C Programming

History of C

- Born at AT & T Bell Laboratory of USA in 1972
- Many of C’s principles and ideas were derived from the earlier language B
- Ken Thompson was the developer of B Language
- C was written by Dennis Ritchie
- C language was created for a specific purpose i.e designing the UNIX operating system (which is currently base of many UNIX based OS)
- Quickly spread beyond Bell Labs in the late 70’s because of its strong features

Features of C language

- Portability - C Programs can run on any compiler with little or no modification
- Low level features: C provides low level features and is closely related to lower level **assembly Languages**
- Modular programming - software design technique that increases the extent to which software is composed of separate parts, called **modules**
- Has many successor languages which are designed to look like C, e.g., C++, C#, Objective-C, Java, JavaScript, PHP and Perl.

C is a structured programming language

- Divides the large problem in to smaller modules called functions or procedures
- Each function or module handles the particular task and the collection of all the functions is called a program, which solves the large problem
- Easier to modify and debug

Difference between C and Python

- **C programs – Compiled**
- **Python programs – Interpreted**

Compiler	Interpreter
Takes **entire** program as input and generate a output file with object code	Takes instruction by instruction as input and gives an output. But does not generate a file
Errors are displayed after **entire program** is checked	**Errors** are displayed for **every instruction** interpreted (if any)

Variable declaration in C

- In C, it is mandatory to do variable declaration
- We say variable's **type**, whether it is an integer (**int**), floating-point number (**float**), character (**char**) etc
- Syntax is type of variable, white space, name of variable semicolon
- Eg: int number;

White spaces and Indentation

- No problem of difference between white space and tab in C (Happy!)
- Block of code in C need not be intended as in Python
- In C, Curly braces are used for giving a block of code

  Eg: Block of code in ‘C’

```
{
-----
}
```

Problem

- Little Bob loves chocolate, and he goes to a store with Rs. *N* in his pocket. The price of each chocolate is Rs. C. The store offers a discount: for every *M* wrappers he gives to the store, he gets one chocolate for free. This offer is available only once. How many chocolates does Bob get to eat?

Pseudocode

```
• READ N and C
• COMPUTE num_of_chocolates as N/C
• CALCULATE returning_wrapper as number of chocolates/m
• TRUNCATE decimal part of returning_wrapper
• COMPUTE Chocolates_received as num_of_chocolates + returning_wrapper
• PRINT Chocolates_received
```

Layout of a C Program

pre-processor directives – Preceded by a ‘#’
global declarations – Optional and not a good programming practice
int main() - standard start for all C programs
{
local variables to function main ; - all variables used in the function must be declared in the beginning
statements associated with function main ;
}
void f1()
{
local variables to function 1 ;
statements associated with function 1 ;
}

Components of a C program

A C program consists of the following parts:

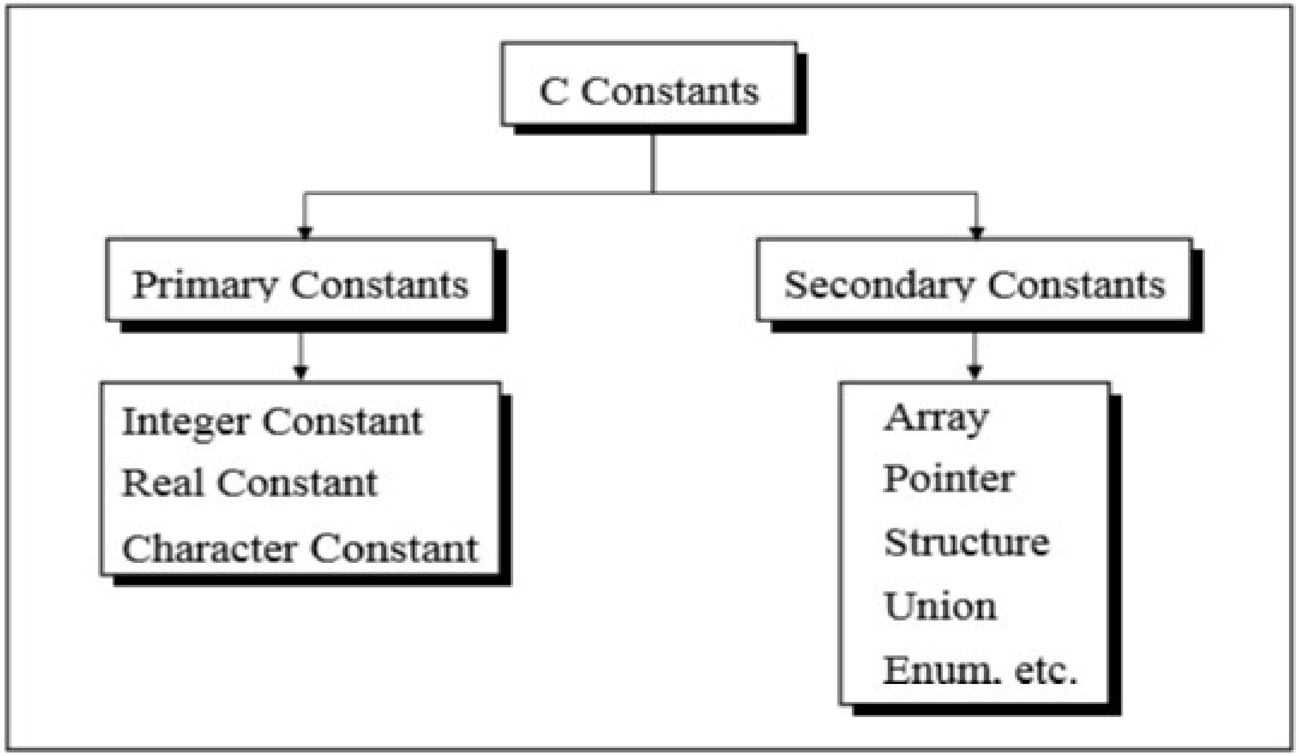
- Comments
- Variables
- Preprocessor Commands
- Functions
- Statements & Expressions

Comments in C program

Two types of comments

- Single line comment
    - Single Line Comment is double forward slash ‘//’ and can be **Placed Anywhere**
- Multi line comment
    - Multi line comment starts with /*
    - Multi line comment ends with */
    - Any symbols written between '/*’ and '*/‘ are ignored by Compiler

Types of C Constants



Now Restrict Discussion to Primary Constants

Data Types in C

Basic Arithmetic types - further classified into: (a) integer types and (b) floating-point types

Enumerated types - arithmetic types that are used to define variables that can be assigned only certain discrete integer values throughout the program

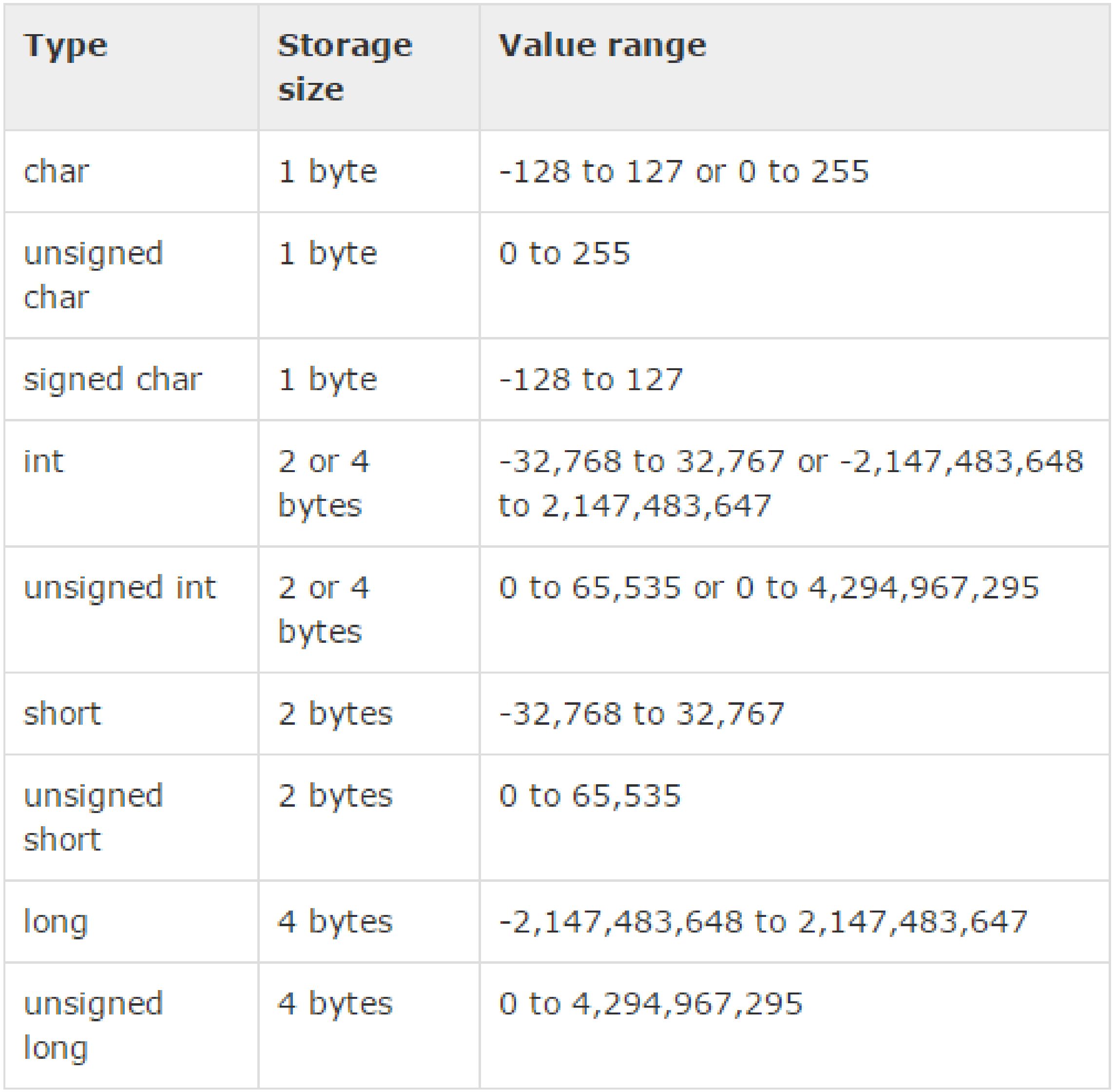
Type void - indicates that no value is available

Derived types - They include (a) Pointer types, (b) Array types, (c) Structure types, (d) Union types and (e) Function types

Broad Classification of C data types

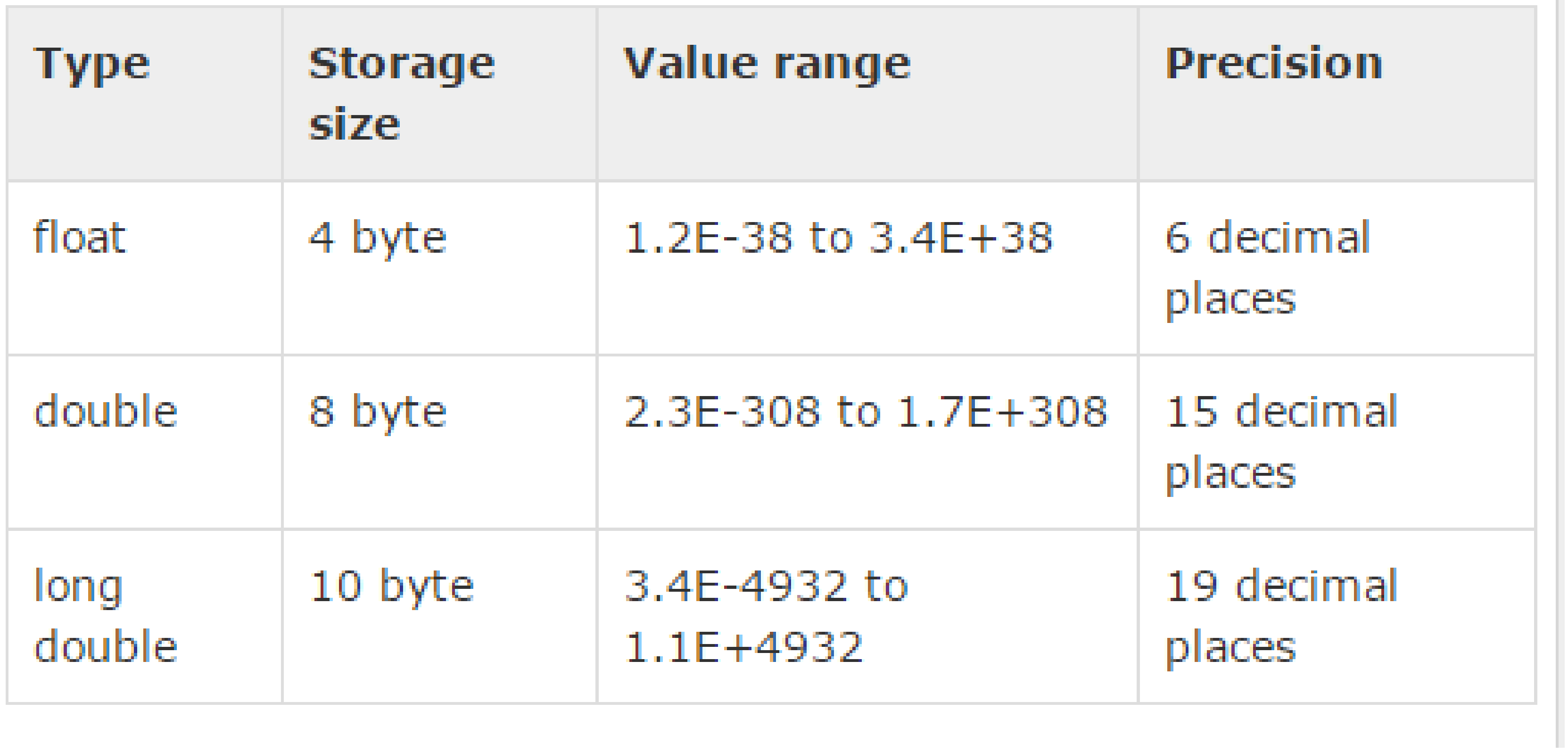
- Numerical data types are broadly classified into
  - Signed
  - Unsigned
- Signed can store zero, positive and negative values
- Unsigned can store only zero and positive values
- Some applications use unsigned data types only Eg: age

Integer Types



Type	Storage size	Value range
char	1 byte	-128 to 127 or 0 to 255
unsigned char	1 byte	0 to 255
signed char	1 byte	-128 to 127
int	2 or 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647
unsigned int	2 or 4 bytes	0 to 65,535 or 0 to 4,294,967,295
short	2 bytes	-32,768 to 32,767
unsigned short	2 bytes	0 to 65,535
long	4 bytes	-2,147,483,648 to 2,147,483,647
unsigned long	4 bytes	0 to 4,294,967,295

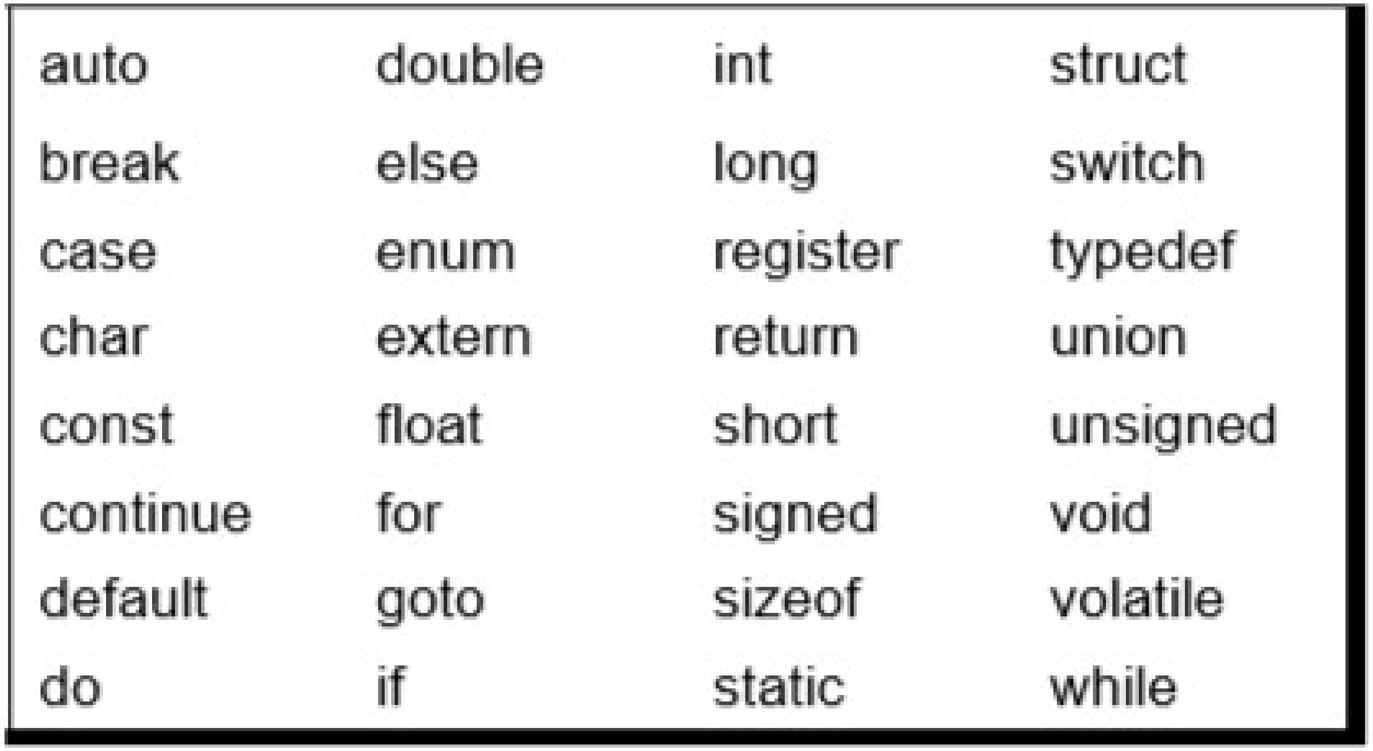
Floating Point Types



Type	Storage size	Value range	Precision
float	4 byte	1.2E-38 to 3.4E+38	6 decimal places
double	8 byte	2.3E-308 to 1.7E+308	15 decimal places
long double	10 byte	3.4E-4932 to 1.1E+4932	19 decimal places

Keywords

- 32 keywords available in C



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

Compiler vendors (like Microsoft, Borland, etc.) provide their own keywords

Variables

- BASICSALARY
- _basic
- basic-hra
- #MEAN
- group.
- 422
- population in 2006
- FLOAT
- hELLO

I/O in C

- Basic operation in any language
- Input is got through a function scanf which is equivalent to input or raw_input in Python
- Syntax of scanf
- **int scanf(const char *format, ...)**
- Basically two or more arguments
- First format string, followed by address of variables that are going to hold values entered by user

Printf and scanf format codes

code	type	format
d	int	decimal (base ten) number
o	int	octal number (no leading '0' supplied in printf)
x or X	int	hexadecimal number (no leading '0x' supplied in printf; accepted if present in scanf) (for printf, 'X' makes it use upper case for the digits ABCDEF)
ld	long	decimal number ('l' can also be applied to any of the above to change the type from 'int' to 'long')

printf and scanf format codes

code	type	format
u	Unsigned int	decimal number
lu	unsigned long	decimal number
c	char	single character
s	char pointer	string
f	float	number with six digits of precision
lf	double	number with six digits of precision

Address of a Variable

- Address of a variable can be obtained by putting a ‘&’ before the variable name

Example 1

```
#include<stdio.h>
void main()
{
int a = 27;
int b = 25;
int c = a - b;
printf("%d",c);
}
```

Output

```
2
```

Example 3

```
#include <stdio.h>
int main()
{
char a = 273;
char b = 25;
int c = a%b;
printf("%d",c);

}
```

Output

```
17
```

Example 3

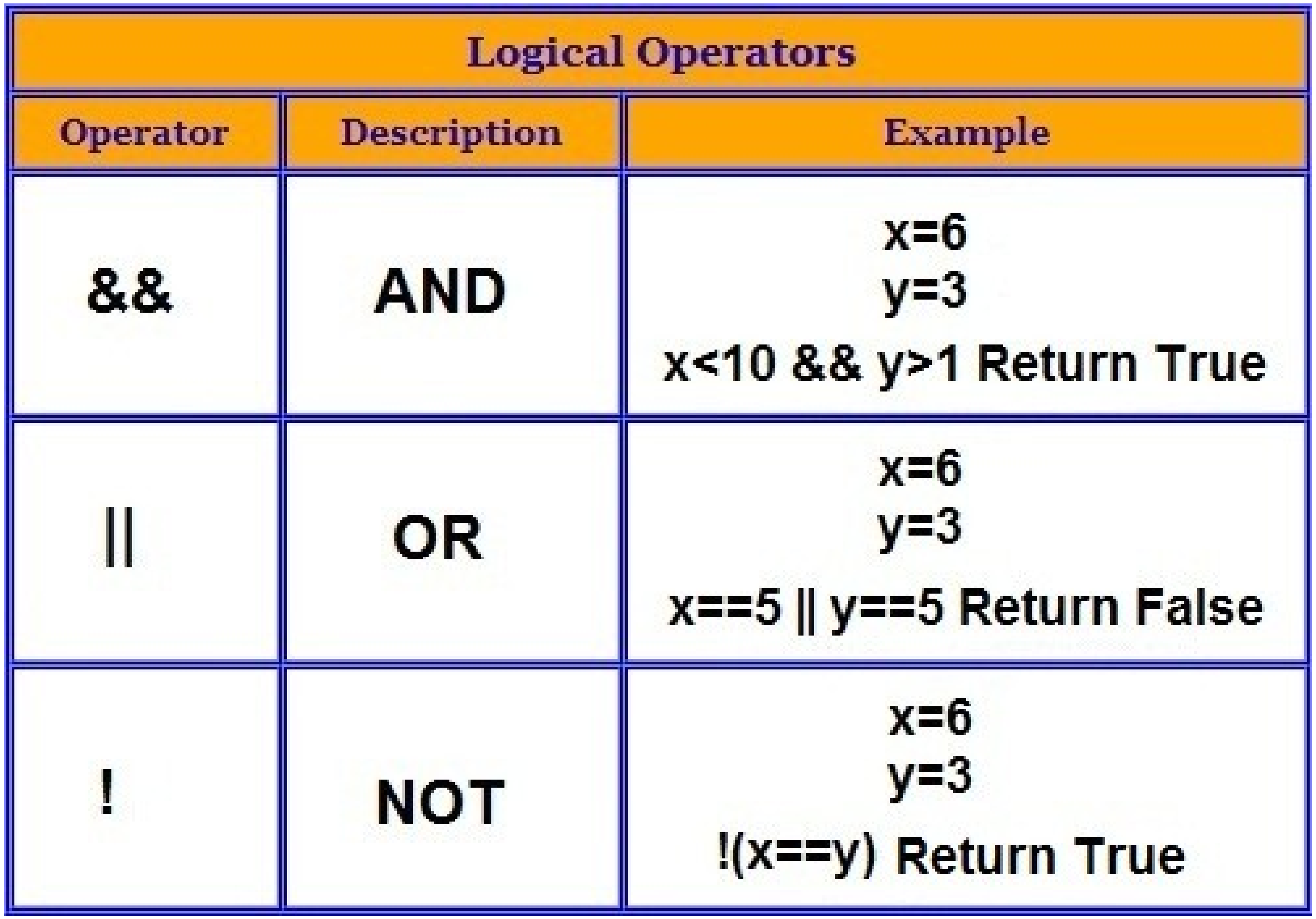
```
#include<stdio.h>
void main()
{
char a = 27;
char b = 25;
char c = a - b;
printf("%c",c);

}
```

Output

A special character

Logical Operators in C



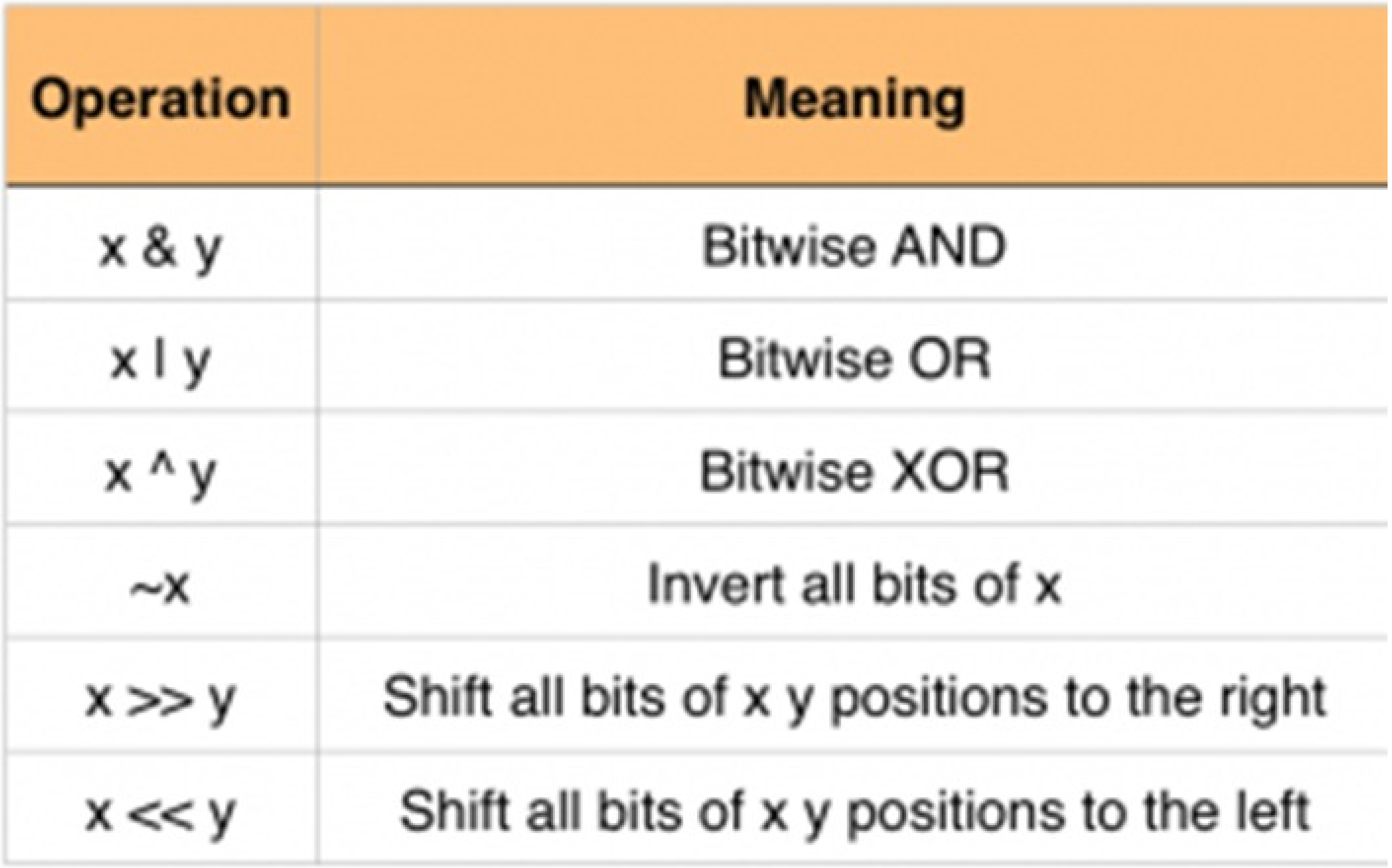
Logical Operators		
Operator	Description	Example

&&	AND	x=6
		y=3
		x<10 && y>1 Return True

||	OR	x=6
		y=3
		x==5 || y==5 Return False

!	NOT	x=6
		y=3
		!(x==y) Return True

Bitwise Operator

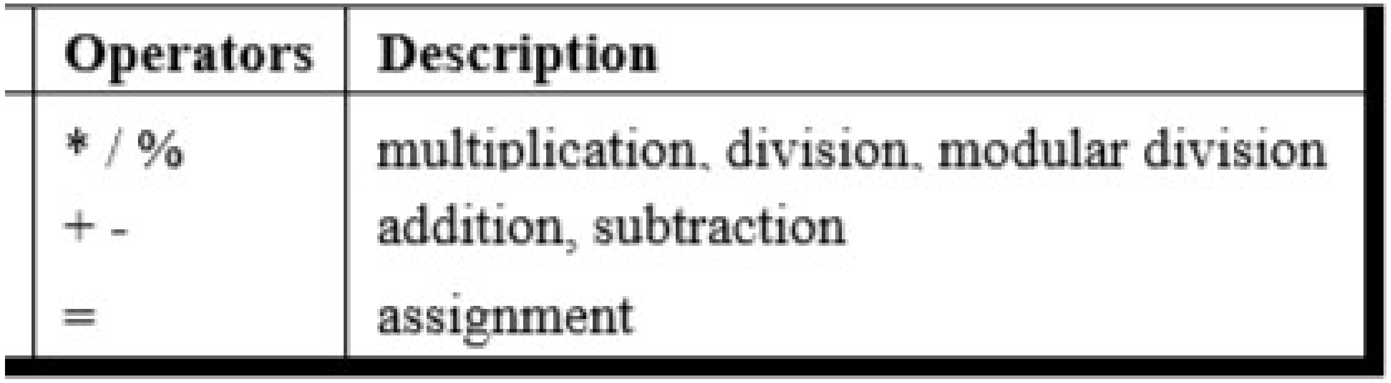


Operation	Meaning
x & y	Bitwise AND
x | y	Bitwise OR
x ^ y	Bitwise XOR
~x	Invert all bits of x
x >> y	Shift all bits of x y positions to the right
x << y	Shift all bits of x y positions to the left

Operator Precedence

++, -- Post increment Operators

++, -- Pre increment Operators



Operators	Description
* / %	multiplication, division, modular division
+ -	addition, subtraction
=	assignment

Parenthesis can be used to override default precedence

Evaluating Pre and Post Operator together

- Precedence of postfix ++ is higher than prefix ++ and their associativity is also different.
- Associativity of postfix ++ is left to right.
- Associativity of prefix ++ is right to left.
- The precedence of prefix ++ and * is the same with the right to left associativity.
- Precedence of postfix ++ is higher than * and their associativity is also different.

Example 4

```
#include<stdio.h>
main()
{
int a, b,c;
a = 4;
b = 2;
c = -a+--b;
printf ( "c = %d", c) ;
}
```

Output

```
-3
```

Example 5

```
#include<stdio.h>
main()
{
int a, b,c;
a = 4;
b = 2;

c = -a+ b--;

printf ( "c = %d", c) ;
printf ( "b = %d", b) ;
}
```

Output

```
c = -2 b = 1
```

Example 6

```
#include<stdio.h>
main()
{
int a, b,c;
a = 4;
c = ++a + a++;
printf ( "c = %d", c) ;
printf ( "a = %d", a) ;
}
```

Output

```
c = 11
a = 6
```

Example 7

```
#include<stdio.h>
main()
{
int a, b,c;
a = 4;
c = ++a + ++a;
printf ( "c = %d", c) ;
printf ( "a = %d", a) ;
}
```

Output

```
c = 12a = 6
```

Example 7

```
#include<stdio.h>
main()
{
int a, b,c;
a = 4;

c = a++ + ++a;

printf ( "c = %d", c) ;

printf ( "a = %d", a) ;

}
```

Output

```
c = 10 a = 6
```

OPERATOR	DESCRIPTION	ASSOCIATIVITY
( )	Parentheses (function call) (see Note 1)	left-to-right
[ ]	Brackets (array subscript)	
.	Member selection via object name	
->	Member selection via pointer	
++ —	Postfix increment/decrement (see Note 2)	
++ —	Prefix increment/decrement	right-to-left
+ –	Unary plus/minus	
! ~	Logical negation/bitwise complement	
(*type*)	Cast (convert value to temporary value of *type*)	
*	Dereference	
&	Address (of operand)	
sizeof	Determine size in bytes on this implementation	
* / %	Multiplication/division/modulus	left-to-right
+ –	Addition/subtraction	left-to-right
<< >>	Bitwise shift left, Bitwise shift right	left-to-right
< <=	Relational less than/less than or equal to	left-to-right
> >=	Relational greater than/greater than or equal to	
== !=	Relational is equal to/is not equal to	left-to-right
&	Bitwise AND	left-to-right

|	Bitwise inclusive OR	left-to-right
&&	Logical AND	left-to-right
| |	Logical OR	left-to-right
? :	Ternary conditional	right-to-left
=	Assignment	right-to-left
+= -=	Addition/subtraction assignment	
*= /=	Multiplication/division assignment	
%= &=	Modulus/bitwise AND assignment	
^= |=	Bitwise exclusive/inclusive OR assignment	
<<= >>=	Bitwise shift left/right assignment	
,	Comma (separate expressions)	left-to-right

Type Conversion in C

Convert a variable from one data type to another data type.

When the type conversion is performed automatically by the compiler without programmers intervention, such type of conversion is known as **implicit type conversion** or **type promotion**.

The compiler converts all operands into the data type of the largest operand.

Rules for Implicit Type Conversion in C

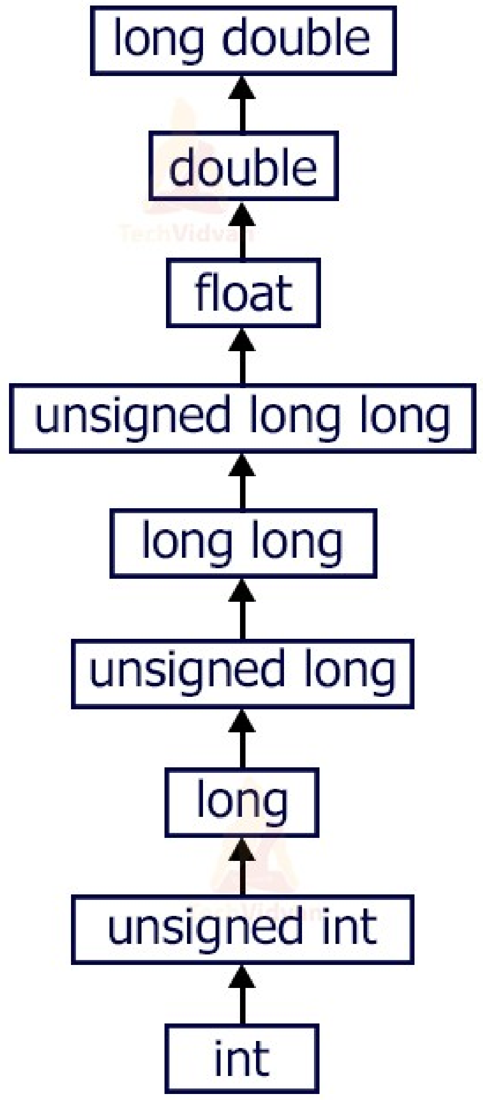
Sequence of rules that are applied while evaluating expressions are given below:

- All short and char are automatically converted to int, then,
- If either of the operand is of type long double, then others will be converted to long double and result will be long double.
- Else, if either of the operand is double, then others are converted to double.
- Else, if either of the operand is float, then others are converted to float.

Rules for Type Conversion in C

- Else, if either of the operand is unsigned long int, then others will be converted to unsigned long int.
- Else, if one of the operand is long int, and the other is unsigned int, then
  - if a long int can represent all values of an unsigned int, the unsigned int is converted to long int.
  - otherwise, both operands are converted to unsigned long int.
- Else, if either operand is long int then other will be converted to long int.
- Else, if either operand is unsigned int then others will be converted to unsigned int.

Conversion Hierarchy in C


$$\begin{tikzcd}
\boxed{\text{long double}} \\
\boxed{\text{double}} \arrow[u] \\
\boxed{\text{float}} \arrow[u] \\
\boxed{\text{unsigned long long}} \arrow[u] \\
\boxed{\text{long long}} \arrow[u] \\
\boxed{\text{unsigned long}} \arrow[u] \\
\boxed{\text{long}} \arrow[u] \\
\boxed{\text{unsigned int}} \arrow[u] \\
\boxed{\text{int}} \arrow[u]
\end{tikzcd}$$

Example

```
#include<stdio.h>
int main(){
     short a=10;
     int b;
     b=a;
     printf("Implicit type casting : %d\n",a);
}
```

Output

```
Implicit type casting : 10
```

Example

```
#include<stdio.h>
void main()
{
int a = 165;
int b = 100;
float c = a/b;
printf("%f",c);

}
```

Output

```
1.000000
```

Explicit Type Conversion

Type conversion performed by the programmer is known

as explicit type conversion

Explicit type conversion is also known as **type casting**.

Type casting in c is done in the following form:

(data_type)expression;

where, *data_type* is any valid c data type,

and *expression* may be constant, variable or an expression

For example, x=(int)a+b*d;

Explicit Type Conversion

The following rules have to be followed while converting the expression from one type to another to avoid the loss of information:

- All integer types to be converted to float.
- All float types to be converted to double.
- All character types to be converted to integer.

Example

```
#include<stdio.h>
void main()
{
int a = 165;
int b = 100;
float c = (float)a/b;
printf("%f",c);


}
```

Output

```
1.650000
```

Example

```
#include <stdio.h>
int main() {
   float c = 2.342;
   int s = (int)c+1;
   printf("Explicit Conversion : %d\n",s);
   return 0;
}
```

Output

```
Explicit Conversion : 3
```

Example

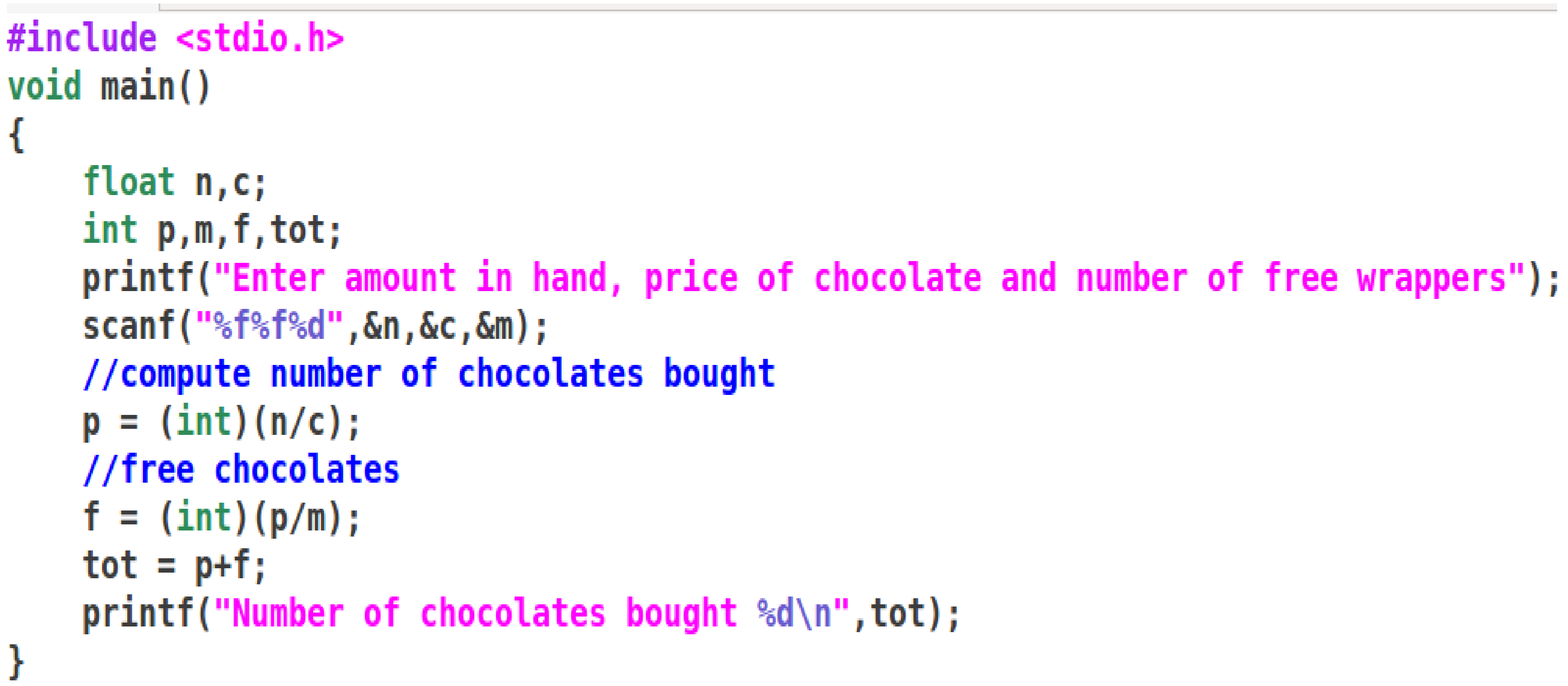
```
#include<stdio.h>
void main()
{
int a = 165;
int b = 100;
float c = (float)(a/b);
printf("%f",c);

}
```

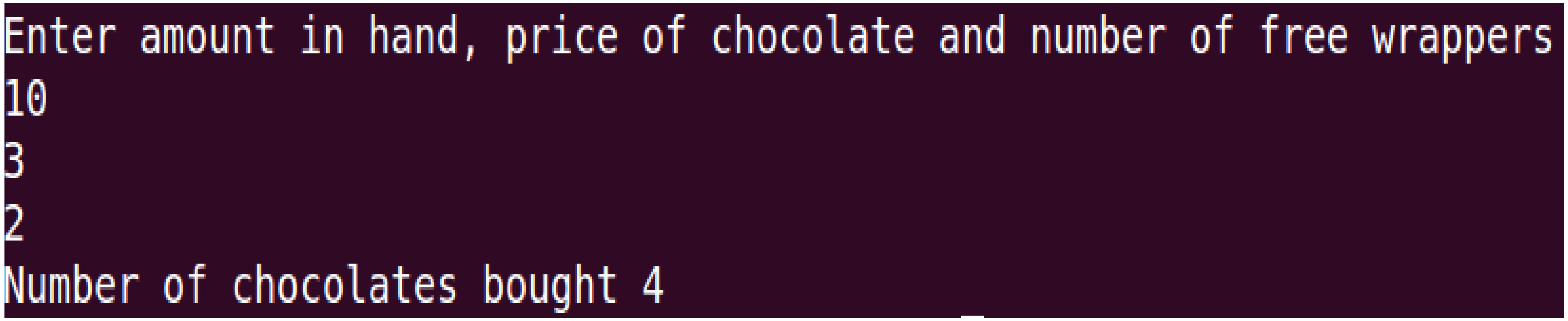
Output

```
1.000000
```

C Program fro Bob Problem



```c
#include <stdio.h>
void main()
{
    float n,c;
    int p,m,f,tot;
    printf("Enter amount in hand, price of chocolate and number of free wrappers");
    scanf("%f%f%d",&n,&c,&m);
    //compute number of chocolates bought
    p = (int)(n/c);
    //free chocolates
    f = (int)(p/m);
    tot = p+f;
    printf("Number of chocolates bought %d\n",tot);
}
```



```
Enter amount in hand, price of chocolate and number of free wrappers
10
3
2
Number of chocolates bought 4
```

Problem

ABC company Ltd. is interested to computerize the pay calculation of their employee in the form of Basic Pay, Dearness Allowance (DA) and House Rent Allowance (HRA). DA and HRA are calculated as certain % of Basic pay(For example, DA is 80% of Basic Pay, and HRA is 30% of Basic pay). They have the deduction in the salary as PF which is 12% of Basic pay. Propose a computerized solution for the above said problem.

Input : Basic Pay

Process : Calculate Salary

(Basic Pay + (Basic Pay * 0.8) + (Basic Pay * 0.3 - (Basic Pay * 0.12)

-----------allowances -------------- --- deductions----

Output : Salary

Problem

- Find the average runs scored by a batsman in 4 matches
- Area of a circle

Exercise

An university is setting up a new lab at their premises. Design an algorithm and write Python code to determine the approximate cost to be spent for setting up the lab. Cost for setting the lab is sum of cost of computers, cost of furnitures and labour cost. Use the following formulae for solving the problem:

Cost of computer = cost of one computer * number of computers

Cost of furniture = Number of tables * cost of one table + number of chairs * cost of one chair

Labour cost = number of hours worked * wages per hour