THERE IS NO NEED HOWEVER TO USE SYMBOLIC NAMES FOR CONSTANTS (LIKE ZERO, TWO, FIFTY.) INSTEAD WE MAY USE LITERAL VALUES.

e.g. 0, 2, 50 (int)
0.0, 2.0, 50.0, 3.14 (double)
'a', 'b', 'c' (char)

NOTE char LITERALS MUST BE ENCLOSED IN SINGLE QUOTES.

C/C++ ALLOWS THE USE OF SYMBOLIC CONSTANTS THROUGH THE USE OF const.

e.g. const double Pi = 3.14;

SYMBOLIC CONSTANTS, LIKE CONSTANT MACROS, ADD TO PROGRAM READABILITY AND MAINTAINABILITY.

EXECUTABLE STATEMENTS ARE OF THREE TYPES: INPUT, OUTPUT, AND CALCULATION.

SIMPLE KEYBOARD INPUT AND SCREEN OUTPUT ARE ACCOMPLISHED IN C++ BY USE OF THE INPUT STREAM cin, AND THE OUTPUT STREAM cout, BOTH DEFINED IN iostream

EX. `Cout << variable;`
`Cout << expression;`
`Cout << literal value;`

EX. `Cin >> variable;`

IN THIS CONTEXT << AND >> ARE CALLED THE EXTRACTION AND INSERTION OPERATORS, RESPECTIVELY.

IN THE FIRST EXAMPLE ABOVE, THE VALUE OF VARIABLE IS PRINTED TO THE STANDARD OUTPUT DEVICE (SCREEN). VARIABLE CAN BE OF ANY BUILT IN C++ DATA TYPE.

EX.
```
int count; double sum; char letter;
  :
Cout << count << sum << letter;
```

WE CAN ALSO PRINT STRING LITERALS

EX. `Cout << " ENTER TEMPERATURE IN FAHRENHEIT: ";`

NOTE STRING LITERALS MUST BE ENCLOSED IN DOUBLE QUOTES. A STRING IS NOT A SEPARATE DATA TYPE IN C/C++. INSTEAD IT IS AN ARRAY OF CHARACTERS. MORE ON ARRAYS LATER.

TWO SPECIAL CHARACTERS ARE THE NEWLINE \n, AND THE TAB \t.

Ex.
```
int hours, minutes;
  :
cout << "THE TIME is NOW \n"
     << hours << ':' << minutes
     << endl ;
```

WHEN PLACED IN THE OUTPUT STREAM, endl CAUSES A NEWLINE TO BE PRINTED.

Ex. `cin >> variable;`

THIS CAUSES EXECUTION TO PAUSE UNTIL A VALUE IS ENTERED AT THE STANDARD INPUT DEVICE (KEYBOARD) WHICH IS THEN ASSIGNED TO variable.

CALCULATIONS CAN BE PERFORMED BY USING THE BASIC ARITHMETIC OPERATORS +, -, *, / (AND % FOR int ONLY.)

A STATEMENT OF THE FORM

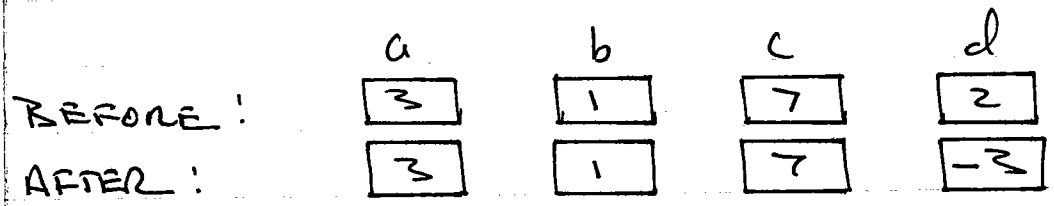
variable = expression;

DIRECTS THE COMPUTER TO EVALUATE expression,

THEN ASSIGN THE RESULTING VALUE TO variable. WE CALL = THE ASSIGNMENT OPERATOR.

Ex
```
int a, b, c, d;
 :
d = a + b - c;
```

DATA AREAS:



NOTE ASSIGNMENT = MUST BE DISTINGUISHED FROM COMPARE FOR EQUALITY ==.

EX. WRITE A C++ PROGRAM TO CALCULATE THE CIRCUMFERENCE AND AREA OF A CIRCLE, GIVEN ITS RADIUS.

ALGORITHM

1.) GET VALUE FOR radius FROM USER.
2.) SET area TO π * radius * radius
3.) SET circumference TO 2*π*radius.
4.) PRINT VALUES OF area AND circumference.

```
////////////////////////////////////////////////////////////////
//   File: circle.cpp
//   Description: Get input value for the radius of a circle,
//   print out the circumference and area.
//   Compile:  g++  -o circle  circle.cpp
////////////////////////////////////////////////////////////////

#include<iostream>
using namespace std;

#define  PI  (3.14159)

int main(void)
{
     double radius,          // Input: radius of circle.
            area,            // Output: area of circle
            circumference;   // Output: circumference of circle

     // Get radius from user.
     cout << "Enter the circle radius: ";
     cin >> radius;

     // Compute area.
     area = PI*radius*radius;

     // Compute circumference.
     circumference = 2*PI*radius;

     // Print out area and circumference.
     cout << "The area is " << area << ", and the circumference"
          << " is " << circumference << "." << endl;

     return (0);
}
```

Compile:

```
% g++ -o circle circle.cpp
```

Output:

```
% a.out
Enter the circle radius: 37
The area is 4300.84, and the circumference is 232.478.
%
```

WE CAN USE A CONSTANT MACRO FOR THE PROGRAM CONSTANT π :

```
#define PI 3.14
```

OR WE CAN DECLARE A SYMBOLIC CONSTANT WITHIN FUNCTION main :

```
const double Pi = 3.14 ;
```

— SEE EXAMPLE circle.cpp —

ARRAYS

AN ARRAY IS A CONTIGUOUS SET OF MEMORY LOCATIONS EACH OF WHICH STORE DATA OF THE SAME TYPE.

ARRAYS ARE DECLARED AS FOLLOWS :

EX
```
int    list [10]
double array[20];
char   word[30];
```

THESE DECLARATIONS SET ASIDE SPACE IN MEMORY FOR 10 ints, 20 doubles, AND 30 chars RESPECTIVELY.

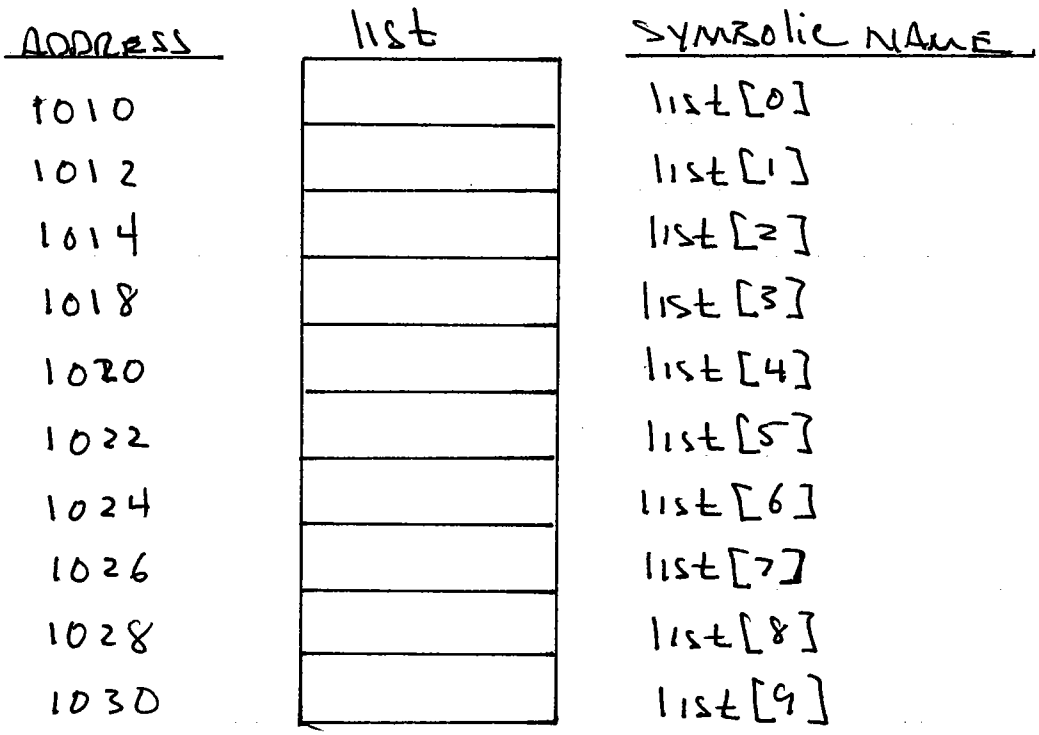
NOTE ARRAY NAMES MUST FOLLOW THE RULES FOR C/C++ IDENTIFIERS. IN PARTICULAR THE WORD "array" IS NOT A RESERVED WORD IN C/C++, THOUGH IT MAY BE A POOR CHOICE FOR AN ARRAY NAME.

THE ELEMENTS OF ARRAY list ARE REFERED TO AS

list[0], list[1], ... , list[9].

THE VALID ARRAY INDICIES START AT 0 AND END AT ONE LESS THAN THE NUMBER OF ELEMENTS IN THE ARRAY.

THE DATA AREA FOR ARRAY list MAY BE PICTURED AS:



OBSERVE THAT list[10] DOES NOT REFER TO A MEMORY LOCATION IN ARRAY list.

WARNING: IT IS POSSIBLE HOWEVER TO READ OR WRITE PAST THE END OF AN ARRAY BY REFERENCING SAY list[10], list[11], ETC. THIS IS NOT A <u>SYNTAX ERROR</u> IN C/C++ BUT CAN CAUSE <u>RUN TIME ERRORS</u> WHICH ARE VERY DIFFICULT TO TRACK DOWN.

AN ARRAY OF chars (LIKE word ABOVE) IS CALLED A <u>STRING</u> IN C/C++. THERE ARE CERTAIN SPECIAL OPERATIONS FOR PROCESSING STRINGS IN THE LIBRARY string.h, WHICH WE WILL NOT COVER.

C/C++ Allows MULTIDIMENSIONAL ARRAYS

<u>EX.</u> double table[5][6]

DECLARES AN ARRAY OF 30 doubles WHICH WE MAY THINK OF AS BEING ARRANGED IN 5 ROWS AND 6 COLUMNS:

table[0][0], ... , table[0][5]
:
:
table[4][0], ... , table[4][5]