BASIC PROGRAMMING LANGUAGE

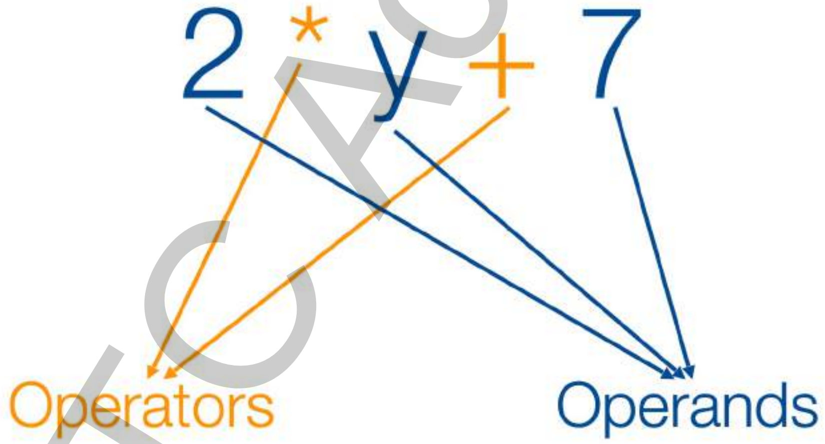
LESSON 3

Expressions, Operators and Type Casting

CONTENT

What is an Expression?

- An expression is a sequence of operators and their operands, that specifies a computation. Mt biu thc là mt chui các toán t và các toán hng ca chúng, xác nh phép tính
- An operation is performed on a data item which is called an operand. An operator indicates an operation to be performed on data.



Type of Expressions

- Primary expressions: It is an operand which can be a name, a constant or any parenthesized expression.
  - Example: `c = a+ (5*b)`
- Postfix expressions: In a postfix expression, the operator will be after the operand.
  - Example: `ab+`
- Prefix expressions: n a prefix expression, the operator is before the operand.
  - Example: `+ab`

Type of Expressions

- Unary expression: It contains one operator and one operand.
  - Example: `a++, --b`
- Binary expression: It contains two operands and one operator.
  - Example: `a+b, c-d`
- Ternary expression: It contains three operands and one operator.
  - Example: `Exp1? Exp2 – Exp3  // if Exp1 is true, Exp2 is executed. Otherwise, Exp3 is executed`

Operators

- C language is rich in built-in operators and provides the following types of operators:
  - Arithmetic Operators
  - Relational Operators
  - Logical Operators
  - Bitwise Operators
  - Assignment Operators
  - Misc Operators

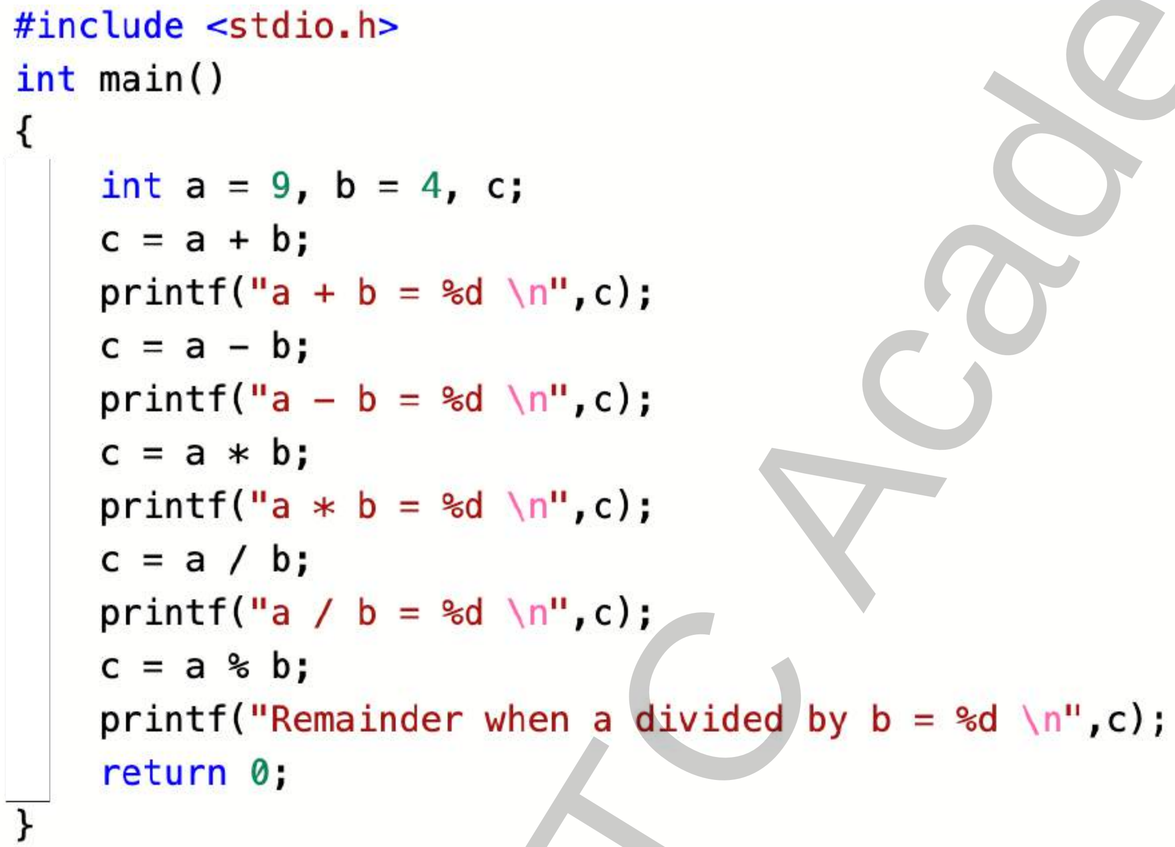
Arithmetic Operators

- An arithmetic operator performs mathematical operations such as addition, subtraction, multiplication, division etc on numerical values (constants and variables).
- The following table shows all the arithmetic operators supported by the C language (assume variable A =10 and variable B = 20).

Arithmetic Operators

Operator	Description	Example
`+`	Adds two operands	`A + B = 30`
`-`	Subtracts second operand from the list	`A – B = -10`
`*`	Multiplies both operands	`A * B = 200`
`/`	Divides numerator by de-numberator	`B / A = 2`
`%`	Modulus & remainder of after an integer division	`B % A = 0`
`++`	Increment operator increases the integer value by one	`A++ = 11`
`--`	Decrement operator increases the integer value by one	`A-- = 9`

Arithmetic Operators Example



```c
#include <stdio.h>
int main()
{
    int a = 9, b = 4, c;
    c = a + b;
    printf("a + b = %d \n",c);
    c = a - b;
    printf("a - b = %d \n",c);
    c = a * b;
    printf("a * b = %d \n",c);
    c = a / b;
    printf("a / b = %d \n",c);
    c = a % b;
    printf("Remainder when a divided by b = %d \n",c);
    return 0;
}
```

Relational Operators

- A relational operator checks the relationship between two operands.
- If the relation is true, it returns 1; if the relation is false, it returns value 0.
- Relational operators are used in decision making and loops.
- The following table shows all the arithmetic operators supported by the C language (assume variable A =10 and variable B = 20).

Relational Operators

Operator	Description	Example
`==`	Checks if the values of two operands are equal or not. If yes, then the condition becomes true	`(A == B)` is not true
`!=`	Checks if the values of two operands are equal or not. If the values are not equal, then the condition becomes true.	`(A != B)` is true
`>`	Checks if the value of left operand is greater than the value of right operand. If yes, then the condition becomes true.	`(A > B)` is not true
`<`	Checks if the value of left operand is less than the value of right operand. If yes, then the condition becomes true	`(A < B)` is true
`>=`	Checks if the value of left operand is greater than or equal to the value of right operand. If yes, then the condition becomes true.	`(A >= B)` is not true
`<=`	Checks if the value of left operand is less than or equal to the value of right operand. If yes, then the condition becomes true.	`(A <= B)` is true

Logical Operators

- An expression containing logical operator returns either 0 or 1 depending upon whether expression results true or false.
- Logical operators are commonly used in decision making in C programming.
- The following table shows all the arithmetic operators supported by the C language (assume variable A =1 and variable B = 0).

Logical Operators

Operator	Description	Example
`&&`	Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.	`(A && B) is false`
`||`	Called Logical OR Operator. If any of the two operands is non-zero, then the condition becomes true.	`(A || B) is true`
`!`	Called Logical NOT Operator. It is used to reverse the logical state of its operand. If a condition is true, then Logical NOT operator will make it false.	`!(A && B) is true`

Bitwise Operators

- In arithmetic-logic unit (which is within the CPU), mathematical operations like: addition, subtraction, multiplication and division are done in bit-level.
- To perform bit-level operations in C programming, bitwise operators are used.
- Assume variable:

```
A = 60 = 0011 1100
B = 13 = 0000 1101
```

Bitwise Operators

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands	**(A & B) = 12** **0000 1100**
|	Binary OR Operator copies a bit if it exists in either operand	**(A | B) = 61** **0011 1101**
^	Binary XOR Operator copies the bit if it is set in one operand but not both	**(A ^ B) = 49** **0011 0001**
~	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits	**(~A) = -61** **1100 0011**
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	**A << 2 = 240** **1111 0000**
<<	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	**A >> 2 = 15** **0000 1111**

Assignment Operators

- An assignment operation assigns the value of the right-hand operand to the storage location named by the left-hand operand.
- The assignment operators in C can both transform and assign values in a single operation.
- C provides the following assignment operators:

Assignment Operators

Operator	Description	Example
=	Simple assignment operator. Assigns values from right side operands to left side operand	`C = A + B will assign the value of A + B to C`
+=	Add AND assignment operator. It adds the right operand to the left operand and assign the result to the left operand.	`C += A is equivalent to C = C + A`
-=	Subtract AND assignment operator. It subtracts the right operand from the left operand and assigns the result to the left operand.	`C -= A is equivalent to C = C - A`
*=	Multiply AND assignment operator. It multiplies the right operand with the left operand and assigns the result to the left operand.	`C *= A is equivalent to C = C * A`
/=	Divide AND assignment operator. It divides the left operand with the right operand and assigns the result to the left operand.	`C /= A is equivalent to C = C / A`
%=	Modulus AND assignment operator. It takes modulus using two operands and assigns the result to the left operand.	`C %= A is equivalent to C = C % A`

Misc Operators

- Besides the operators discussed above, there are a few other important operators including `sizeof` and `? :` supported by the C language.

Operator	Description	Example
`sizeof()`	Returns the size of a variable.	**`sizeof(a), where a is integer, will return 4.`**
`? :`	Conditional Expression.	**`if condition is true ? then value X : otherwise value Y`**
`&`	Returns the address of a variable.	**`&a; returns the actual address of the variable.`**
`*`	Pointer to a variable.	**`*a;`**

Operators Precedence in C

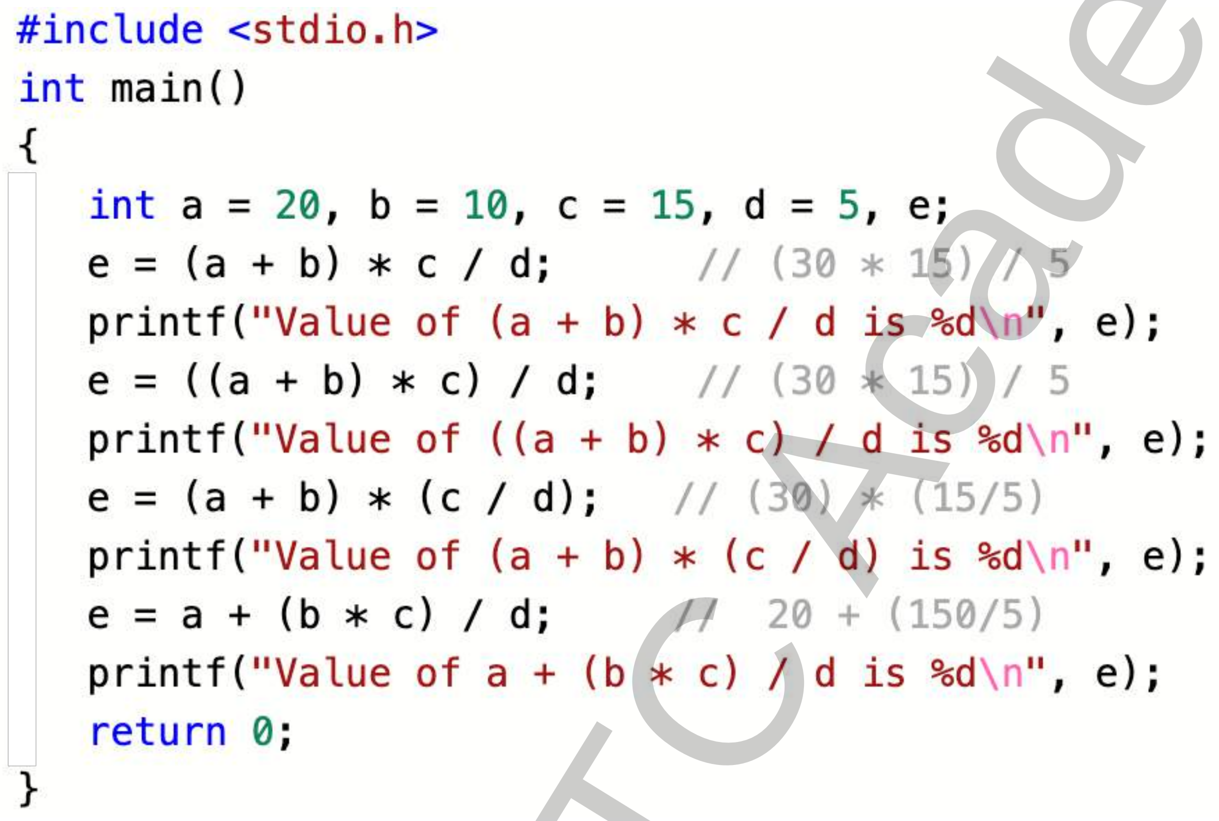
- Operator precedence determines the grouping of terms in an expression and decides how an expression is evaluated
- Operators with the highest precedence appear at the top of the table, those with the lowest appear at the bottom
- Within an expression, higher precedence operators will be evaluated first

Category	Operator	Associativity
Postfix	`() [] -> . ++ - -`	Left to Right
Unary	`+ - ! ~ ++ -- (type)* & sizeof()`	Right to Left
Multiplicative	`* / %`	Left to Right
Additive	`+ -`	Left to Right

Operators Precedence in C

Category	Operator	Associativity
Shift	`<< >>`	Left to Right
Relational	`< <= > >=`	Left to Right
Equality	`== !=`	Left to Right
Bitwise	`& | ^ |`	Left to Right
Logical	`&& ||`	Left to Right
Conditional	`?:`	Right to Left
Assignment	`= += -= *= /= %=>>= <<= &= ^= |=`	Right to Left
Comma	`,`	Left to Right

Operators Precedence Example

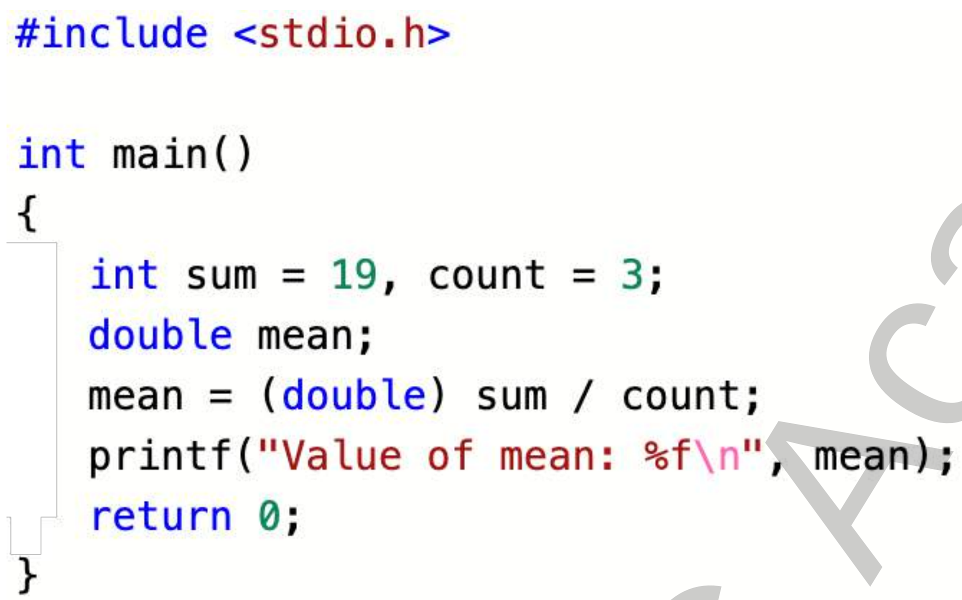


```c
#include <stdio.h>
int main()
{
    int a = 20, b = 10, c = 15, d = 5, e;
    e = (a + b) * c / d;      // (30 * 15) / 5
    printf("Value of (a + b) * c / d is %d\n", e);
    e = ((a + b) * c) / d;    // (30 * 15) / 5
    printf("Value of ((a + b) * c) / d is %d\n", e);
    e = (a + b) * (c / d);   // (30) * (15/5)
    printf("Value of (a + b) * (c / d) is %d\n", e);
    e = a + (b * c) / d;      //  20 + (150/5)
    printf("Value of a + (b * c) / d is %d\n", e);
    return 0;
}
```

Type Casting

- Typecasting is a way to make a variable of one type, such as an `int`, act like another type, such as a `char`, for one single operation
- To typecast something, simply put the type of variable you want the actual variable to act as inside parentheses in front of the actual variable
- Syntax: `(type_name) expression`
- Example where the cast operator causes the division of one integer variable by another to be performed as a floating-point operation:

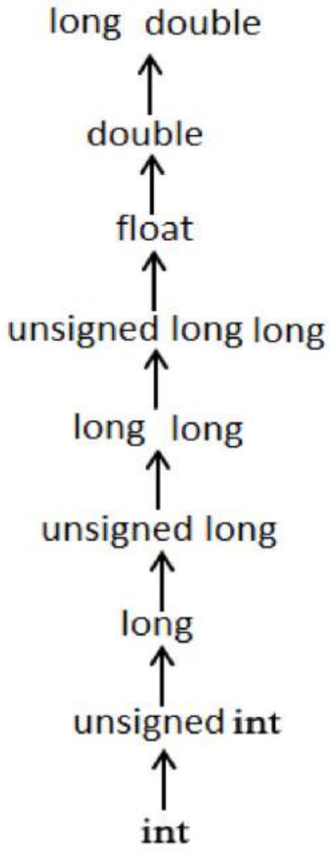
Type Casting Example



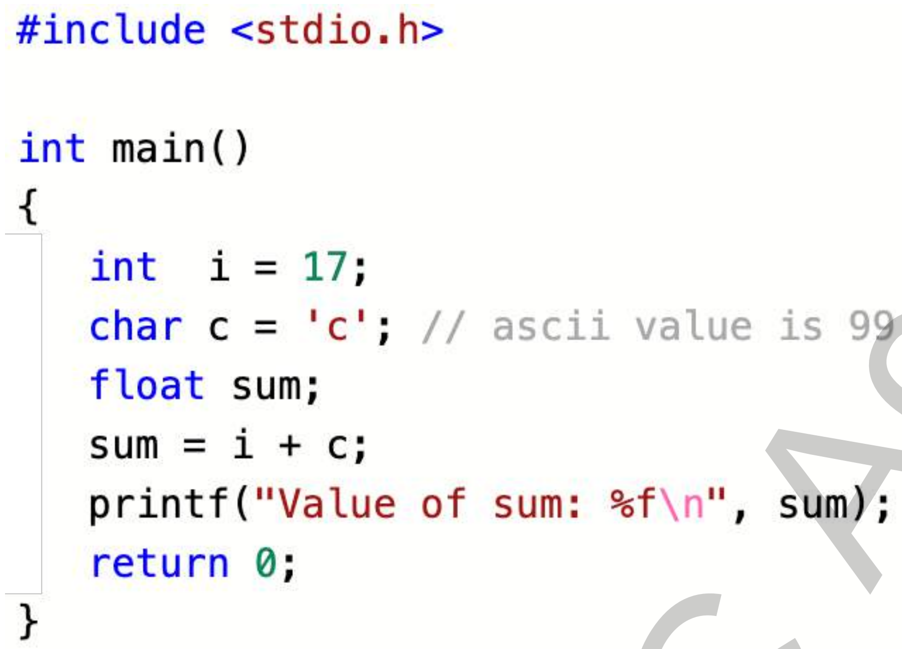
```c
#include <stdio.h>

int main()
{
    int sum = 19, count = 3;
    double mean;
    mean = (double) sum / count;
    printf("Value of mean: %f\n", mean);
    return 0;
}
```

Usual Arithmetic Conversion

- The usual arithmetic conversions are implicitly performed to cast their values to a common type.
- The compiler first performs integer promotion; if the operands still have different types, then they are converted to the type that appears highest in the right hierarchy.
- The usual arithmetic conversions are not performed for the assignment operators, nor for the logical operators && and ||.



Usual Arithmetic Conversion Example



```c
#include <stdio.h>

int main()
{
    int  i = 17;
    char c = 'c'; // ascii value is 99
    float sum;
    sum = i + c;
    printf("Value of sum: %f\n", sum);
    return 0;
}
```

Summary

- Expression is a combination of Operators and Operands
- C language is rich in built-in operators and provides the following types of operators:
  - Arithmetic Operators
  - Relational Operators
  - Logical Operators
  - Bitwise Operators
  - Assignment Operators
- Typecasting is a way to make a variable of one type

VTC Academy
Think Ahead, Beyond Boundaries

Thank you!