C PROGRAMMING FOR ENGINEERS

Program Structure
Constants and Variables
Assignment Statements

Program Structure

```
/*******************************************************
 SAMPLE PROGRAM

 This program computes the distance between two points.
*******************************************************/

/* Preprocessor directives */
#include <stdio.h>
#include <math.h>

/* Main function */
int main(void)
{
    /* Declare variables */
    double x1=1, y1=5, x2=4, y2=7, side1, side2,
           distance;

    /* Compute sides of triangle and distance */
    side1 = x2-x1;
    side2 = y2-y1;
    distance = sqrt(side1*side1 +  side2*side2);

    /* Print distance */
    printf("Distance = %5.2f\n",distance);

    /* Exit program */
    return 0;
}
/*******************************************************/
```

Program Structure – General Form

```
preprocessor directives
int main(void)
{
        declarations;
        statements;
}
```

Comments

```
/**********************************************************
  SAMPLE PROGRAM

  This program computes the distance between two points.
**********************************************************/
```

- Comments begin with /* and end with */.
- Used for program documentation and readability.
- A comment can be one line or it can have multiple lines, as seen above.
- Comments are ignored by the C compiler.

Preprocessor Directives

```
/*********************************************************
  SAMPLE PROGRAM

  This program computes the distance between two points.
*********************************************************/

/* Preprocessor directives */
#include <stdio.h>
#include <math.h>
```

Provides instructions that are performed before the program is compiled.

Header files in Standard C Library

#include <stdio.h>

#include <math.h>

The .h indicates a header file.

The < > symbols around the file name indicate that the file is included in the Standard C library, which is contained in the ANSI C compiler.

Main function

```
int main(void)
{

        (body of function goes between braces)

}
```

- Every C program will have a main function.
- The body of the function is enclosed in braces { }.
- The keyword int indicates that the main function returns an integer value to the operating system when finished executing.
- The keyword void indicates that the function is not receiving any information from the operating system.

Declarations

```
/* Main function */
int main(void)
{
    /* Declare variables */
    double x1=1, y1=5, x2=4, y2=7, side1, side2, distance;

}
```

must end with a semicolon

data type

variable names

- Variables are declared by stating the **type** and **variable name**, which defines a memory location in RAM.
- Variables can be assigned values or left unassigned.
- Commas separate variable names.
- All declarations end with a semicolon.

Statements

```
/* Compute sides of triangle and distance */
side1 = x2-x1;
side2 = y2-y1;
distance = sqrt(side1*side1 +  side2*side2);

/* Print distance */
printf("Distance = %5.2f\n",distance);
```

must end with a semicolon

must end with a semicolon

- Statements are operations performed during execution, like calculations and printing as seen above.
- Statements must end with a semicolon.

Return statement

```
    /* Exit program */
    return 0;
}
```

- The return statement provides a means to stop program execution. The digit 0 (zero) is returned to the operating system and signals that the program has finished executing.
- Return statements can be placed anywhere in the body of the program as required for other purposes, but should always be placed at the end of the program.

Constants and Variables

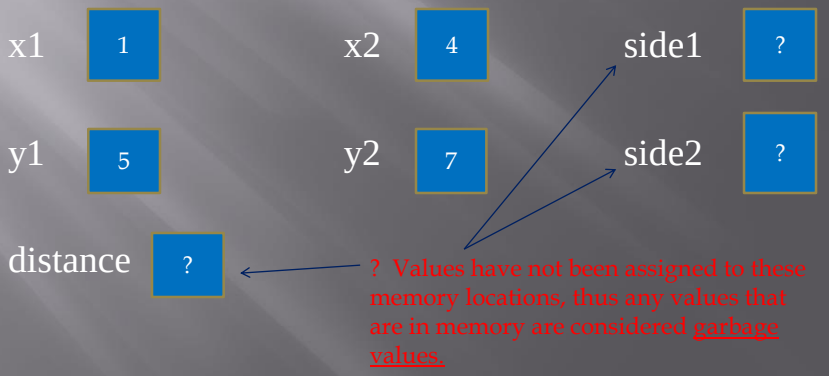
- Constants
  Specific values like:
  3.14
  5
  -2.56
  0
  'true'

- Variables
  Memory locations created in RAM that are assigned a name.

  The variable name is used to reference the value stored in the memory location.

Memory Locations

```
/* Declare variables */
double x1=1, y1=5, x2=4, y2=7, side1, side2,
       distance;
```



Rules for Selecting Valid Variables

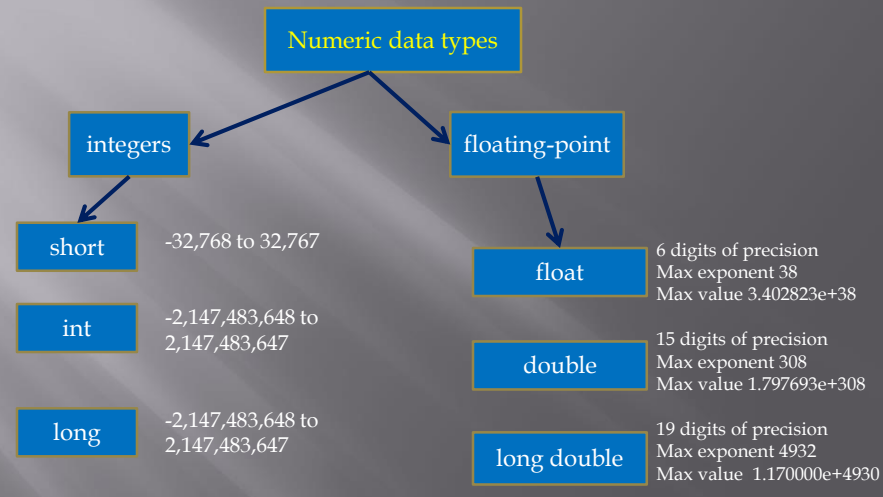
- Must begin with an alphabetic character or an underscore (%rate, $sum and #2014 are invalid variable examples).
- Case sensitive (e.g., UND, Und, und are completely different variables).
- Can contain digits, but not as the first character (e.g., 1x is not a valid variable).
- Can be of any length, but the first 31 characters must be unique.

Variable Rules Continued

- Cannot use the following keywords that are included 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

Numeric Data Types

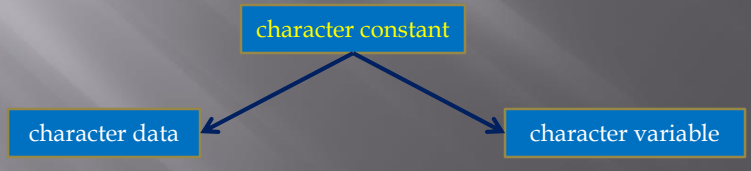


NOTE: The values listed above are system-dependent. The values may be different on different computers and using different C compilers.

Character Data

- Character data is stored in RAM as a binary coded value.
- The most common code format for PC's is ASCII:
  (American Standard Code of Information Interchange).

Examples:	Character	ASCII code	Integer equivalent
	a	1100001	97
	3	0110011	51



Examples: 'B', 'j', '6'
Note: must be enclosed in single quotations.

Example: char label1, label2;
Note: must be type declared.

Symbolic Constants

- Defined with a preprocessor directive that assigns a variable to the constant.
- Generally the directive is placed with other directives, but can appear anywhere in the program.
- Only one symbolic constant can be defined in a directive.

Examples: #define pi 3.14159

#define gravity 32.0

Some programmers will capitalize the symbolic name, but it is not required.

Notice that no semicolon is used to end the directive, and you do not place units either.