C INSTRUCTIONS

Lecture 3

Instructor: Yasher Ali

TYPES OF INSTRUCTIONS

- There are basically three types of instructions in C:
  a) Type Declaration Instruction – This instruction is used to declare the type of variables used in a C program.
  b) Arithmetic Instruction – This instruction is used to perform arithmetic operations on constants and variables.
  c) Control Instruction – This instruction is used to control the sequence of execution of various statements in a C program.
- Since, the elementary C programs would usually contain only the type declaration and the arithmetic instructions; we would discuss only these two instructions at this stage.
- The control instruction would be discussed in detail in the subsequent lectures.

TYPE DECLARATION INSTRUCTION 1

- This instruction is used to declare the type of variables being used in the program. Any variable used in the program must be declared before using it in any statement.
- The type declaration statement is written at the beginning of main( ) function.

```
Ex.:   int   bas ;
       float   rs, grosssal ;
       char   name, code ;
```

- There are several subtle variations of the type declaration instruction. These are discussed in next slides

TYPE DECLARATION INSTRUCTION 2

- While declaring the type of variable we can also initialize it as shown below

```
int  i = 10, j = 25 ;
float  a = 1.5, b = 1.99 + 2.4 * 1.44 ;
```

- The order in which we define the variables is sometimes important sometimes not. For example,

```
int  i = 10, j = 25 ;
```

is same as

```
int  j = 25, i = 10 ;
```

However,

```
float  a = 1.5, b = a + 3.1 ;
```

is alright, but

```
float  b = a + 3.1, a = 1.5 ;
```

TYPE DECLARATION INSTRUCTION 3

- is not. This is because here we are trying to use a before defining it.
- The following statements would work

```
int  a, b, c, d ;
a = b = c = 10 ;
```

- However, the following statement would not work

```
int  a = b = c = d = 10 ;
```

- Once again we are trying to use b (to assign to a) before defining it.

ARITHMETIC INSTRUCTION 1

- A C arithmetic instruction consists of a variable name on the left hand side of = and variable names and constants on the right hand side of =.
- The variables and constants appearing on the right hand side of = are connected by arithmetic operators like +, -, *, and /.

```
Ex.:   int   ad ;
       float   kot, deta, alpha, beta, gamma ;
       ad = 3200 ;
       kot = 0.0056 ;
       deta = alpha * beta / gamma + 3.2 * 2 / 5 ;
```

ARITHMETIC INSTRUCTION 2

- Here,
  - *, /, -, + are the arithmetic operators.
  - = is the assignment operator.
  - 2, 5 and 3200 are integer constants.
  - 3.2 and 0.0056 are real constants.
  - ad is an integer variable.
  - kot, deta, alpha, beta, gamma are real variables.

ARITHMETIC INSTRUCTION 3

- Though Arithmetic instructions look simple to use, one often commits mistakes in writing them. Let us take a closer look at these statements. Note the following points carefully:
- C allows only one variable on left-hand side of =. That is, z = k * l is legal, whereas k * l = z is illegal.
- In addition to the division operator C also provides a modular division operator. This operator returns the remainder on dividing one integer with another. Thus the expression 10 / 2 yields 5, whereas, 10 % 2 yields 0.
  - Note that the modulus operator (%) cannot be applied on a float.
  - Also note that on using % the sign of the remainder is always same as the sign of the numerator. Thus -5 % 2 yields –1, whereas, 5 % -2 yields 1.

ARITHMETIC INSTRUCTION 4

- Arithmetic operations can be performed on ints, floats and chars.

```
char   x, y ;
int    z ;
x = 'a' ;
y = 'b' ;
z = x + y ;
```

- Thus the statements, are perfectly valid, since the addition is performed on the ASCII values of the characters and not on characters themselves. The ASCII values of ‘a’ and ‘b’ are 97 and 98, and hence can definitely be added.
- No operator is assumed to be present. It must be written explicitly.

```
a = c.d.b(xy)          usual arithmetic statement
a = c * d * b * ( x * y )    C statement
```

ARITHMETIC INSTRUCTION 5

- There is no operator in C to perform exponentiation operation. Exponentiation has to be carried out as shown below:

```
# include <math.h>
# include <stdio.h>
int main( )
{
    float  a ;
    a = pow ( 3.0, 2.0 ) ;
    printf ( "%f", a ) ;
}
```

- Here pow( ) function is a standard library function. It is being used to raise 3.0 to the power of 2.0. The pow( ) function works only with real numbers, hence we have used 3.0 and 2.0 instead of 3 and 2.

INTEGER AND FLOAT CONVERSIONS

- An arithmetic operation between an integer and integer always yields an integer result.
- An operation between a real and real always yields a real result.
- An operation between an integer and real always yields a real result. In this operation the integer is first promoted to a real and then the operation is performed. Hence the result is real.

Operation	Result	Operation	Result
5 / 2	2	2 / 5	0
5.0 / 2	2.5	2.0 / 5	0.4
5 / 2.0	2.5	2 / 5.0	0.4
5.0 / 2.0	2.5	2.0 / 5.0	0.4

TYPE CONVERSION IN ASSIGNMENTS 1

- It may so happen that the type of the expression on right hand side and the type of the variable on the left-hand side of an assignment operator may not be same.
- In such a case, the value of the expression is promoted or demoted depending on the type of the variable on lefthand side of =.
- For example, consider the following assignment statements.

```
int   i ;
float   b ;
i = 3.5 ;
b = 30 ;
```

- Float 3.5 is demoted to an int
- 30 is promoted to 30.0 and then stored in b

TYPE CONVERSION IN ASSIGNMENTS 2

- Observe the results of the arithmetic statements shown in previous Figure.
- It has been assumed that k is an integer variable and a is a real variable.

Arithmetic Instruction	Result	Arithmetic Instruction	Result
k = 2 / 9	0	a = 2 / 9	0.0
k = 2.0 / 9	0	a = 2.0 / 9	0.222222
k = 2 / 9.0	0	a = 2 / 9.0	0.222222
k = 2.0 / 9.0	0	a = 2.0 / 9.0	0.222222
k = 9 / 2	4	a = 9 / 2	4.0
k = 9.0 / 2	4	a = 9.0 / 2	4.5
k = 9 / 2.0	4	a = 9 / 2.0	4.5
k = 9.0 / 2.0	4	a = 9.0 / 2.0	4.5

HIERARCHY OF OPERATIONS 1

- The priority or precedence in which the operations in an arithmetic statement are performed is called the hierarchy of operations.

Priority	Operators	Description
1st	* / %	Multiplication, Division, Modular division
2nd	+ -	Addition, Subtraction
3rd	=	Assignment

HIERARCHY OF OPERATIONS 2

- Example: Determine the hierarchy of operations and evaluate the following expression, assuming that i is an integer variable.

```
i = 2 * 3 / 4 + 4 / 4 + 8 - 2 + 5 / 8
```

- Stepwise evaluation of this expression is shown below:

```
i = 2 * 3 / 4 + 4 / 4 + 8 - 2 + 5 / 8
i = 6 / 4 + 4 / 4 + 8 - 2 + 5 / 8          operation: *
i = 1 + 4 / 4 + 8 - 2 + 5 / 8              operation: /
i = 1 + 1+ 8 - 2 + 5 / 8                   operation: /
i = 1 + 1 + 8 - 2 + 0                      operation: /
i = 2 + 8 - 2 + 0                          operation: +
i = 10 - 2 + 0                             operation: +
i = 8 + 0                                  operation : -
i = 8                                      operation: +
```

HIERARCHY OF OPERATIONS 3

- Example: Determine the hierarchy of operations and evaluate the following expression, assuming that kk is a float variable.

```
kk = 3 / 2 * 4 + 3 / 8
```

- Stepwise evaluation of this expression is shown below:

```
kk = 3 / 2 * 4 + 3 / 8
kk = 1 * 4 + 3 / 8          operation: /
kk = 4 + 3 / 8              operation: *
kk = 4 + 0                  operation: /
kk = 4                      operation: +
```

ALGEBRAIC EXPRESSION CONVERSION

- Some examples of algebraic expressions and their equivalent C expressions

Algebraic Expression	C Expression
a x b - c x d	a * b - c * d
(m + n) (a + b)	(m + n) * (a + b)
$3x^2 + 2x + 5$	3 * x * x + 2 * x + 5
$\frac{a+b+c}{d+e}$	(a + b + c) / (d + e)
$\left[\frac{2BY}{d+1} - \frac{x}{3(z+y)}\right]$	2 * b * y / (d + 1) - x / 3 * (z + y)