1,y1,x2,y2,x3,y3,area;
   //I'm lazy, here's a better equation:
   //|(x1(y2-y3))+(x2(y3-y1))+(x3(y1-y2))|/2
   cout<<"Enter three points for a triangle: ";
   cin>>x1>>y1>>x2>>y2>>x3>>y3;
   area=abs((x1 * (y2 - y3)) + (x2 * (y3 - y1)) + (x3 * (y1 - y2))) /2;
```

- cout<<endl<<"The area of the triangle is "<<area;
- return 0;
- }