Rebooting C

Revisiting the Primary Concepts

Outline

What is C?

- C is a language written by Brian Kernighan and Dennis Ritchie. This was to be the language that UNIX was written in to become the first “portable” language.

- In recent years C has been used as a general-purpose language because of its popularity with programmers.

Why use C?

Mainly because it produces code that runs nearly as fast as code written in assembly language. Some examples of the use of C might be:

- Operating Systems
- Language Compilers
- Assemblers
- Text Editors
- Print Spoolers
- Network Drivers
- Modern Programs
- Data Bases
- Language Interpreters
- Utilities

Mainly because of the portability that writing standard C programs can offer.

History

- In 1972 Dennis Ritchie at Bell Labs wrote C and in 1978 the publication of **The C Programming Language** by Kernighan & Ritchie caused a revolution in the computing world.

- In 1983, the American National Standards Institute (ANSI) established a committee to provide a modern, comprehensive definition of C. The resulting definition, the ANSI

Why C Still Useful?

- C Provides
  - Efficiency, high performance and high quality
  - Flexibility and power
  - Many high-level and low-level operations middle-level
  - Stability and small size code
  - Provide functionality through rich set of function libraries
  - Gateway for other professional languages like C C++ JAVA

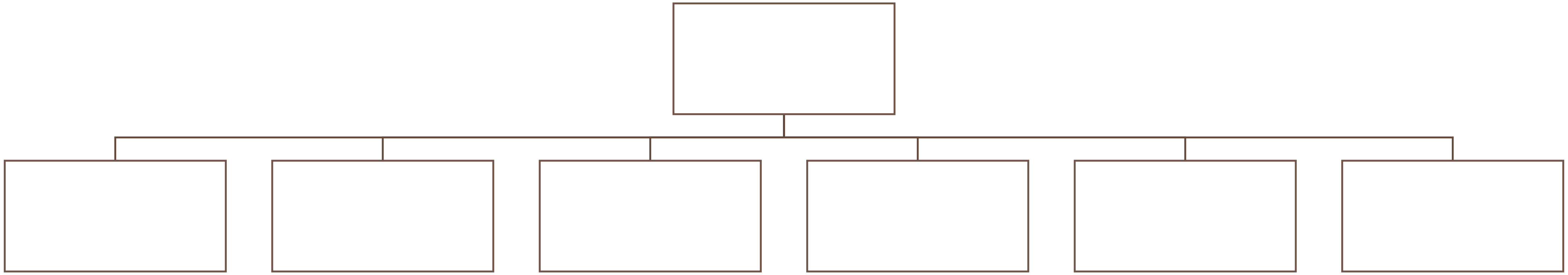
Why C Still Useful?

- C is used:
  - System software Compilers, Editors, Embedded systems
  - Data compression, graphics and computational geometry, utility programs
  - Databases, operating systems, device drivers, system level routines
  - There are zillions of lines of C legacy code
  - Also used in application programs

Software Development Method

- Requirement Specification
  - Problem Definition
- Analysis
  - Refine, Generalize, and Decompose the problem definition
- Design
  - Develop Algorithm
- Implementation
  - Wite Code
- Verification and Testing
  - Test and Debug the code

C Tokens



float	main	-15.5	“ABC”	[ ]	+ -
while	amount	100	“year”	{ }	* ,

C Keywords

- Keywords are the words whose meaning has already been explained to the C compiler (or in a broad sense to the computer).
- The keywords cannot be used as variable names because if we do so we are trying to assign a new meaning to the keyword, which is not allowed by the computer.

C Keywords

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

A Simple Program Example

```
#include<stdio.h>
void main()
{
int basic, da, hra;
float gross;
da=45; hra=15;
printf("Enter the Basic Salary of the Employee: ");
scanf("%d",&basic);
/*Formula for Gross Calculation*/
gross=basic + (basic*da/100) + (basic*hra/100);
printf("The gross salary of the employee is %f", gross);
}
```

C Instructions

There are basically three types of instructions in C:

a) **Type declaration instruction** – To declare the type of variables used in a C program.

b) **Arithmetic instruction** – To perform arithmetic operations between constants and variables.

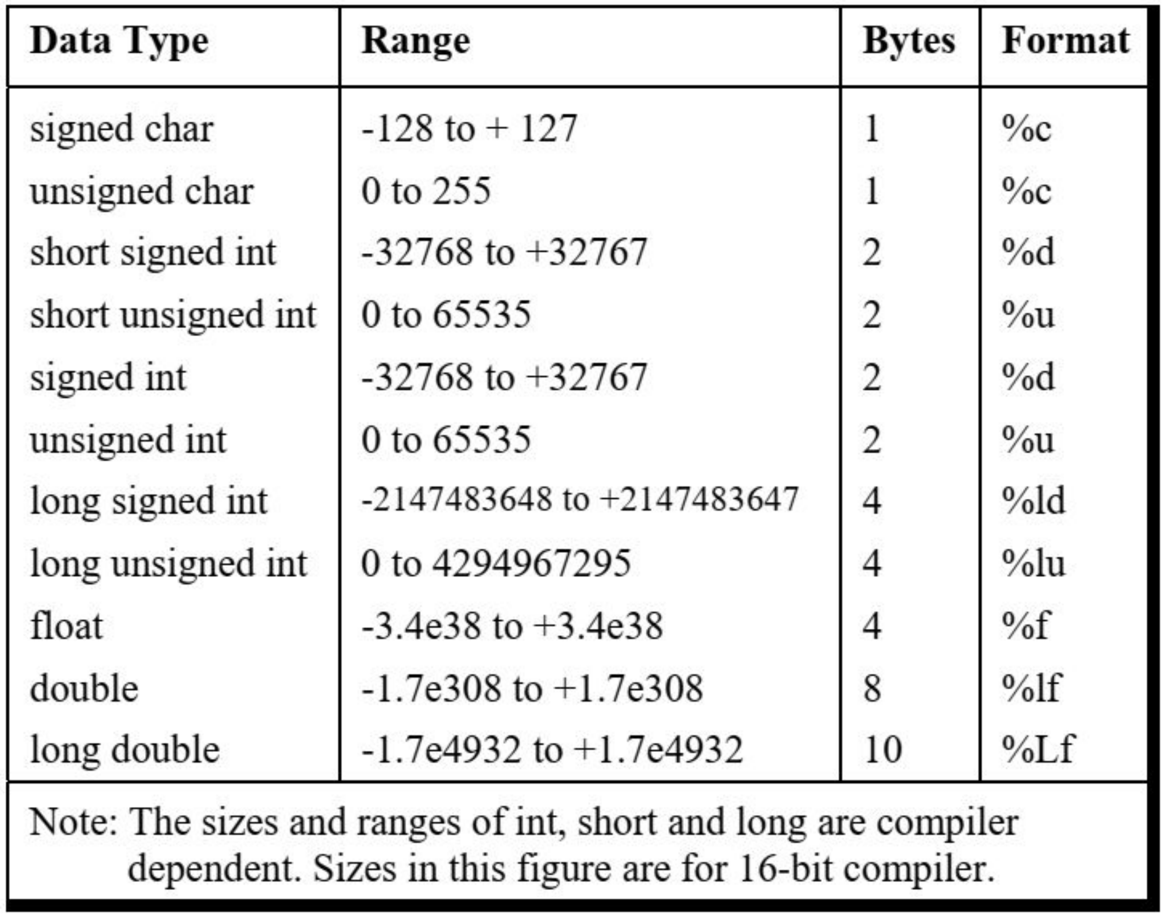
c) **Control instruction** – To control the sequence of execution of various statements in a C program.

Type Declaration Instruction

- This instruction is used to declare the type of variables being used in the program.
- Any variable used in the program must be declared before using it in any statement.
- The type declaration statement is written at the beginning of main( ) function.

```
Ex. int bas ;
            float rs, grosssal ;
            char name, code ;
```

Data Types



Data Type	Range	Bytes	Format
signed char	-128 to + 127	1	%c
unsigned char	0 to 255	1	%c
short signed int	-32768 to +32767	2	%d
short unsigned int	0 to 65535	2	%u
signed int	-32768 to +32767	2	%d
unsigned int	0 to 65535	2	%u
long signed int	-2147483648 to +2147483647	4	%ld
long unsigned int	0 to 4294967295	4	%lu
float	-3.4e38 to +3.4e38	4	%f
double	-1.7e308 to +1.7e308	8	%lf
long double	-1.7e4932 to +1.7e4932	10	%Lf

Note: The sizes and ranges of int, short and long are compiler dependent. Sizes in this figure are for 16-bit compiler.

Arithmetic Instruction

A **C** arithmetic instruction consists of a variable name on the left hand side of = and variable names & constants on the right hand side of =.

Ex.:

```
int ad ;
float kot, deta, alpha, beta, gamma ;
ad =
 3200 ;
kot = 0.0056 ;
deta = alpha * beta / gamma + 3.2 * 2 / 5 ;
```

Here,

*, /, -, + are the arithmetic operators.

= is the assignment operator.

2, 5 and 3200 are integer constants.

3.2 and 0.0056 are real constants.

ad is an integer variable.

kot, deta, alpha, beta, gamma are real variables.

Integer and Float Conversions

The implicit conversion of floating point and integer values in C.

a) An arithmetic operation between an integer and another integer always yields an integer result.

b) An operation between a real and real always yields a real result.

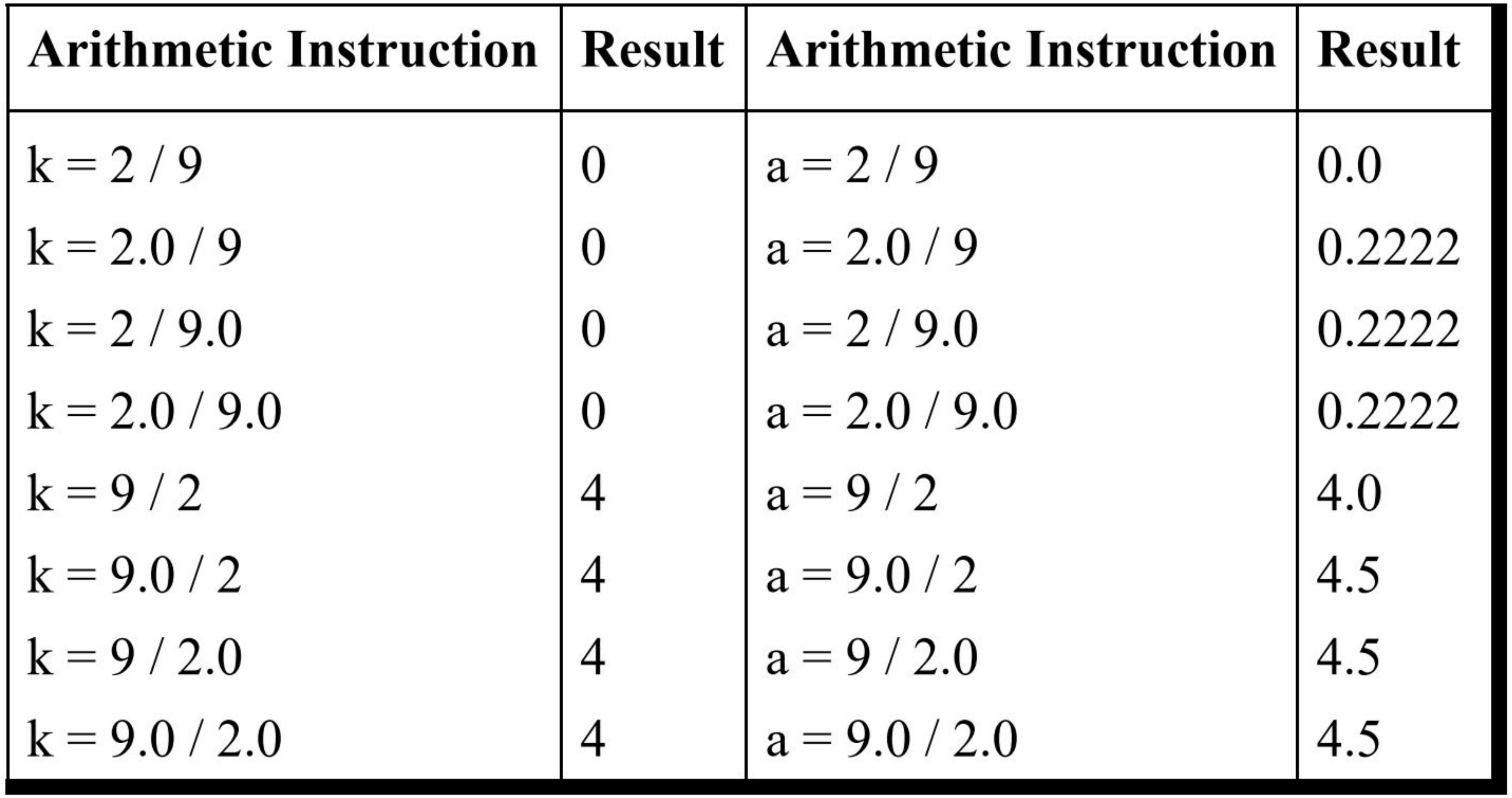
c) An operation between an integer and real always yields a real result.

 In this operation, the integer is first promoted to a real and then the operation is performed. Hence the result is real.

Integer and Float Conversions

Operation	Result	Operation	Result
5 / 2	2	2 / 5	0
5.0 / 2	2.5	2.0 / 5	0.4
5 / 2.0	2.5	2 / 5.0	0.4
5.0 / 2.0	2.5	2.0 / 5.0	0.4

Integer and Float Conversions



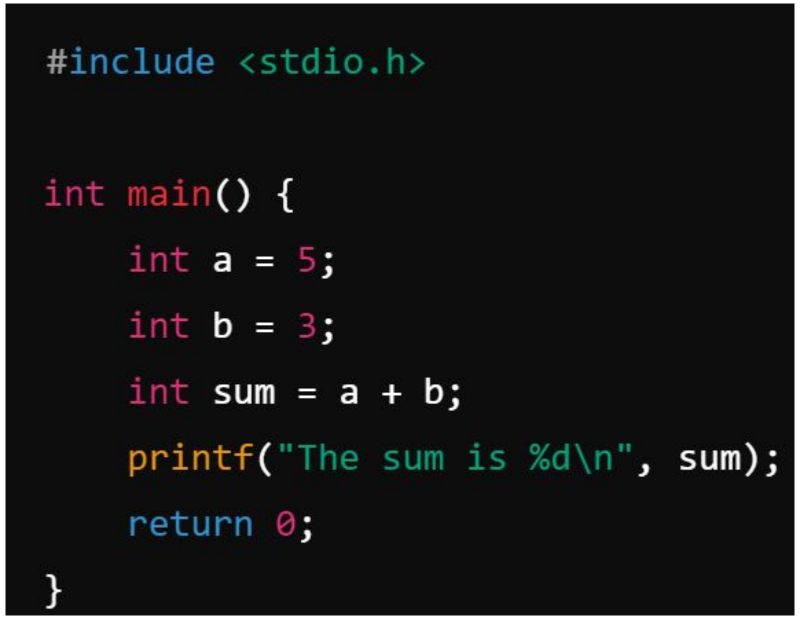
Arithmetic Instruction	Result	Arithmetic Instruction	Result
k = 2 / 9	0	a = 2 / 9	0.0
k = 2.0 / 9	0	a = 2.0 / 9	0.2222
k = 2 / 9.0	0	a = 2 / 9.0	0.2222
k = 2.0 / 9.0	0	a = 2.0 / 9.0	0.2222
k = 9 / 2	4	a = 9 / 2	4.0
k = 9.0 / 2	4	a = 9.0 / 2	4.5
k = 9 / 2.0	4	a = 9 / 2.0	4.5
k = 9.0 / 2.0	4	a = 9.0 / 2.0	4.5

Control Instructions in C

- As the name suggests the ‘Control Instructions’ enable us to specify the order in which the various instructions in a program are to be executed by the computer.
- In other words the control instructions determine the ‘flow of control’ in a program.
- There are four types of control instructions in C. They are:
  a) Sequence Control Instruction
  b) Selection or Decision Control Instruction
  c) Repetition or Loop Control Instruction
  d) Case Control Instruction

Control Instructions in C

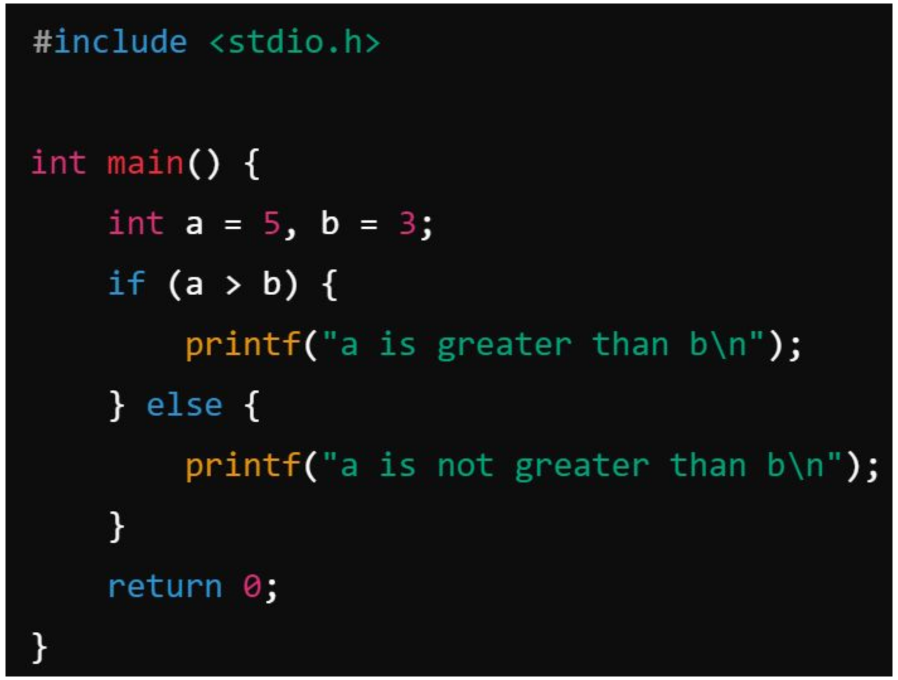
- Sequence Control Instruction



```c
#include <stdio.h>

int main() {
    int a = 5;
    int b = 3;
    int sum = a + b;
    printf("The sum is %d\n", sum);
    return 0;
}
```

- Selection or Decision Control Instruction

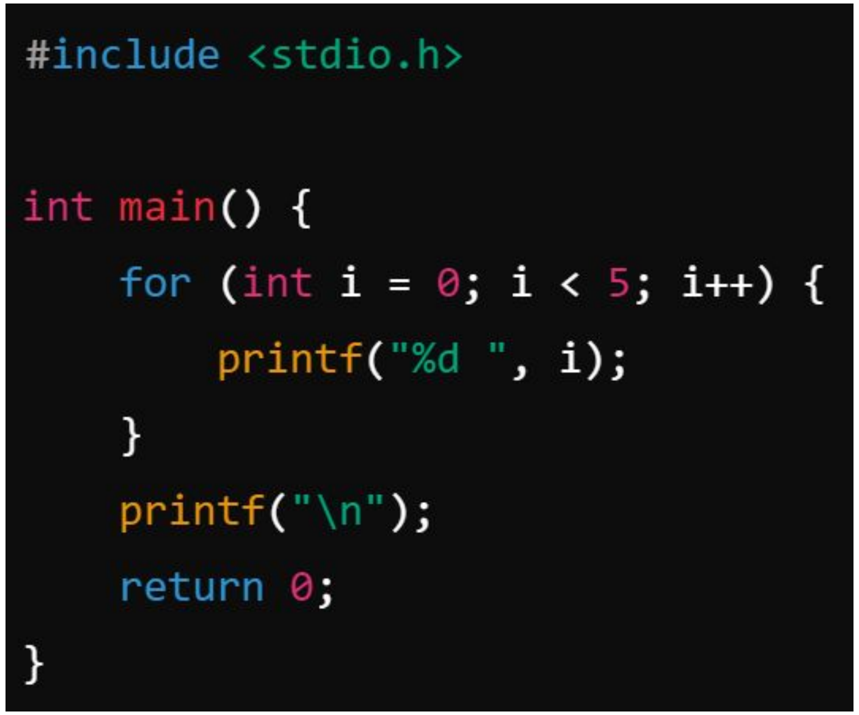


```c
#include <stdio.h>

int main() {
    int a = 5, b = 3;
    if (a > b) {
        printf("a is greater than b\n");
    } else {
        printf("a is not greater than b\n");
    }
    return 0;
}
```

Control Instructions in C

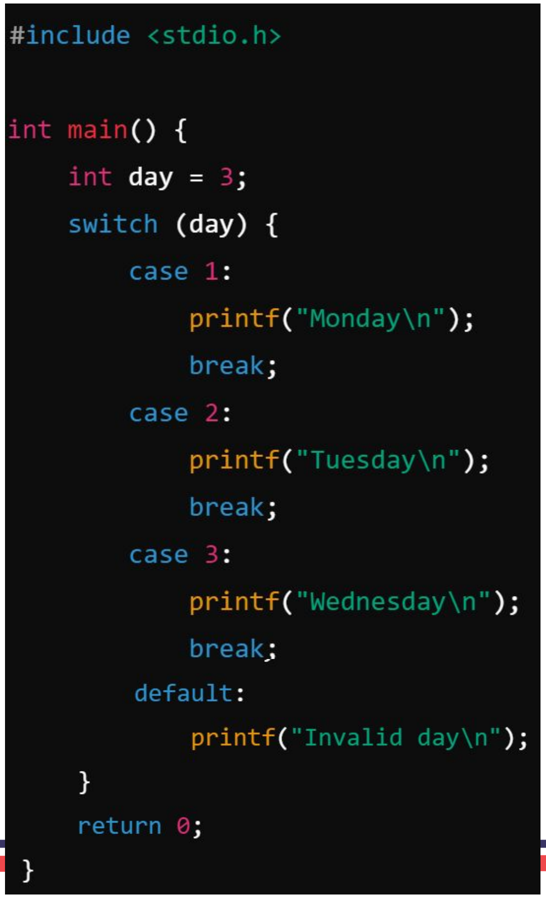
- Repetition or Loop Control Instruction



```c
#include <stdio.h>

int main() {
    for (int i = 0; i < 5; i++) {
        printf("%d ", i);
    }
    printf("\n");
    return 0;
}
```

- Case Control Instruction



```c
#include <stdio.h>

int main() {
    int day = 3;
    switch (day) {
        case 1:
            printf("Monday\n");
            break;
        case 2:
            printf("Tuesday\n");
            break;
        case 3:
            printf("Wednesday\n");
            break;
        default:
            printf("Invalid day\n");
    }
    return 0;
}
```

Some Other Instruction Types

- Input/Output Instructions
- Logical Instructions
- Bitwise Instructions
- Assignment Instructions

UNIVERSITY OF ENGINEERING & MANAGEMENT
Good Education, Good Jobs

C - Operators

- An operator is a symbol that tells the compiler to perform specific mathematical or logical functions.
- C language is rich in built-in operators and has the following types of operators –
  - Arithmetic Operators
  - Relational Operators
  - Logical Operators
  - Bitwise Operators
  - Assignment Operators
  - Misc. Operators

Arithmetic Operators

A=10 $(00110)_2$ B=20 $(10100)_2$

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

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

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

- Bitwise operator works on bits and performs a bit-by-bit operation. The truth tables for &, |, and ^ are as follows −

p	q	p & q	p | q	p ^ q
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

- Assume A = 60 and B = 13 in binary format, they will be as follows −

A = 0011 1100 B = 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, i.e., 0000 1100
|	Binary OR Operator copies a bit if it exists in either operand.	(A | B) = 61, i.e., 0011 1101
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) = 49, i.e., 0011 0001
~	Binary One's Complement Operator is unary and has the effect of 'flipping' bits.	(~A) = ~(60), i.e., -0111101
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 = 240 i.e., 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 i.e., 0000 1111

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

Assignment Operators

Operator	Description	Example
<<=	Left shift AND assignment operator.	C <<= 2 is same as C = C << 2
>>=	Right shift AND assignment operator.	C >>= 2 is same as C = C >> 2
&=	Bitwise AND assignment operator.	C &= 2 is same as C = C & 2
^=	Bitwise exclusive OR and assignment operator.	C ^= 2 is same as C = C ^ 2
|=	Bitwise inclusive OR and assignment operator.	C |= 2 is same as C = C | 2

Misc. Operators ↦ sizeof & ternary

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

Operators Precedence in C

Category	Operator	Associativity
Postfix	() [] -> . ++ - -	Left to right
Unary	+ - ! ~ ++ - - (type)* & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right

Operators Precedence in C

Category	Operator	Associativity
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %=>>= <<= &= ^= |=	Right to left
Comma	,	Left to right

THANK YOU