Structured Programming Language

Lecture 3

Outline

- C Basics
- Character Set in C
- C Basic Syntax
- Variables
- Data Types
- Variable Declaration
- Lvalue and Rvalue

C Basics Cont…

```
/*
 * Author Name:CSE-48
 * Source file name: hello.c
 * Comments: My first program in C.
 */
```

Comment

```
#include <stdio.h>
```

Preprocessor

```
/*the main function*/
```

Comment

```
int main ()
{
  /*print the phrase */
  printf("Hello, CSE 48!!");

   return 0;
}
```

Main Function

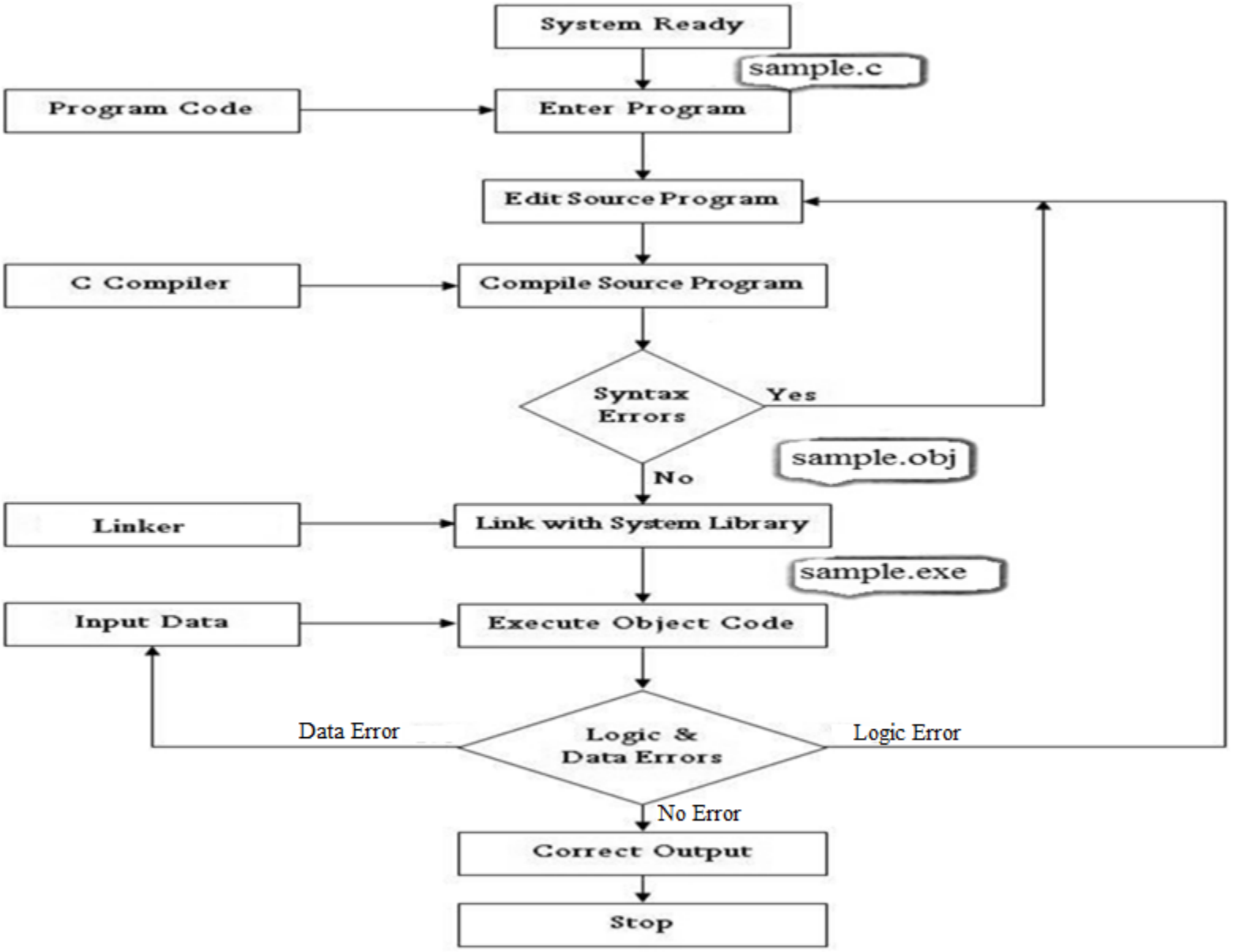
Comment

Function

C Basics Cont…

- Let us take a look at the various parts of the above program −
  - The first line of the program *#include <stdio.h>* is a preprocessor command, which tells a C compiler to include stdio.h file before going to actual compilation.
  - The next line *int main()* is the main function where the program execution begins.
  - The next line **/*this is a comment*/** will be ignored by the compiler and it has been put to add additional comments in the program. So such lines are called comments in the program.
  - The next line *printf("Hello, CSE 48!!");* is another function available in C which causes the message "Hello, CSE 48!!" to be displayed on the screen.
  - The next line **return 0;** terminates the main() function and returns the value 0.

C Basics Cont…



Process of compiling and running a C program.

Character Set in C

- **Character set**:-
  - The character set is the fundamental raw material of any language and they are used to represent information.
  - Like natural languages, computer language will also have well defined character set, which is useful to build the programs.

Character Set in C Cont…

- The characters in C are grouped into the following **two** categories:
  - **Source character set**
    - Alphabets
    - Digits
    - Special Characters
  - **Execution character set**
    - Escape Sequence

Character Set in C Cont…

Types	Character Set
Lowercase Letters	a-z
Uppercase Letters	A to Z
Digits	0-9
Special Characters	!@#$%^&*
White Spaces	Tab Or New line Or Space

- C is case sensitive which means the small letters and capital letters are distinct.

Character Set in C

- Special Characters

Symbol	Meaning	Symbol	Meaning	Symbol	Meaning	Symbol	Meaning
~	Tilde	`	Apostrophe	(	Left parenthesis	"	Quotation mark
!	Exclamation mark	-	Minus sign	)	Right parenthesis	;	Semicolon
#	Number sign	=	Equal to sign	_	Underscore	<	Opening angle bracket
$	Dollar sign	{	Left brace	+	Plus sign	>	Closing angle bracket
%	Percent sign	}	Right brace	|	Vertical bar	?	Question mark
^	Caret	[	Left bracket	\	Backslash	,	Comma
&	Ampersand	]	Right bracket	/	Slash	.	Period
*	Asterisk	:	Colon				

A Simple Program: Printing a Line of Text

Escape Sequence	Description
`\n`	Newline. Position the screen cursor to the beginning of the next line.
`\t`	Horizontal tab. Move the screen cursor to the next tab stop.
`\r`	Carriage return. Position the screen cursor to the beginning of the current line; do not advance to the next line.
`\a`	Alert. Sound the system bell.
`\\`	Backslash. Used to print a backslash character.
`\"`	Double quote. Used to print a double quote character.

Basic Syntax

- You have seen the basic structure of a C program
- The basic building blocks of C are:
  - Tokens
  - Semicolons
  - Comments
  - Identifiers
  - Keywords
  - Whitespace

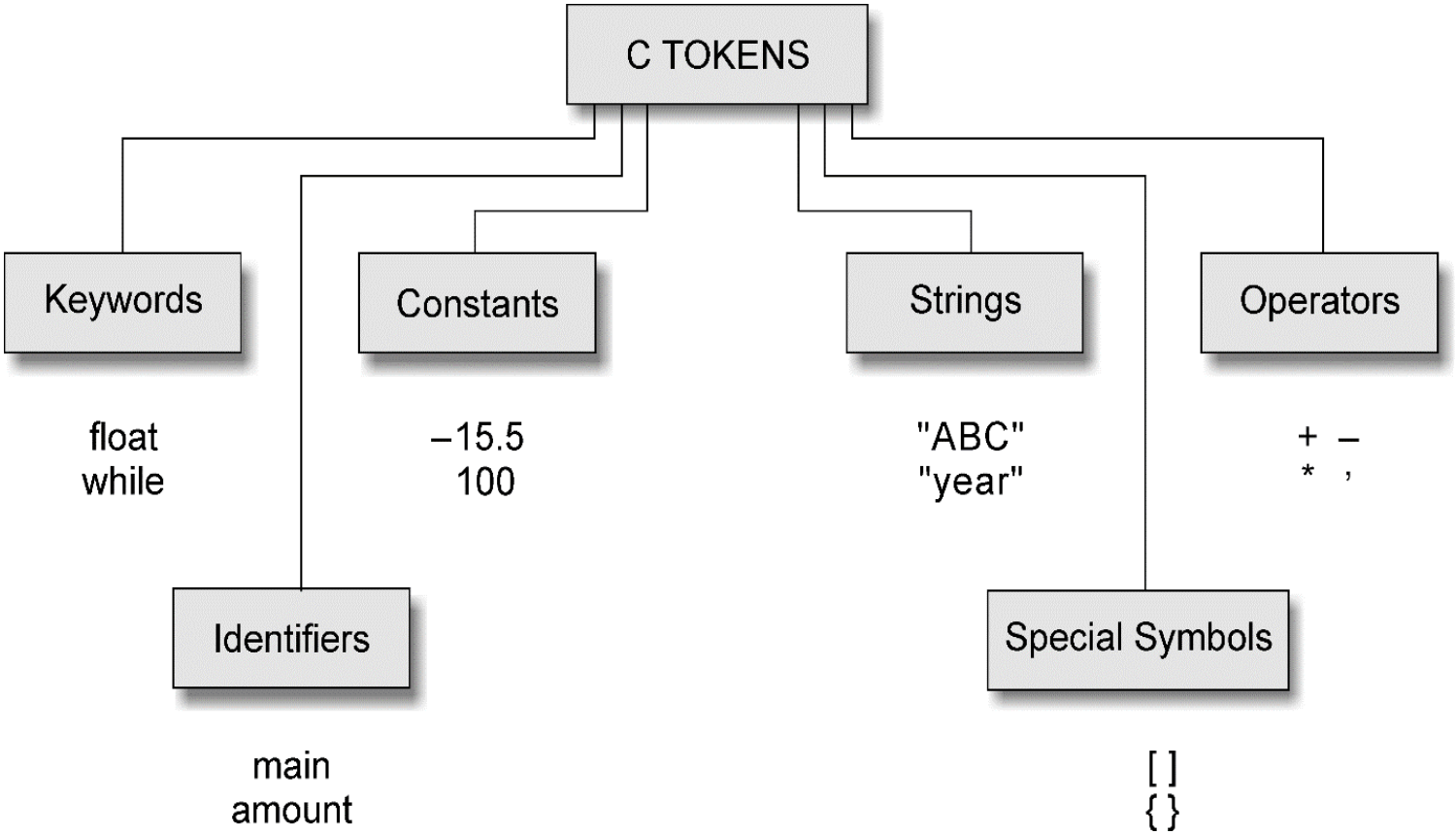
Basic Syntax: Tokens

- In a passage of text, individual words and punctuation marks are called *tokens*.
- A C program consists of various tokens and a token is either a *keyword*, an *identifier*, an *operator*, a *constant*, a *string literal*, or a *symbol*.
- For example, the following C statement consists of five tokens –

```
printf("Hello, CSE 48!!");
```

```
1.printf
2.(
3."Hello, CSE 48!!"
4.)
5.;
```

Basic Syntax: Tokens Cont.



Basic Syntax: Semicolons

- In a C program, the semicolon(;) is a statement terminator.
- That is, each individual statement must be ended with a semicolon.
- It indicates the end of one logical entity.

- Given below are two different statements −

```
printf("Hello, CSE 48!!");
return 0;
```

Basic Syntax: Comments

- Comments are like helping text in your C program and they are ignored by the compiler.
- Two Types
  - Block Comment
  - Single Line Comment (Only Supported in C99 or later)
- Block Comment *start* with /* and *terminate* with the characters */ as shown below −

```
/* my first program
   in C */
```

- You cannot have comments within comments and they do not occur within a string or character literals.

Basic Syntax: Comments

- Single Line Comment (C99 or later)*start* with '//' and continue until the end of the line.

- **// my first program in C**

- Multiple // operator can be used in a single comment.

Basic Syntax: Identifier

- A C identifier is a name used to identify a variable, function, or any other user-defined item.
- An identifier starts with a letter
  - A to Z,
  - a to z,
  - or an underscore '_'
  - followed by
    - zero or more letters,
    - underscores,
    - and digits (0 to 9).
- C does not allow punctuation characters such as @, $, and % within identifiers.
- C is a **case-sensitive** programming language.

Basic Syntax: Keyword

- Keywords are set of reserved words in C
- Have special meaning and that meaning unchangeable.
- Keywords serve as basic building blocks for program statements.
- All keywords must be written in **lowercase**.
- Keywords cannot be used as constants or variables or any other identifier names.

Basic Syntax: Keyword Cont…

- The following list shows the reserved words in C.

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	double

Basic Syntax: Whitespaces

- A line containing only whitespace, possibly with a comment, is known as a blank line, and a C compiler totally ignores it.
- Whitespace is the term used in C to describe
  - blanks
  - tabs
  - newline characters
  - and comments
- Whitespace
  - separates one part of a statement from another
  - enables the compiler to identify where one element in a statement (such as int,)ends and the next element begins

Basic Syntax: Whitespaces Cont…

- Therefore, in the following statement –

```
int age;
```

- There must be at least one whitespace character (usually a space) between **int** and age for the compiler to be able to distinguish them.
- On the other hand, in the following statement –

```
fruit=apples+oranges;  // get the total fruit
```

- no whitespace characters are necessary
  - between fruit and =,
  - or between = and apples,
  - although you are free to include some if you wish to increase readability.

Variables

- A variable is a data name that may be used to store a data value.
- C is very fussy about how you create variables and what you store in them.
- It demands that you declare the name of each variable that you are going to use and its *type*, or *class*, before you actually try to do anything with it.
- A variable name can be chosen by the programmer in a meaningful way so as to reflect its function or nature in the program. Examples
  - Average
  - Height
  - Total
  - Counter

Variables Cont…

- Variables name may consists of letters(A-Z, a-z), digits(0-9) and the underscores (_).
- Variables follow following conventions:
  - They must begin with a letter or underscore
  - Max length 63 or 31 (depending on the machine)
  - Case sensitive
  - It should not be a keyword
  - Whitespaces are not allowed.
- Valid examples - John, Val__ue, minMax, number1, _int
- Invalid examples- 123, %, 25th, (area), #length, per cent

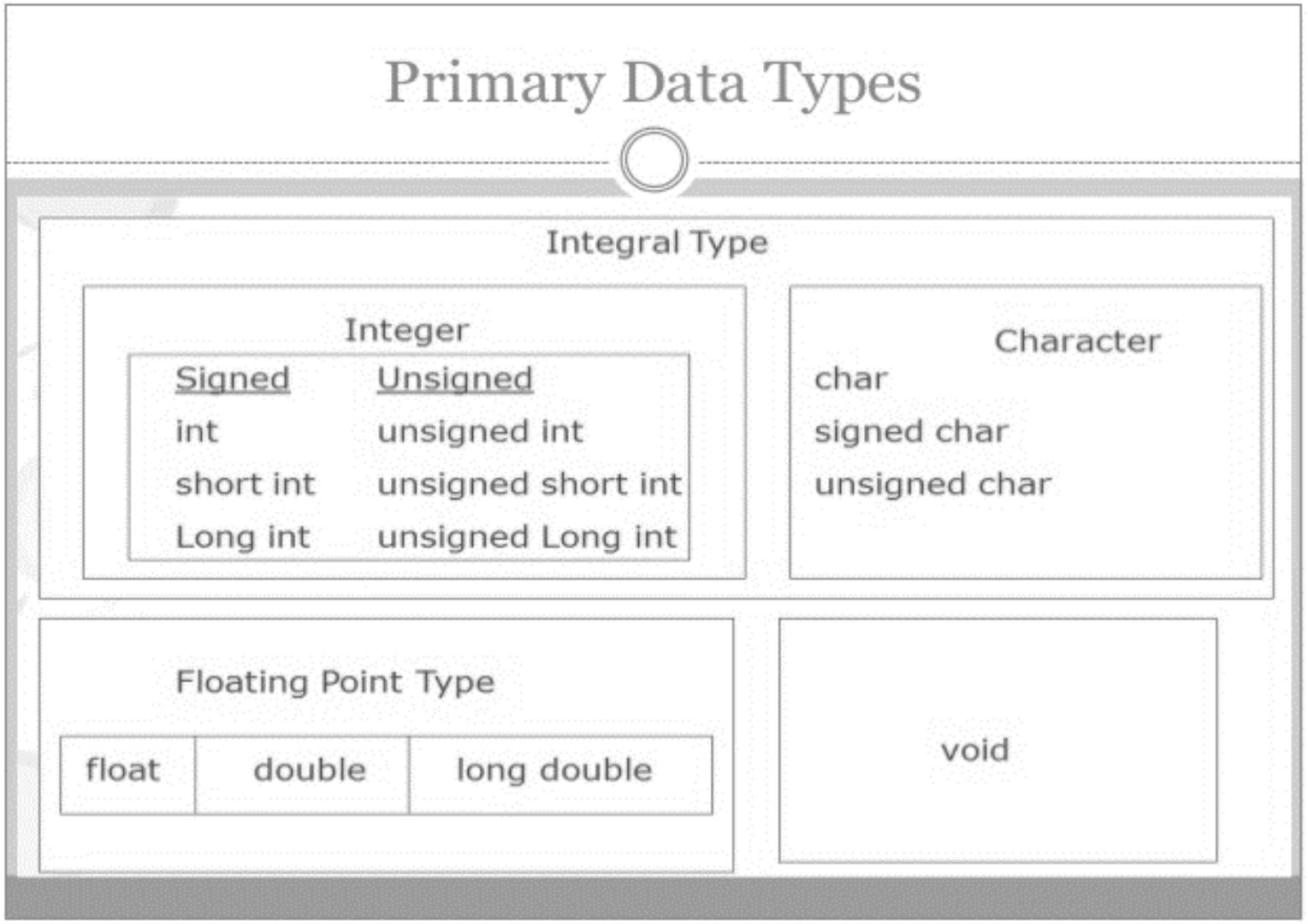
Data Types

- The type of a variable determines how much space it occupies in storage and how the bit pattern stored is interpreted.
- C supports three classes of data types:
  1. Primary data types
  2. Derived data types
  3. User-defined data types

Data Types Cont…

- All C compiler support following fundamental data types
  - Integer (int)
  - Character (char)
  - Floating point (float)
  - Double precision floating point (double)
  - And void
  - Long int (long long int)
  - Long double (long double)

Data Types: Primary



Data Types: Integers

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	4 bytes	-2,148,483,648 to 2,148,483,648
unsigned int	4 bytes	0 to 4,294,967,295
short	2 bytes	-32,768 to 32,767
unsigned short	2 bytes	0 to 65,535
long long	8 bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
unsigned long long	8 bytes	0 to 18,446,744,073,709,551,615

Data Types: Decimal

- The following table provide the details of standard floating-point types with storage sizes and value ranges and their precision −

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: Void

- The void type specifies that no value is available. It is used in three kinds of situations −

Sr.No.	Types & Description
1	**Function returns as void** There are various functions in C which do not return any value or you can say they return void. A function with no return value has the return type as void. For example, **void exit (int status);**
2	**Function arguments as void** There are various functions in C which do not accept any parameter. A function with no parameter can accept a void. For example, **int rand(void);**
3	**Pointers to void** A pointer of type void * represents the address of an object, but not its type. For example, a memory allocation function void *malloc(size_t size); returns a pointer to void which can be casted to any data type.

Variable Declaration

- A variable definition tells the compiler where and how much storage to create for the variable.
- A variable definition specifies a data type and contains a list of one or more variables of that type separated by commas −
  - type **variable_list**;
- type must be a valid C data type including char, int, float, double, bool, or any user-defined object; and
- **variable_list** may consist of one or more identifier names separated by commas.

Variable Declaration Cont…

- Some valid declarations are shown here −
  - int i, j, k;
  - char c, ch;
  - float f, salary, Float;
  - double d,_double, Double;
- The line **int i, j, k;** declares and defines the variables i, j, and k; which instruct the compiler to create variables named i, j and k of type int.
- Variables can be initialized (assigned an initial value) in their declaration. The initializer consists of an equal sign followed by a constant expression as follows −
  - type variable_name = value;

Variable Declaration Cont…

- Some valid declarations are shown here −

```
int d = 3, i = 5;        // definition and initializing i and f.
byte z = 22;             // definition and initializes z.
char x = 'x';            // the variable x has the value 'x'.
float f = 13.25;         // the variable f  has the value 13.25.
```

- For definition without an initializer:
  - variables with static storage duration are implicitly initialized with NULL (all bytes have the value 0);
  - the initial value of all other variables are undefined.

Simple C program using variables

```c
#include <stdio.h>// Variable declaration:
int main ()
{
   /* variable definition: */
   int number1, number2;
   int summation;
   float average;
   /* actual initialization */
   number1 = 10;
   number2 = 20;
   summation = number1 + number2;
   printf("value of summation : %d \n", summation);
   average = summation/2.0;
   printf("value of average : %f \n", average);

   return 0;
}
```

Lvalues and Rvalues in C

- There are two kinds of expressions in C −
  - **lvalue** − Expressions that refer to a memory location are called "*lvalue*" expressions.
    - An **lvalue** may appear as either the left-hand or right-hand side of an assignment.
  - **rvalue** − The term "*rvalue*" refers to a data value that is stored at some address in memory.
    - An **rvalue** is an expression that cannot have a value assigned to it which means an **rvalue** may appear on the right-hand side but not on the left-hand side of an assignment.

Lvalues and Rvalues cont…

- Variables are **lvalues** and so they may appear on the left-hand side of an assignment.
- Numeric literals are **rvalues** and so they may not be assigned and cannot appear on the left-hand side.
- Take a look at the following valid and invalid statements −

```
int g = 20; // valid statement

10 = 20; // invalid statement; would generate compile-time error
```