CSE 1201 "Structured Programming Language"

Constants, Variables, and Data Types

A. H. M. Saiful Islam
Associate Professor, Dept. of CSE, NDUB

Constants, Variables, and Data Types

- Like any other language, C has its own vocabulary and grammar.
- Here the concepts of Constants, Variables, and Data Types will be discussed.

Character Set

The characters that can be used to form words, numbers, and expressions upon which the program is run.

The characters in C are grouped into the following categories:

- Letters
- Digits
- Special characters
- White spaces

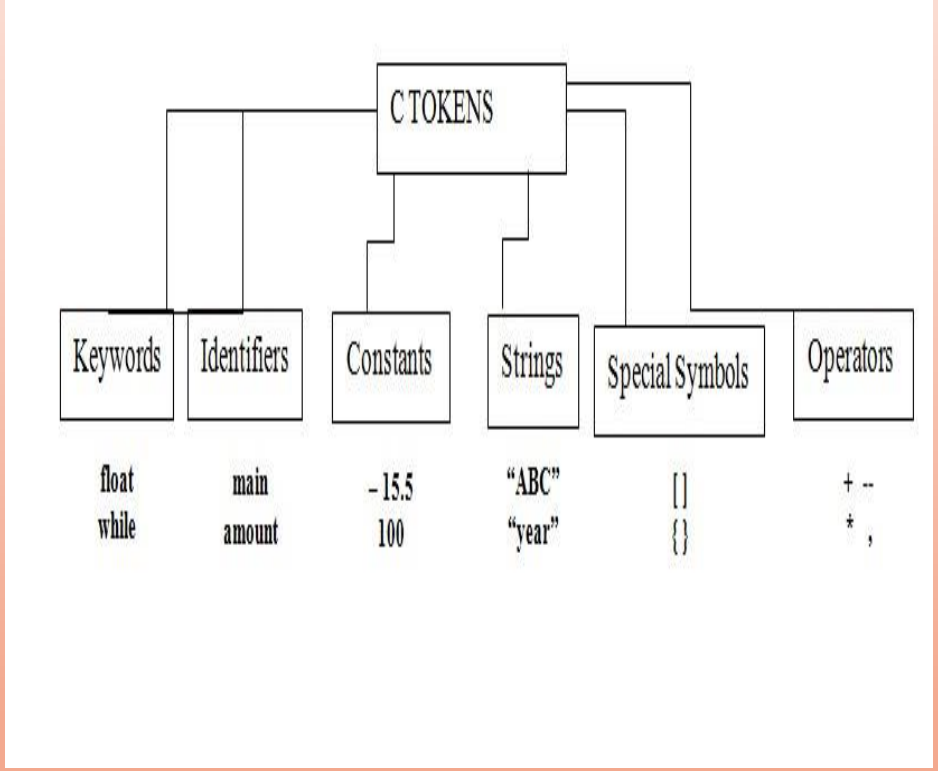
Letters, Digits, Special characters, White spaces

- **Letters:**
- Uppercase: A….Z Lowercase: a…..z
- **Digits:**
- All decimal digits 0—9
- **Special characters:**
- , comma
- . period
- ; semicolon
- / slash
- \ backslash
- $ dollar sign
- % percent sign

C tokens

- In a passage of text individual words and punctuation marks are called tokens.
- Similarly in a C program the smallest individual units are known as C tokens.
- C has six types of tokens.

- C programs are written using these tokens and the syntax of the language.



Keywords and Identifiers

- Every C word is classified as either a keyword or an identifier.
- All keywords have fixed meanings and these meanings cannot be changed.
- All keywords must be written in lowercase.

Keywords

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

Identifiers

- Identifiers refer to the names of variables, functions and arrays.
- These are user-defined names and consists of a sequence of letters, and digits, with a letter or underscore or dollar as a first character.

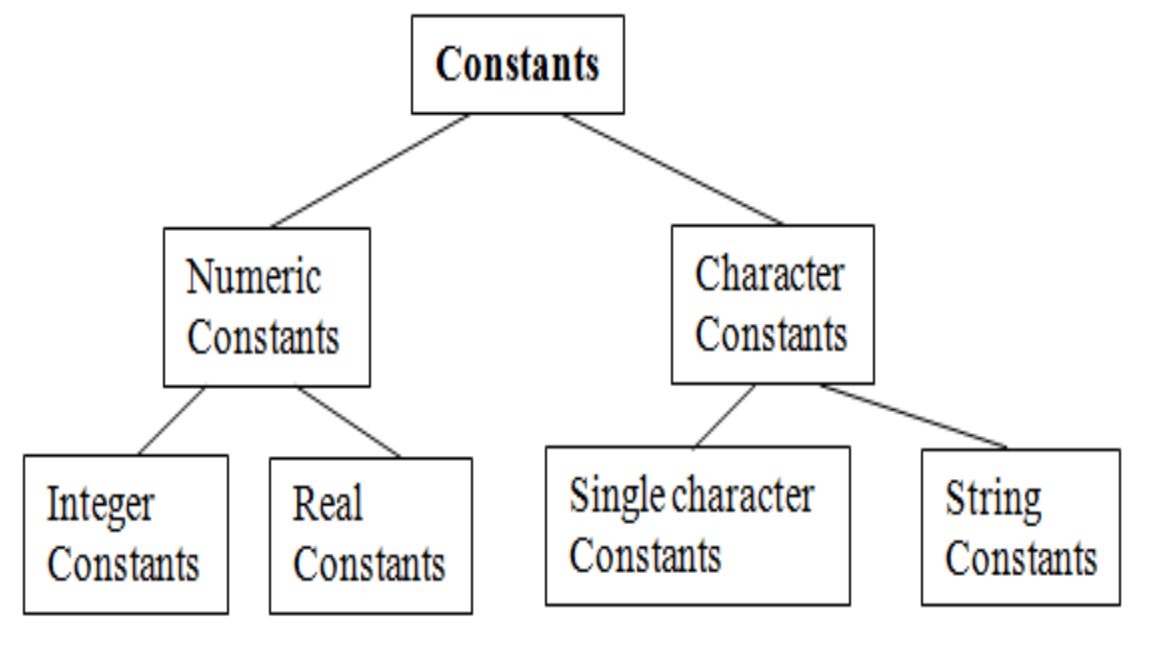
- Both uppercase and lowercase letters are commonly used. The underscore character is also permitted in identifiers.

Rules for Identifiers

- First character must be an alphabet (or underscore or dollar)
- Must consist of only letters, digits, underscore and dollar symbol
- Cannot use a keyword
- Must not contain white space.

Constants

Constants in C refer to fixed values that do not change during the execution of a program.



Integer Constants

An integer constant refers to a sequence of digits.

There are three types of integers, namely decimal, octal and hexadecimal integers.

Decimal integer consists of a set of digits, 0 through 9, preceded by an optional – or + sign.

Valid examples of decimal integers constants are:

123 -321 0 654321 +78

Embedded spaces, commas, and non-digit characters are not permitted between digits.

For example,

15 750 20,000 $1000

are illegal numbers.

Integer Constants (Cont.)

An **octal integer constant** consists of any combination of digits from the set 0 through 7, with a leading 0. Some examples of octal integer are:
037 0 0435 0551

A sequence of digits preceded by 0x or 0X is considered as hexadecimal integer. They may also include alphabets A through F or a through f. The letter A through F represent the numbers 10 through 15.

Valid examples:
0X2 Ox9F 0Xbcd 0x

We rarely use octal and hexadecimal numbers in programming.

The largest integer value that can be stored is machine dependent. It is 32767 on 16-bit machines and 2,147,483,647 on 32-bit machines.

Real Constant

Integer numbers are inadequate to represent quantities that vary continuously, such as distances, heights, temperatures, prices, and so on.

These quantities are represented by numbers containing fractional parts like 17.548.

Such numbers are called real (or floating point) constants.
Example of real constant:

0.0083 -0.75 435.36 +247.0

These numbers are shown in decimal notation, having a whole number followed by a decimal point and the fractional part.
It is possible to omit digits before the decimal point or digits after the decimal point. That is,
215. .95 -.71 +.5
are all valid real numbers.

Real Constants

The value 215.65 may be written as 2.1565e2 in exponential notation.

e2 means multiply by 10^2.

The general form is :

mantissa e exponent

the mantissa is either a real number expressed in decimal notation or an integer.

The exponent is an integer number with an optional plus or minus sign. The letter e separating the mantissa and the exponent can be written in either lowercase or uppercase. Since the exponent causes the decimal point to 'float' this notation is said to represent a real number in floating point form.

Examples of legal floating point constants are:

0.65e4 12e-2 1.5e+5 3.18E3 - 1.2E-1

Embedded white space is not allowed.

Some examples of valid and invalid numeric constants are given below.

Constant	Valid?	Remarks
698354L	Yes	Represents long integer
25,000	No	Comma is not allowed
+5.0E3	Yes	C supports unary plus
3.5e-5	Yes	
7.1e 4	No	No white space is permitted
-4.5e-2	Yes	
1.5E+2.5	No	Exponent must be an integer
$255	No	$ symbol is not permitted
0X7B	Yes	Hexadecimal Integer

Single character Constants

- A single character constant contains a single character enclosed within a pair of single quote marks.

- Example of character constants are : ‘5’ ‘X’ ‘; ’ ‘ ’

Note that the character constant ‘5’ is not the same as the number 5. The last constant is a blank space.

- Character constants have integer values known as ASCII values. For example, the statement printf (“%d”, ‘a’); would print the number 97, the ASCII value of letter a. similarly, the statement printf (“%c”, 97); would output the letter ‘a’.
- Since each character constant represents an integer value , it is also possible to perform arithmetic operations on character constants.

String Constants

- A string constant is a sequence of characters enclosed in double quotes. The characters may be letters,
- Numbers, special characters and blank space. Examples are:

  “Hello” ”1987” ”WELL DONE”

  ”?…!” ”5+3” ”X”

- Remember that a character constant (e.g., ‘X’)is not equivalent to single character string constant(e.g., “X”) .

- Further, a single character string constant does not have an equivalent integer value while a character constant has an integer value.

Backslash character Constants

- C supports some special backslash character constants that are used in output functions.
- For example, the symbol '\n' stands for newline character. A list of such backslash character constants is given here.

Constant	Meaning
'\a'	audible alert
'\b'	Back space
'\f'	Form feed
'\n'	New line
'\r'	Carriage return
'\t'	horizontal tab
'\v'	Vertical tab
'\''	single quote
'\"'	Double quote
'\?'	Question mark
'\\'	Backslash
'\0'	Null

VARIABLES

- A variable is a data name that may be used to store a data value. Unlike constants that remain unchanged during the execution of a program, a variable may take different values at different times during execution.

- A variable name can be chosen by the programmer in a meaningful way so as to reflect its function or nature in the program.

- Some examples of such names are:

- Average
- height
- total
- Counter_1
- Class_strength

VARIABLES

Variable names may consist of letters, digits and the underscore (_) character subject to the following conditions:

- They must begin with a letter. Some systems permit underscore as the first character.
- Length should not be more than eight characters
- Uppercase and lowercase letters are significant. The variables Total and total is not the same variable.
- It must not be a keyword
- White space is not allowed

Some examples of valid variable names are:

John	Value	T_raise	Delhi	x1
	ph_value mark	sum1	distance	

- **Invalid examples**

123	(area)	%	25th

VARIABLES

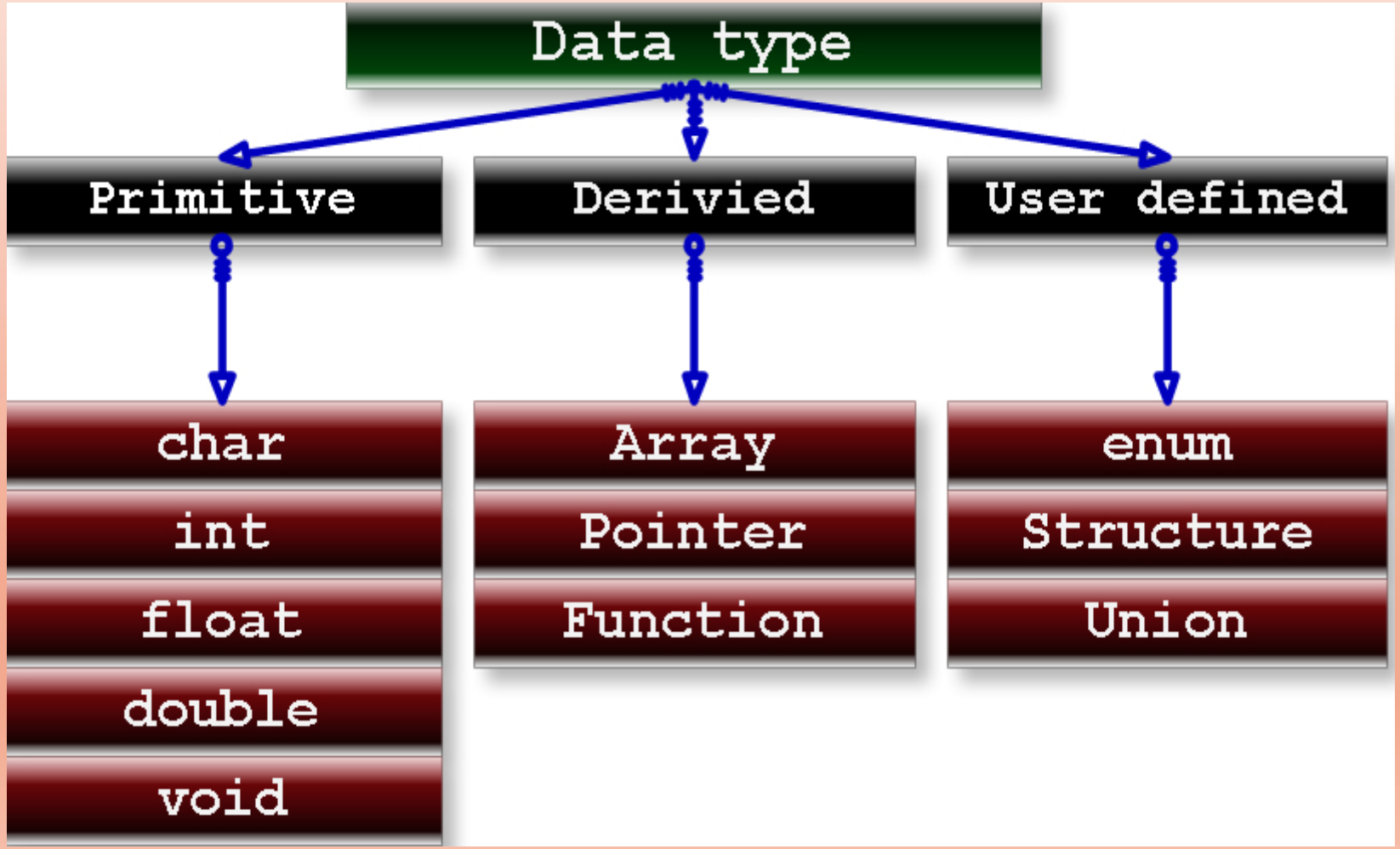
Variable name	Valid?	Remarks
First_tag	Yes	
char	No	char is a keyword
Price$	Yes	
group one	No	Blank space is not permitted
average_number	Yes	
int_type	Yes	Keyword may be a part of the name

Data Types

- There are 4 data types in C language. They are:-
- **int** – This data type is used to define an integer number (-….-3,-2,-1,0,1,2,3….). A single integer occupies 2 bytes.
- **char** – Used to define characters. A single character occupy 1 byte.
- **float** – Used to define floating point numbers (single precision). Occupies 4 bytes.
- **double** – Used for double precision floating point numbers(double precision). Occupies 8 bytes.
- Note:- Single precision and Double precision basically differs in the number of digits represented after the decimal point. Double precision number will represent more digits after the decimal point than a single precision number. Example:- Single precision – 32.75 and double precision - 32.7543

Data Types

Data Types can also be classified as shown in the image below – Primitive, Derived and User Defined.



Data Type Qualifiers

- Each of these data type has got qualifiers. The purpose of a qualifier is to manipulate the range of a particular data type or its size.

- **The 4 qualifiers in C are long, short, signed and unsigned.**

- First two **long and short** are called **size qualifiers** and the other two **signed and unsigned** are called **sign qualifiers**.

Data Type Qualifiers

Example:

- **int** – when declared normally is of 2 bytes.
- If it is declared as **unsigned int** – then its range is from 0 to 65535.
- In other case, if it is declared as **signed int** – then its range is from (-32767 to 32768).
- In the case of signed int, one bit (MSB) is used to store the sign of the integer +/-.

- This basically means the programmer will not be able to display/store a number higher than 65535 using unsigned int. Similarly it is not possible to manipulate a number beyond -32767 or +327678 using signed int.

Data Type Qualifiers

- The qualifiers long and short are used to increase storage size of the data type.

- **Example:** Integer data type int is normally 2 byte. If you declare it as long int – then its size will increase from 2 bytes to 4 bytes. Similarly if you declare it as short int – its size will reduce from 2 bytes to 1 byte.

Representation of integer constants on a 16-bit computer.

- **Program**

```
main()
{
   printf("Integer values\n\n");
   printf("%d %d %d\n", 32767,32767+1,32767+10);
   printf("\n");
   printf("Long integer values\n\n");
   printf("%ld %ld %ld\n",
    32767L,32767L+1L,32767L+10L);
}
```

- **Output**

```
Integer values

32767 -32768 -32759

Long integer values

32767 32768 32777
```

The output shows that the integer values larger than 32767 are not properly stored on a 16-bit machine. However, when they are qualified as long integer (by appending L), the values are correctly stored.

Output will be shown correctly on 32 bit Computer

Data types

- The table here describes all data types and the most commonly used qualifier combinations – with its size, range and format specifier.

Keyword	Format Specifier	Size	Date Range
	DataTypesSummary.pdf		

Data Type

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

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