Introduction to C Programming

- C was originally developed in the 1970s, by Dennis Ritchie at Bell Telephone Laboratories, Inc.
- C is a High level , general –purpose structured programming language. Instructions of C consists of terms that are very closely same to algebraic expressions, consisting of certain English keywords such as if, else, for ,do and while
- C contains certain additional features that allows it to be used at a lower level , acting as bridge between machine language and the high level languages.
- This allows C to be used for system programming as well as for applications programming

Best C compilers

- Borland Turbo C
- Tiny C compiler
- GCC
- Clang

C Language Elements

- **Preprocessor directive:** a C program line beginning with # that provides an instruction to the preprocessor.

  Example:
- *#include, #define*

- **Preprocessor:** a system program that modifies a C program prior to its compilation

- **Library:** a collection of useful functions and symbols that may be accessed by a program

- Each library has a standard header file whose name ends with the

  symbols .h

A simple C Program

```
/*Converts distances from miles to kilometers*/

#include <stdio.h>                         /* printf, scanf definitions */
#define KMS_PER_MILE 1.609                 /* conversion constant */

 int main(void)
{
double miles;                              /* distance in miles*/
double kms;                                /* equivalent distance in kilometers */

printf("Enter the distance in miles> ");
scanf("%lf", &miles);
kms = KMS_PER_MILE * miles;                /* Convert the distance to kilometers. */
printf("That equals %f kilometers.\n", kms);   /*Display the distance in kilometers.*/
return (0);
}
```

Function main

- A function body has two parts:

➢ declarations

➢ executable statements

Declarations: declarations the part of a program that tells the compiler the names of memory cells in a program.

Executable statements: program lines that are converted to machine language instructions and executed by the computer.

main Function Definition

```
 int main(void)
 {
function body
 }
```

Example:

```
int main(void)
  {
 printf("Hello world\n");
return (0);
}
```

- Program execution begins with the main function.
- Braces enclose the main function body,
- It contains declarations and executable statements.
- The line int indicates that the main function returns an integer value (0) to the operating system when it finishes normal execution.
- The symbols (void) indicate that the main function receives no data from the operating system before it begins execution.

Comments in C

- Comments provide supplementary information and is used to explain code and make it more readable.
- It is ignored by the preprocessor and compiler.

- **Single-lined comment**

//Any text

- **Multi-lined comment**

beginning with /* and ending with */

Example:

/*This is an

example comment*/

Reserved word

- A word that has special meaning in C
- All the reserved words appear in lowercase;
- they have special meaning in C and cannot be used for other purposes.

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

IDENTIFIERS

Standard Identifiers

- standard identifier a word having special meaning but one that a programmer may redefine
- Standard identifiers have special meaning in C.

printf and *scanf* are names of operations defined in the standard input/output library.

User-defined identifiers

We choose our own identifiers to name memory cells that will hold data and program results and to name operations that we define

Example**:** *KMS_PER_MILE*

Rules

1. An identifier must consist only of letters, digits, and underscores.
2. An identifier cannot begin with a digit.
3. A C reserved word cannot be used as an identifier.
4. An identifier defined in a C standard library should not be redefined.

Valid Identifiers

Letter_1, letter_2, inches, cent, CENT_PER_INCH, Hello, variable

per_capita_meat_consumption_in_1980
per_capita_meat_consumption_in_1995

are identical for a C compiler

Because it considers only the first 31 characters to be significant.

Invalid Identifiers

double	reserved word
Int	reserved word
TWO*FOUR	character * not allowed
joe’s	character ' not allowed

Variable declaration and Data types in C

Variable: a name associated with a memory cell whose value can change.

Variable declarations: statements that communicate to the compiler the names of variables in the program and the kind of inform.tion stored in each variable.

Example:
```
double miles;     /* input - distance in miles. */
double kms;       /* output - distance in kilometers */
```

SYNTAX:

int   variable_list ;

double variable_list ;

char variable_list ;

EXAMPLES

```
int count, large;
double x, y, z;
char first_initial;
char ans;
```

Data Types

A data type specifies the type of data that a variable can store such as integer, floating, character, etc.

Data Type	Example of Data Type
Basic Data Type	Floating-point, integer, double, character.
Derived Data Type	Union, structure, array, etc.
Enumerated Data Type	Enums
Void Data Type	Empty Value
Bool Type	True or False

Data Type	Format Specifier	Minimal Range	Typical Bit Size
unsigned char	%c	0 to 255	8
char	%c	-127 to 127	8
signed char	%c	-127 to 127	8
int	%d, %i	-32,767 to 32,767	16 or 32
unsigned int	%u	0 to 65,535	16 or 32
signed int	%d, %i	Same as int	Same as int 16 or 32
short int	%hd	-32,767 to 32,767	16
unsigned short int	%hu	0 to 65,535	16
signed short int	%hd	Same as short int	16
long int	%ld, %li	-2,147,483,647 to 2,147,483,647	32
long long int	%lld, %lli	-(263 – 1) to 263 – 1 (It will be added by the C99 standard)	64
signed long int	%ld, %li	Same as long int	32
unsigned long int	%lu	0 to 4,294,967,295	32
unsigned long long int	%llu	264 – 1 (It will be added by the C99 standard)	64
float	%f	1E-37 to 1E+37 along with six digits of the precisions here	32
double	%lf	1E-37 to 1E+37 along with six digits of the precisions here	64
long double	%Lf	1E-37 to 1E+37 along with six digits of the precisions here	80

Data type int

- The **int** data type can be 4 bytes/ 2 bytes.
- **int** will be 2 bytes or 16 bits in the case of an environment that is 16-bit.
- int will be 4 bytes or 32 bits in case of an environment that is 32-bit.
- The int data type is used to represent integers in C.

Example:

```
int num;
```

Double data type

- A real number has an integral part and a fractional part that are separated by a decimal point.

- In C, the data type double is used to represent real numbers

- The **double** data type is 8 bytes or 64 bits.
- It is capable of storing values that are comparatively double the size of the bytes that the float data type can store.

- When looking at the 64 bits in total, the program has 1 bit for the sake of sign representation, the exponent uses 11 bits, and it uses the remaining 52 bits for the mantissa.

- We can store a real number in a type double variable.

Example

X = 3.45

```
double x;
```

Valid double constants	Invalid
2.1234	150 (no decimal point)
32.78	.12345e (missing exponent)
3.67E4	15e-0.3 (0.3 is invalid exponent)
15.0e-04	34,500.99 (comma is not allowed)

Internal formats of type int and double

- Integer

Binary number

Double

sign	exponent	mantissa
1 bit	11 bits	52 bits

The float data type is 4 bytes.

- The size of the float data type is basically 32 bits or 4 bytes.
- The float data type is single-precision in nature, and we use it for holding the decimal values.

Data Type char

The char data type is 1 byte or 8 bits.

Data type char represents an individual character value

—a letter, a digit, or a special symbol.

Each type char value is enclosed in apostrophes (single quotes) as shown here.

‘A’   'z’   '2’   '9’   '*’   ':’   '“’   ‘ ‘

Example

char x = ‘a’

Executable statements

Programs in Memory

Assignment Statements

Input/Output Operations and Functions

- All input/output operations in C are performed by special program units called input/output functions
- In C a function call is used to call or activate a function.

The printf Function

```
printf( format string, print list );
printf(format string );
printf("That equals %f kilometers.\n", kms);
```

Function name

Function argument: enclosed in parentheses following the function name provides information needed by the function format string in a call to printf, a string of characters enclosed in quotes ("), which specifies the form of the output line

Print list: in a call to printf, the variables or expressions whose values are displayed

Placeholder:

a symbol beginning with % in a format string that indicates where to display the output value.

Placeholder	Variable Type	Function Use
%c	char	printf/scanf
%d	Int	printf/scanf
%f	Double	printf
%lf	double	scanf

Multiple Placeholders

- Format strings can have multiple placeholders.
- If the print list of a printf call has several variables, the format string should contain the same number of placeholders.
- C matches variables with placeholders in left-to-right order.

Example

printf("Hi %c%c%c - your age is %d\n", letter_1, letter_2, letter_3, age);

displays a line

Hi EBK - your age is 35

Escape sequence

newline escape sequence:

the character sequence \n, which is used in a format string to terminate an output line.

Escape sequences

Escape sequence	**Character represented**
\a	Alert (bell, alarm)
\b	Backspace
\t	Horizontal tab
\n	New-line
\”	Double quotation mark
\?	Question mark
\\\\	Backslash
\’	Single quotation mark

scanf function

- Data can be stored in memory in two different ways:
  - ➢ either by assignment to a variable
  - ➢ by copying the data from an input device into a variable using a function like scanf

SYNTAX:

scanf(format string, input list);

EXAMPLE:

```
scanf(“%c”, &first_initial);
scanf("%c%d", &first_initial, &age);
```

The scanf function copies into memory data typed at the keyboard by the program user during program execution.

The format string is a quoted string of placeholders, one placeholder for each variable in the input list.

& first_initial gets the address of first_initial, and the value entered by the user is stored in that address.

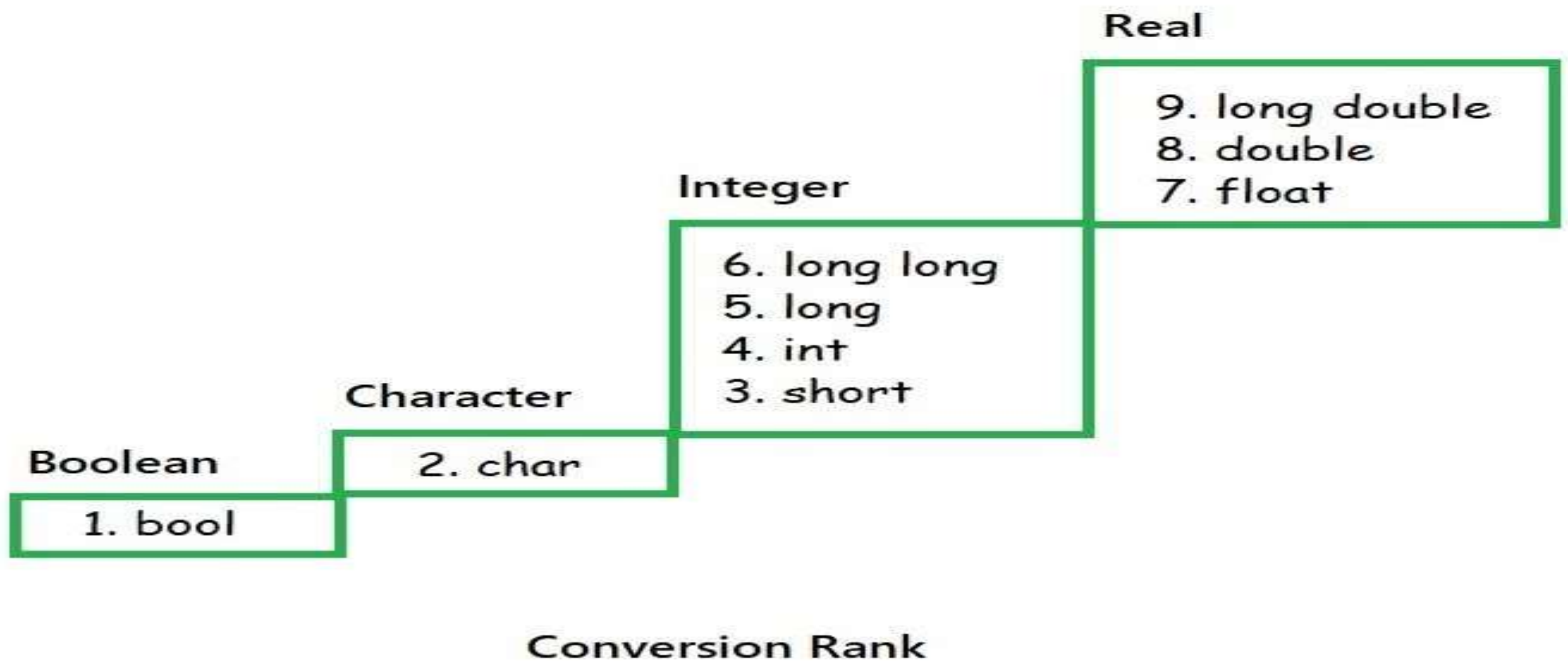
- Each int, double, or char variable in the input list is preceded by an ampersand (&).
- Commas are used to separate variable names.
- The order of the placeholders must correspond to the order of the variables in the input list.
- You must enter data in the same order as the variables in the input list. You should insert one or more blank characters or carriage returns between numeric items.
- If you plan to insert blanks or carriage returns between character data, you must include a blank in the format string before the %c placeholder.

Type conversion

1. Implicit Type Conversion (Automatic type conversion)

- Done by the compiler on its own, without any external trigger from the user.
- Generally takes place when in an expression more than one data type is present. In such condition type conversion (type promotion) takes place to avoid loss of data.
- All the data types of the variables are upgraded to the data type of the variable with largest data type.

Hierarchy of data types



Example

```
#include<stdio.h>
int main(){
int x = 5;
float z;
char y = ‘a’;
X = x + y;
Z = x + 2.0;
printf(“value of x is %f \n value of  z is %d”, x, z);
 return 0;
}
```

2. Explicit type conversion (Type casting)

It is user defined.

A data type is converted into another data type by a programmer using casting operator.

In type casting, the destination data type may be smaller than the source data type when converting the data type into another data type.

Syntax

(type)expression

Example

```
main(){
double x = 3.0;
sum = (int)x + 1;
printf(“sum = %d”, sum);
return 0;
}
```

Formatting numbers in program output

Formatting Values of Type int

Add a number between the **%** and the d of the **%d** placeholder in the printf format string.

This number specifies the field width —the number of columns to use for the display of the value.

It is right justified.

Formatting Values of Type double

we must indicate both the total field width needed and the number of decimal places desired.

The total field width should be large enough to accommodate all digits before and after the decimal point.

A display column is also considered for the decimal point and for the minus sign if the number can be negative.

Form of the format string placeholder: % n. mf

where n is total field width,

m is the desired number of decimal places.

A placeholder such as **% . mf** specify only the number of decimal places, the value will be printed with no leading blanks.

Interactive mode, batch mode and data files

Interactive mode: a mode of program execution in which the user responds to prompts by entering (typing in) data.

Batch mode: a mode of program execution in which the program scans its data from a previously prepared data file.

Input Redirection

Output Redirection

Few questions

Programming 1.

Write the #define preprocessor directive and declarations for a program that has a constant macro for PI (3.14159) and variables radius , area , and circumf declared as double , variable num_circ as an int , and variable circ_name as a char .

Programming 2.

C Program to Print an Integer (Entered by the User)

Print the ascii values of the characters entered.