Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF AIML

OBLEM SOLVING AND C PROGRAMMING

I YEAR - I SEM

UNIT 2 – C Programming Basics

TOPIC 5 – Variables

s a data name that may be used to store a data value.

nay take different values at different times during execution.

bles of variables' names are:
e

_1
rength

nes may consist of letters, digits, and the underscore(_) character, a
ving conditions:

egin with a letter.
ystems permit underscore as the first character.
rd recognizes a length of 31 characters.
er, length should not be normally more than eight characters, since
aracters are treated as significant by many compilers.
nd lowercase are significant.
the variable 'Total' is not the same as 'total' or 'TOTAL'.
be a keyword.
is not allowed.

ples of valid variable names are:

ue T_raise Delhi x1 ph_value mark

nples include: 123 (area) % 25th

ing suitable variable names, we must declare them to the compile
does two things:
tells the compiler what the variable name is.
specifies what type of data the variable will hold.

NT NOTE:
ion of variables must be done before they are used in the prog

can be used to store a value of any data type.
is, the name has nothing to do with its type.
for declaring a variable is as follows:

data-type v1,v2,....vn ;

vn are the names of variables.
are **separated by commas.**
ion statement must end with a semicolon.

le, valid declarations are:
ount;
umber, total;
le ratio;

uble are the keywords to represent integer type and real type data v
y

Program to Add Two In

```
#include <stdio.h>
int main() {

    int number1, number2, sum

    printf("Enter two integer
    scanf("%d %d", &number1,

    // calculating sum
    sum = number1 + number2;

    printf("%d + %d = %d", nu
    return 0;
}
```

Output

```
Enter two integers: 12
11
12 + 11 = 23
```

ntifier:

feature known as “type definition” that allows **users to ‘define’ a**
epresent an existing data type.

fined data type identifier can later be used to declare variables.

general form:

» typedef type identifier;

’ refers to an existing data type and “identifier” refers to the “new
e.

hat the new type is ‘new’ only in name, but not the data type.

edef type identifier;

bles of type definition are:
edef int units;
edef float marks;
nits symbolizes **int** and **marks** symbolizes **float**.
later used to declare variables as follows:
its batch1, batch2;
arks name1[50], name2[50];
atch1 and batch2 are declared as **int** variable and name1[50] and n
l as **floating point** array variables.

vantage of typedef is that we can create meaningful data type nam
ty of the program.

ifier:

r-defined data type is enumerated data type provided by ANSI sta

as follows:

enum identifi er {value1, value2, ... valuen};

ier" is a user-defined enumerated data type which can be used to d

e one of the values enclosed within the braces (known as enumera

finition, we can declare variables to be of this 'new' type as belo

enum identifier v1, v2, ... vn;

ated variables v1, v2, ... vn can only have one of the values value1

m identifier {value1, value2, ... valuen};
:
enum day {Monday,Tuesday, ... Sunday};
enum day week_start, week_end;

week_start = Monday;
week_end = Sunday;

if(week_st = = Tuesday)
week_end = = Monday;

er automatically assigns integer digits beginning with "**0**" to all the

enumeration constant value1 is assigned 0, value2 is assigned 1, a
e automatic assignments can be overridden by assigning values ex
constants.

in C can have not only data type but also **storage class** that provides inform
r location and visibility.
ge class decides **the portion of the program** within which the variables are
d.
the following example:

```
mple of storage classes */
int m;
main()
{
        int i;
        float balance;
        ....
        function1();
}
function1()
{
        int i;
```

n which has been declared before the **main** is called
le".
in all the functions in the program.
declared in other functions.
able is also known as an external variable.

i, balance and sum are called "local variables".
are declared inside a function.
es are **visible and meaningful only inside** the functions in
e declared.
known to other functions.

SCOPE OF A VARIABLE

variable **i** has been declared in both the functions.
n the value of i in one function does not affect its value in

```
/* Example of s
        int m;
        main()
        {
                int
                flo
                ....
                fu
        }
        function
        {
                int
                flo
                ....
        }
```

storage class specifiers:

ge class	Meaning
	Local variable known only to the function in which it is declared. *Default is auto*.
	Local variable which exists and retains its value even after the control is transferred to the calling function.
	Global variable known to all functions in the file.
	Local variable which is stored in the register.

ass is another qualifier (like long or unsigned) that can be added to a variabl

unt;
ar ch;

total;

ernal (extern) variables are automatically initialized to zero.
to) variables contain undefined values (known as ‘garbage’) unless they are

```
<stdio.h>
<conio.h>
in()

 340;
(“C = %d”, c);

t c = 450;
intf(“C = %d”, c);

(“C = %d”, c);
);
```

```
#include<stdio.h>
#include<conio.h>
Void main()
{
   int static c= 340;
   Printf(“C = %d”, c);
    {
        int c = 450;
        Printf(“C = %d”, c);
    }
   Printf(“C = %d”, c);
   getch();
}
```

Output:

C = 340

C = 340

created for use in program statements such as:
ount + inrate * amount;
<= PERIOD)

= year + 1;

tement, the **numeric value** stored in the variable **inrate** is multiplied by the
he product is added to **amount**.
tored in the 'variable' value.
s possible only if the variables amount and inrate have already been given v
alue is called the **target variable**.
variables are declared for their type, the variables that are used in expressio
=) sign of a computational statement) must be assigned values before they a
n.
variable **year** and the symbolic constant **PERIOD** in the while statement m

assigned to variables using the assignment operator "= " as follows:

variable_name = constant;

e = 0;
e = 100;
75.84;

tiple assignments in one line.

e = 0; fi nal_value = 100; are valid statements.

t statement implies that the value of the variable on the **left** of the 'equal sig
e quantity (or the expression) on the **right**.
:
+ 1;
the 'new value' of year is equal to the 'old value' of year plus 1.

ment operation, C converts the type of value on the right-hand side to the ty
lve **truncation** when real value is converted to an integer.
ble to assign a value to a variable at the time the variable is declared.
following form:
type variable_name = constant;
es are:
nal_value = 100;
yes = ‘x’;
le balance = 75.84;
f giving initial values to variables is called **initialization**.
initialization of more than one variables in one statement using multiple ass

= s = 0;
= z = 10;
first statement initializes the variables p, q, and s to zero while the second i

of giving values to variables is to input data through keyboard using the **scan**
input function available in C and is very similar in concept to the **printf** fun
like an INPUT statement.
rmat of **scanf** is as follows:

scanf("control string", &variable1,&variable2,....);

ring contains the format of data being received.
d symbol **&** before each variable name is an operator that specifies the varia

>

ber2, sum;
integers: ");
&number1, &number2);
number2;
= %d", number1, number2, sum);

OUTPUT:

Enter two integers: 12 11
12+11 = 23

>

OUTPUT:

Enter two integers: 12 11

12+11 = 23

ber2, sum;

integers: ");

&number1, &number2);

number2;

= %d", number1, number2, sum);

%d", &number1, &number2);

ement is encountered by the computer, the execution stops and waits for the

er to be typed in.

rol string “%d” specifies that an integer value is to be read from the termina

ue in integer form.

ber is typed in and the ‘Return’ Key is pressed, the computer then proceeds

Entire Data types in c:

Data type	Size(bytes)	Range	Format string
Char	1	128 to 127	%c
Unsigned char	1	0 to 255	%c
Short or int	2	-32,768 to 32,767	%i or %d
Unsigned int	2	0 to 65535	%u
Long	4	-2147483648 to 2147483647	%ld
Unsigned long	4	0 to 4294967295	%lu
Float	4	3.4 e-38 to 3.4 e+38	%f or %g
Double	8	1.7 e-308 to 1.7 e+308	%lf
Long Double	10	3.4 e-4932 to 1.1 e+4932	%lf

ine symbolic-name value of constant

s of constant definitions are:
RENGTH 100
SS_MARK 50
AX 200
3.14159
es are sometimes called constant identifiers.
bolic names are constants (not variables), they do not appear in declarations

Statement	Validity	Remark
2.5	Invalid	'=' sign is not allowed
X 10	Invalid	No white space between # and define
5;	Invalid	No semicolon at the end
M 10	Invalid	A statement can define only one name.
RAY 11	Invalid	define should be in lowercase letters
CE$ 100	Invalid	$ symbol is not permitted in name

g rules apply to a #define statement which define a symbolic constant:

mes have the same form as variable names. (Symbolic names are written in **CAPITALS**
hem from the normal variable names, which are written in lowercase letters.

ace between the pound sign '#' and the word define is permitted.

the first character in the line.

ce is required between #define and symbolic name and between the symbolic name and
.

ements must not end with a semicolon.

tion, the symbolic name should not be assigned any other value within the program by
signment statement. For example, STRENGTH = 200; is illegal.

ames are NOT declared for data types. Its data type depends on the type of constant.

ements may appear anywhere in the program but before it is referenced in the program
ractice is to place them in the beginning of the program).