MODULE 3: Data Types, Constants, Identifiers, Basic Input and Output, and Arithmetic Operators

DATA TYPES

In C, there are A LOT of Basic Data types (see link below).

<http://en.wikipedia.org/wiki/C_data_types>

What will be used here are the standard ones, VOID, INT , CHAR and FLOAT.

C has no logical data type.

In C, any nonzero number is considered TRUE and zero as FALSE.

CONSTANTS:

Literal Constant: Unnamed Constant

Example: 85 -> this is in decimal

Defined Constant: no semicolon after definition using the format #define <directive>

Example: #define TAX\_RATE 0.67

Memory Constant: declared same as the local variables

Example: const float TAX\_RATE = 0.67;

(*There is a semicolon*)

INTEGER

Literal Value Data Type

+123 123 int

-378 -378 int

-32271L -32,271 long int

76542LU 76,542 unsigned long int

FLOAT

Literal Value Data Type

0.0 0.0 double

2.0 2.0 double

-2.0f -2.0 float

3.1416L 3.1416 long double

CHARACTER

ESCAPE SEQUENCES

Null character ‘\0’ Vertical tab ‘\v’

Alert (bell) ‘\a’ Form feed ‘\f’

Backspace ‘\b’ Carriage return ‘\r’

Horizontal tab ‘\t’ Single quote ‘\’ ’

Newline ‘\n’ backslash ‘\\’ To indicate e*scape sequences, they start with a backslash(‘\’).*

IDENTIFIERS:

• way of naming data and objects

• names of functions, variables, derived data types, and more

Valid identifiers

All letters from A to Z and a to z.

All numbers from 0 to 9.

Underscore

Examples:

Var1

Var\_1

Var

var

gwapo\_ko\_12

YOU CANNOT USE:

RESERVED WORDS

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

Variables

Syntax:

<data type> variable\_name

Examples:

int number;

int num = 7;

A variable maybe local, global or a static variable.

Local variables are inside functions.

Example:

int main()

{

char c;

}

Global variables are declared outside the functions right after the preprocessor directives.

#include <stdio.h>

char c;

const char key = ‘K’;

void main()

{

/\* insert code here \*/

}

BASIC INPUT AND OUTPUT

INPUT:

scanf(“format string”, data\_address\_list);

Example:

#include <stdio.h>

void main()

{

int num;

scanf(“%d”, &num);

}

*TAKE NOTE:*

* int num is a variable declaration without initializing its value. The variable num contains *garbage* inside.
* The ampersand (&) indicates that you want the ADDRESS of the variable num to be used.

OUTPUT:

printf(“format string”, data\_list);

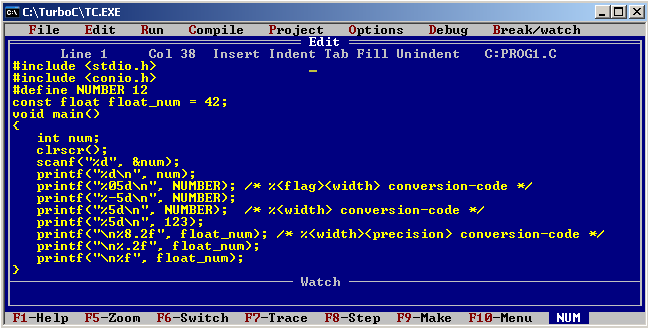
*Here, it is the data\_list and, not the data\_address\_list unlike scanf()*

Format Specification Syntax

%< flag > <width> <precision> <size> conversion-code

Conversion code is the letter/s that comes after the % in %d, %lf, etc.

Consider the following code.



Observe the output carefully.

ARITHMETIC OPERATORS

Addition + Result is the sum of two operands

Subtraction - Result is the difference between 2 operands

Multiplication \* Result is the product of 2 operands

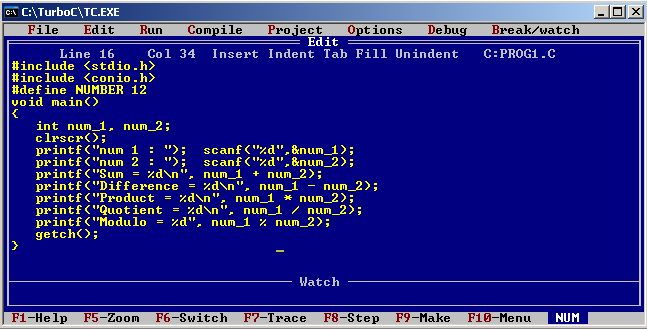
Division / Result is the quotient of first operand by second operand.

•Integer quotient if both operands are integers.

•Floating point quotient if either operand is of float data type.

Modulo % Result is remainder after first operand is divided by the second operand. Both operands are integers.

Consider this program.



Next, do this.

Change num\_2’s data type into float.

Change to scanf(“%f”,&num\_2);

Change all %d of Sum to Quotient into %f

Ommit or comment the printf for modulo because it only works on int.

num = num + 4;

* this is the same as

num += 4;

This is true for other arithmetic operands. Try -=, \*=, etc.

COMMENTS:

To make a comment in C, simply add /\* before the area to comment and then \*/ after.

Examples:

/\* Insert code or comment here \*/

Or

/\*

Insert code or comment here

\*/

Or

/\* Insert code

Or comment here \*/