Unit IV- Part 01 (Basic Constructs and fundamentals)

Lecture Slide

AS2023

Objectives

By the end of this session, students will be able to:

- Explain importance of C
- Explain basic structure of C program
- Explain various data types
- Define identifiers and its rule
- Explain C tokens
- Assign values to a variable
- Explain data type conversion
- Write a simple C Program

Introduction to C

- Structured, high level machine independent programming language
- Allows software developer to develop programs without worrying about the hardware platforms where they will be implemented
- Developed by Dennis Ritchie in 1972 at Bell Laboratories
- Evolved from ALGOL, BCPL & B
- Today C is running under a variety of operating systems and hardware platforms

Importance of C

- Increasing popularity probably due to its many desirable qualities
- Suitable for writing both system and business packages
- Programs written in C are fast and efficient
- Highly portable
- Well suited for structured programming
- Ability to extend itself

Basic Structure of C

Include header file section

Global Declaration Section

```
/*comments*/
main(){
    Declaration part
    Executable part
}
```

User-defined Functions{
Statement
}

- C program depends upon some header files for function definition that are used in the program
- Example: #include<stdio.h>

Basic Structure of C

Include header file section

Global Declaration Section

```
main(){
   Declaration part
   Executable part
}
```

User-defined Functions{
Statement
}

- Declares some variables that are used in more than one function
- These must be declared outside of all the functions

Basic Structure of C

Include header file section

Global Declaration Section

```
/*comments*/
main(){
   Declaration part
   Executable part
}
```

User-defined Functions{
Statement
}

COMMENTS

- To understand the flow of program
- Useful for documentation and clarity of the program
- Compiler doesn't execute comments
- Eg. /*this is my first program*/

Basic Structure of C

Include header file section

Global Declaration Section

```
main(){
   Declaration part
   Executable part
}
```

User-defined Functions{
Statement
}

- Every C program must have main() function
- Starting point of every C program
- Execution begins from main()
- Program execution starts from opening brace **{** and ends with closing brace **}**
- **Declaration part** and **Executable** are written between these two braces

Basic Structure of C

Include header file section

Global Declaration Section

```
main(){
  Declaration part
  Executable part
}
```

User-defined Functions{
Statement
}

- Functions defined by user is called user-defined functions
- User-defined functions can be defined before or after the **main()** function

Programming Rules

- A programmer while writing a program should follow certain rules
  1. All statements should be written in lowercase letters. Uppercase letters are only used for symbolic constants
  2. Blank spaces may be inserted between the words but not during the declaring a variable, keyword, constants and functions
  3. It is not necessary to fix the position of statement in the program. Users can also write one or more statements in one line separating them with a semi colon (;).

```
eg. A=b+c;
    e=f+g;
or
    A=b+c;   e=f+g;
```

  4. The opening and closing braces should be balanced

Sample C Program

```
#include<stdio.h>
int main(){
  printf("Hello World, I am new to 
C");
  return 0;
}
```

Executing the Program

- The following steps are essential in executing a program in C
  - Creation of program
  - Compilation and linking of a program
  - Executing the program

- **Homework:** Elaborate on the execution of the program

Constants, Variables & data types

- Programming language is designed to process certain kinds of data consisting of numbers, characters and to provide useful output known as information
- Instructions are formed using certain symbols and words according to some rigid rules known as ***syntax***

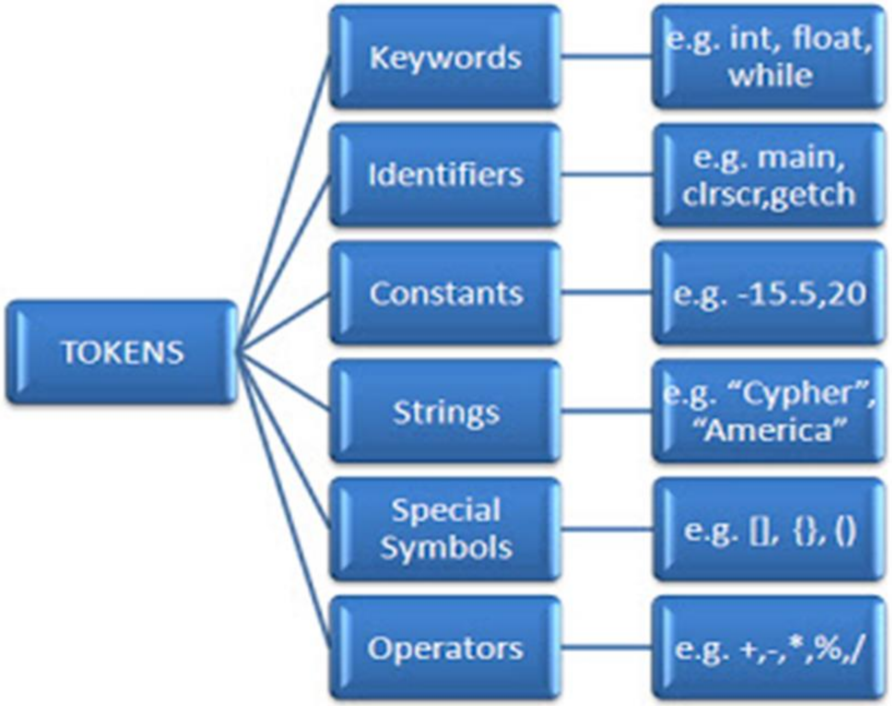
Character Set

- In C, the characters used to form words, numbers and expressions are grouped into following categories
  - Letters
  - Digits
  - Special characters
  - White spaces

- Homework: Write examples each for all the categories.

C Tokens

- Smallest individual units are known as C Tokens
- C has six types of token



- Homework: Write down all the keywords available in C.

Identifiers

- Identifiers refers to the names of variables, functions and arrays.
- These are user defined names and consist of a sequences of letters and digits
- **Rules for Identifiers:**
  - First character must be an alphabet (or underscore)
  - Must consist of only letters, digits or underscore
  - Only first 31 characters are significant
  - Cannot use a keyword
  - Must not contain white space

Variables

- Data name that is used to store a data value
- Variables may take different values at different times during the execution
- A variable is chosen by a programmer in a meaningful way
  - Eg. TOTAL, average etc
- Variable names may consist of letters, digits and underscore character subject to following condition:
  - They mist begin with letter or underscore.
  - It should not be normally more than 8 characters long
  - Upper case and lower case are significant
  - It should not be a keyword
  - White space is not allowed
- Eamples: **John, Value** are **valid** while **char, int, (area)** are invalid

Data Types

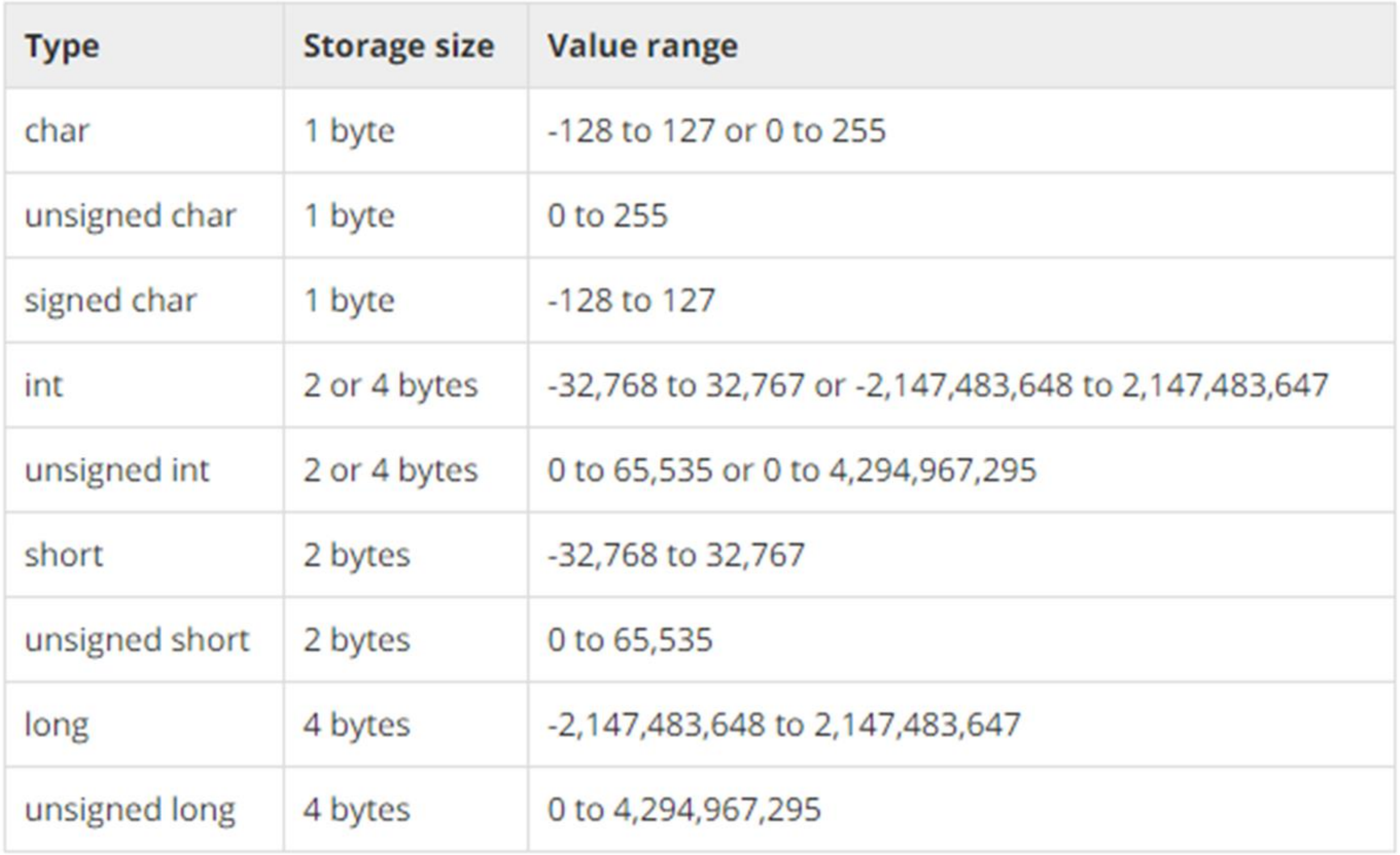
- C is rich in its data types
- Storage representation & machine instruction to handle constants differs from one machine to other
- To enable programmer to select the appropriate data type according to the need of application.
- All C compilers support five fundamental data types
  - Integer (int)
  - Character (char)
  - Floating point (float)
  - double precision floating point (double)
  - void

Integer

- Integers are whole numbers with range of values supported by a particular machine
- Generally occupies the size of word (16 or 32 bits) & differs from machine
- So the integer size too
- **short int**, **int** & **long int** are provided to control over the range of numbers and storage space in both signed and unsigned forms
- **long** and **unsigned** integers are declared to increase the range of values

Integer

- Size and range of data types on a 16bit machine



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

Void Type

- Has no values
- Usually used to specify the type of function
- The function is said to be void when it does not return any values to the calling function
- Can also play the role of generic type that it can represent any of the other standard types

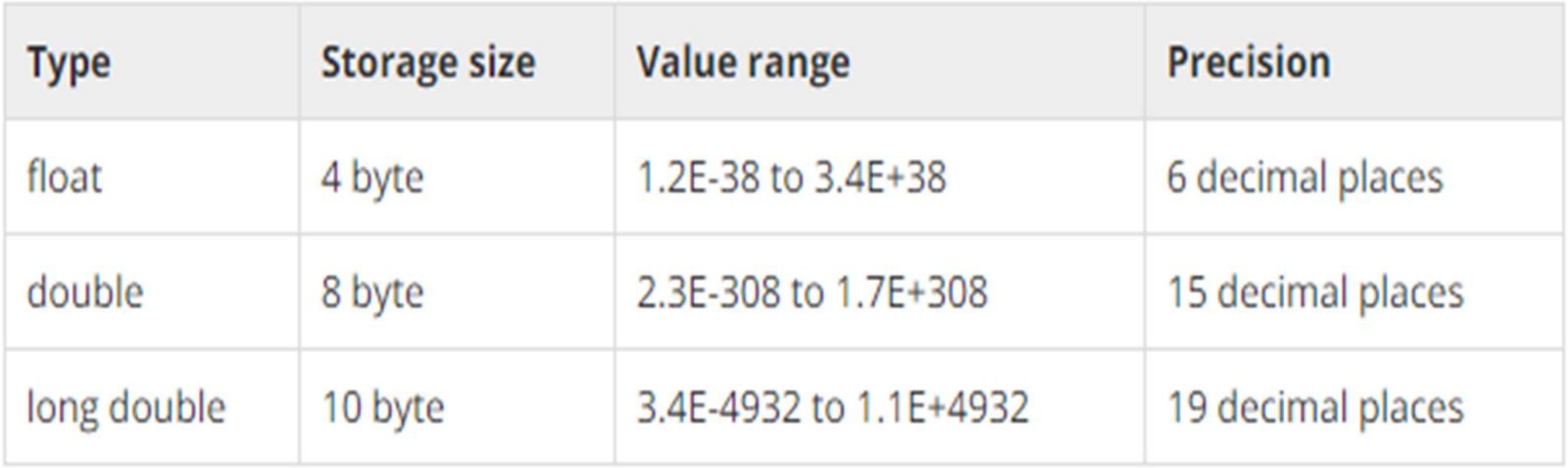
Character Type

- A single character can be defined as character (char) data type
- Characters are usually stored in 8bits
- The qualifier **signed** and **unsigned** can be explicitly applied to char
- While unsigned have values from 0-255 and signed chars have values from -128 to 127

Float Type

- Floating point (real) numbers are stored in 32bits or 4bytes
- Floating point numbers are defined in C by the keyword **float**
- This is also called as single precision number
- When the accuracy provided by a float is not sufficient, the type **double** can be used to define the number
- A double data type number uses 64bits or 8bytes
- This is also called as double precision number
- To extend the precision further, long double is used which uses 80bits or 10bytes

Float type Cont..



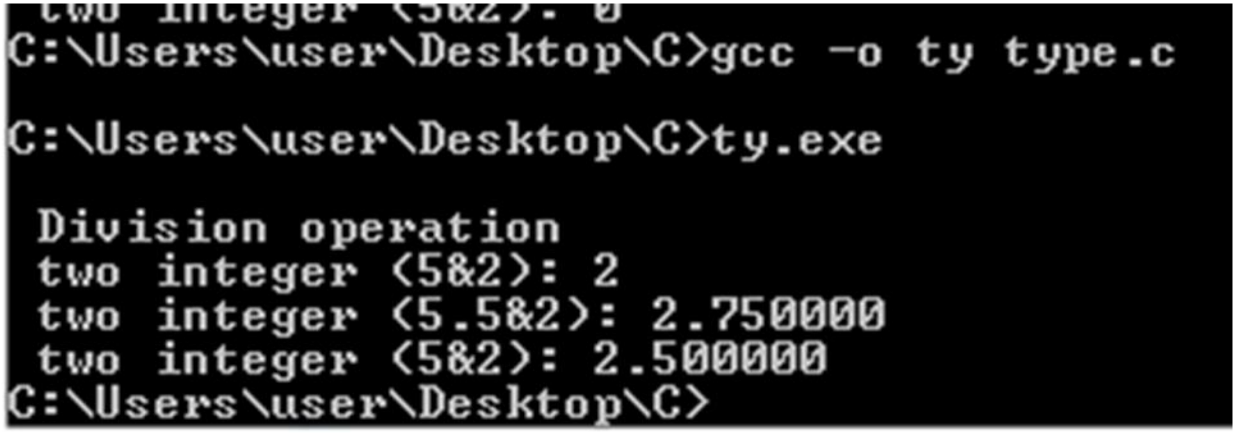
Type	Storage size	Value range	Precision
float	4 byte	1.2E-38 to 3.4E+38	6 decimal places
double	8 byte	2.3E-308 to 1.7E+308	15 decimal places
long double	10 byte	3.4E-4932 to 1.1E+4932	19 decimal places

Data types and their keywords

Data type	Keyword equivalent
Character	char
Unsigned character	unsigned char
Signed character	signed char
Singed integer	signed int (or int)
Signed short integer	signed short int (or short int or short)
Signed long integer	signed long int (or long int or long)
unsinged integer	usigned int (or int)
unsigned short integer	unsigned short int (or short int or short)
unsigned long integer	unsigned long int (or long int or long)
Floating point	float
Double precision floating point	double
Extended double precision floating point	Long double

Type conversion

```
#include<stdio.h>
int main(){
   printf("\n Division operation result");
   printf("\n two integer (5&2): %d", 5/2);
   printf("\n two integer (5.5&2): %f", 5.5/2)
   printf("\n two integer (5&2): %f", (float)5/2);
}
```



```
 two integer (5&2): 0
C:\Users\user\Desktop\C>gcc -o ty type.c

C:\Users\user\Desktop\C>ty.exe

 Division operation
 two integer (5&2): 2
 two integer (5.5&2): 2.750000
 two integer (5&2): 2.500000
C:\Users\user\Desktop\C>
```

Declaration of variables

- After designing the suitable variable names, we must declare them to the compiler
- Declaration does two things
  - It tells the compiler to what the variable name is
  - It specifies data type of data the variable will hold
- The declaration of variable must be done before they are used in the program
- Syntax for declaring variable

  Data-type variable-name;

Assigning values to variables

- Variables are created for use in program statement

  Eg. value = amount + inrate * amount;
- The process would be possible only if the variables are assigned with values
- The variable value is called target variable
- Variable used in expression must be assigned values before they are encountered in the program
- Values can be assigned to a variable using the assignment operator “=”

  Variable_name = constant;

Reading data from user keyboard

- Another way of assigning value to variable is to input data through keyboard using **scanf** function.
- Is a general function in C
- General format of scanf

```
scanf("control string", &variable_name);
```

- ***scanf*** provides an interactive features and makes program user friendly

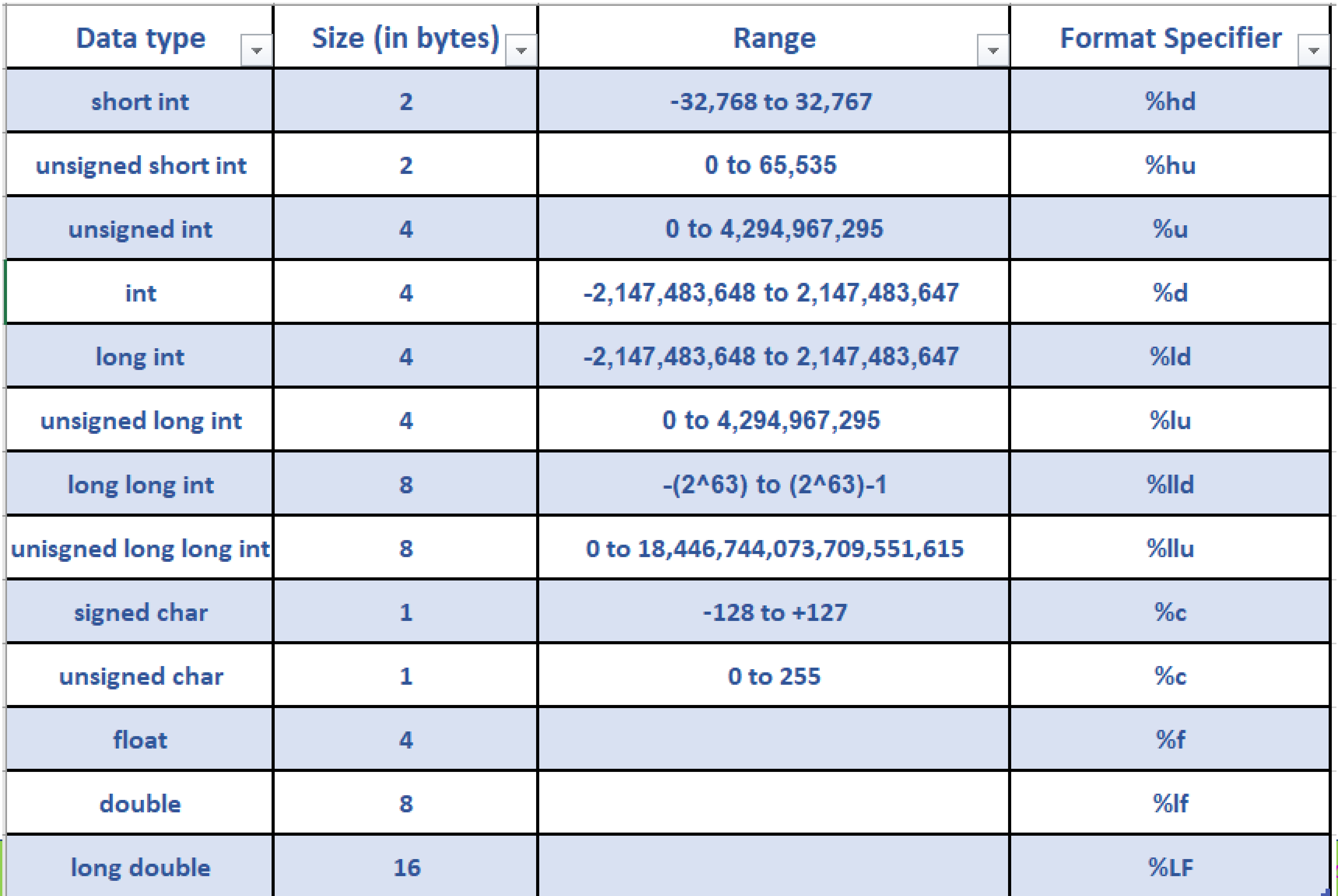
ESCAPE SEQUENCE CHARACTERS

- C supports some special escape sequence characters that are used to do special tasks
- These are also called as '*Backslash characters*'

ESCAPE SEQUENCE CHARACTERS

Character Constant	Meaning
\n	New line (Line break)
\b	Backspace
\t	Horizontal Tab
\f	Form feed
\a	Alert (alerts a bell)
\r	Carriage Return
\v	Vertical Tab
\?	Question Mark
\'	Single Quote
\"	Double Quote
\\\\	Backslash
\0	Null

Data types and format specifier



Data type	Size (in bytes)	Range	Format Specifier
short int	2	-32,768 to 32,767	%hd
unsigned short int	2	0 to 65,535	%hu
unsigned int	4	0 to 4,294,967,295	%u
int	4	-2,147,483,648 to 2,147,483,647	%d
long int	4	-2,147,483,648 to 2,147,483,647	%ld
unsigned long int	4	0 to 4,294,967,295	%lu
long long int	8	-(2^63) to (2^63)-1	%lld
unisgned long long int	8	0 to 18,446,744,073,709,551,615	%llu
signed char	1	-128 to +127	%c
unsigned char	1	0 to 255	%c
float	4		%f
double	8		%lf
long double	16		%LF

Writing C program

Home Assignment

- List down all other data types and format specifiers

Writing C program

- Write a program to print the pattern

```
* * * *
*
* * * *
```

- Write a program to compute the product of two number
  - Input (compile time initialization)
  - Input (Run time initialization)

Thank you