Chapter-3

Variables and Data Types

Date
Page

Structure of ~~Variables~~ C-programs.

```
/* Name:.....
   Roll no:.....
   faculty:.....*/
```
} Documentation / Comment Section

```
#include <stdio.h>
#includ <conio.h>
```
} ~~Preprocessed~~ Preprocessor Statement.

```
// Global Declaration Section.
int main()
{
    Printf ("Hello World");
    getch();
    return 0;
}
// Sub-Program Section
```

Date
Page

1. * Documentation section: It is the part of the program where the programmer gives the details associated with the program.

2. * Pre-processor Directives: This statement begins with '#' symbol. It directs the C ~~pre-processor~~ 'C' compiler to include header files. and symbolic constant into the C program.

~~3. * Main~~

3. Global Declaration Section().
This part of the code includes global variables.

4. Main().
Execution of 'C' program starts from main function. There must be exactly one main function in every 'C' program.

5. Sub-program Section.
All user-defined functions are defined in this section of the program.

~~Compo~~ p & &

Date
Page

Tokens in C: These are also known as building blocks of C programming. They are small units that helps in developing a fully executable program.

a. Keywords: keywords are the predefined words that have special meaning to the compiler. There are 32 keywords defined by C compiler. For example:

- main
- for
- if
- switch
- while
- int.

b. Special symbols: All the symbols except alphabets, digits and wide space are considered as special symbols
For examples: @ , ! , # , * , ? & .

c. Operators: These are the mathematical symbols that are used to perform calculation and comparisons.
For examples: ++ , --

V.V.I

Date
Page

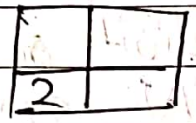
(d) Variables: Variable is a symbol that represents a storage location in the computes memory.

Syntax:

data-type variable name;

For example:

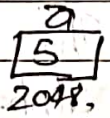
int num;



Main Memory.

- Static Initialization of variable ⇒

If the initilization of the variable is done at compiled time, it is called Static Initialization of variable. for example: int a = 5;



- Run time Initilization of variable.

If the Initilization of variable is done during the execution of program, it is called Run time Initilization of Variable

For example; int n;

int i=1,

Date
Page

Rules of Variable

Rules for Naming a variable:

i. A variable name is any combination alphabets, digits or underscore (_).

ii. No ~~sym~~ special symbol other than underscore (_) can be used in variable name.
for example:
num@ 2 (x) : Invalid
num_2 (✓) : valid

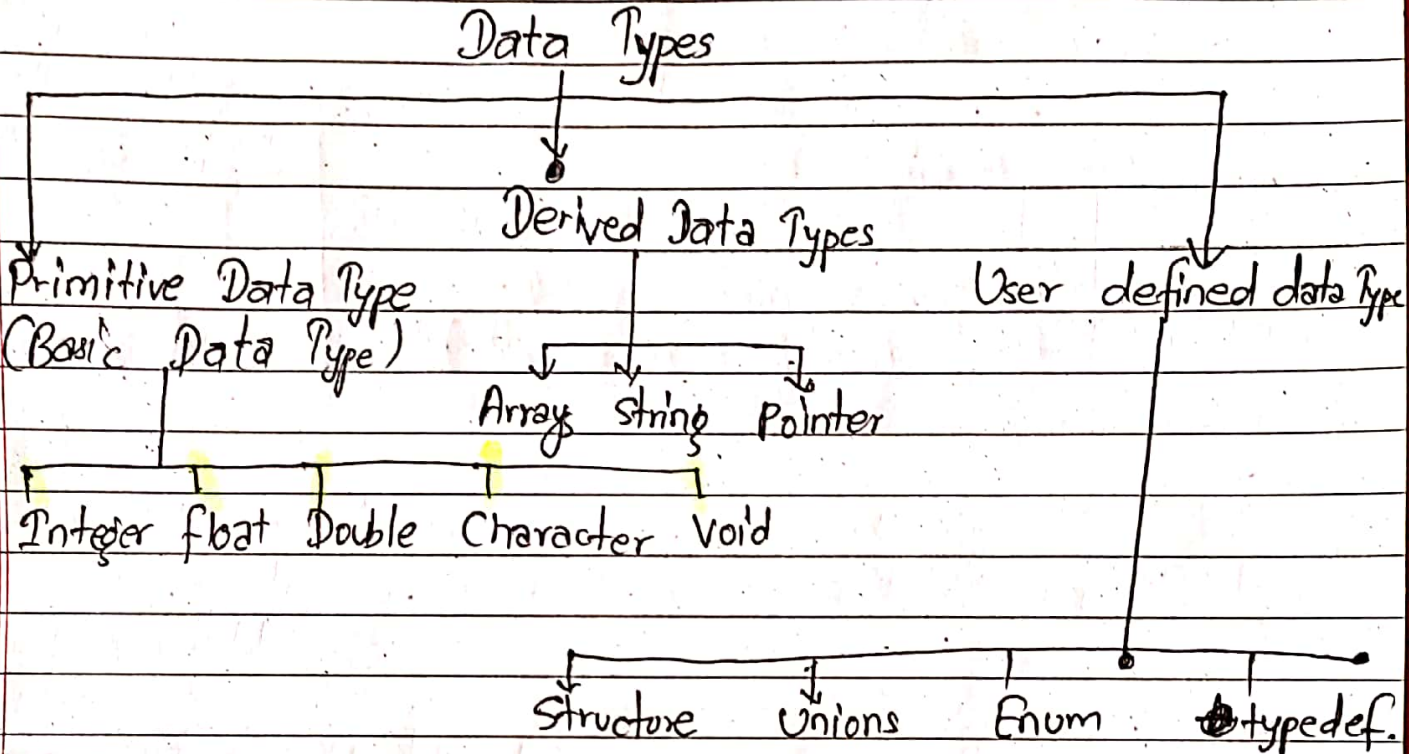
iii. The first letter of a variable name must be a alphabet or underscore (_).
For example: 2_num (x) : Invalid.
_2num (✓) : valid
2num (x) : Invalid.

iv. No commas or blanks are allowed within a variable name.
for example:
Oribinal number (x) Invalid.
Original_number (✓) valid.
nuum] (x) Invalid.

v. A variable name cannot be a keyword.
for example: int for, (x)
int , (x)
Int (✓)

Date
Page

※ Data Types in C



Primitive Data Type: The Primitive Data Type in C language are the ~~inbuild~~ inbuilt data types provided by 'C' language itself.

a. Integer:

- Used to store whole Number.
- Size: 2 to 4 bytes (depending on compiler).
- keyword Int.
- format specifier, %d.

Date
Page

Q.N.1 WAP to add two Integers Number.

Ans

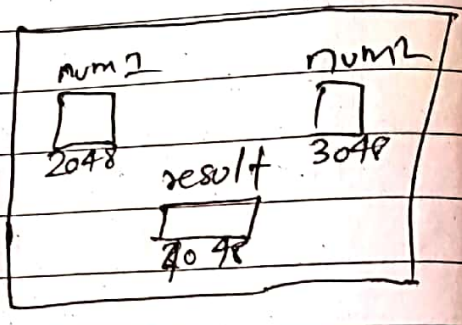
```
#includes <stdio.h>
# include <conio.h>

int main ()
{
    int num1, num2, result;
Printf("Enter first number");
Scanf ("%d",&num1);

Printf ("Enter second numbr");
Scanf ("%d",& num2);

    result = num1+num2;
Printf(" The sum is %d", result);

    getch ();
    return 0;
}
```



Output

```
Enter first number; 2
Enter second number; 3
The sum is 5
```

2. Float Data Type.

They are used for storing decimal number

- keywords float.
- format specifier = %f
- Size 4 bytes.
- Precision 7 digits.
- Range -3.4×10^{38} to 3.4×10^{38}

* WAP for devision of two number.

```
# include <stdio.h>
# include <conio.h>

int main(){
    float num1, num2, result;
printf ("Enter First Number:");
scanf ("%f", &num1);

printf ("Enter Second Number");
Scanf ("%f", &num2);

result = num1/num2;

printf ("The result is %f", result);
getch();
return 0;
}
```

Note
for 0.333
.3%f

Output:

Enter first Number 2
Enter Second Number 2
result is 0.800.

3. Double data Type -
- keyword double
- format specifier %lf
- precision 14 digits
- size: 8 bytes.
- range -1.7×10^{308} to 1.7×10^{308}

4. Character data Type
- It used to store single character.
- keyword char
- Size : 1 byte
- format specifies : %c
- range : 0 to 255

Date
Page

* WAP to input and display a single character.

```
#include <stdio.h>
#include <conio.h>

int main(){
  char ch;

  printf("Enter any one character");
  scanf("%c",&ch);

  Print f("The character is %c",ch);
  getch(.);
  return 0;
```

Enter any one character a
The character is a

* WAP to find ASCII code of a given character?

```
# include<stdio.h>
# include<conio.h>
    int main()
    {
      char ch;
    printf("Enter any one character");
     Scanf("%c", &ch);

Printf("The ASCII code of %c is %d", ch, ch);
      getch();
       return 0;
       }
```

* WAP to Input ASCII code and Print Corresponding character.

```
# include<stdio.h>
# include<conio.h>

int main(){
    int a;

printf("Enter ASCII code:");
Scanf("%d", &ch);

print f("The character of %d is %c", a, a);
      getch();
      return 0;
    }
```

Date
Page

(7) Enter ASCII code 97
The character is 97 is a

Date
Page

(5) Void Data Type

It is generally used to denote the type of a function that doesn't have a return type.

- Size : 1 byte

* WAP to calculate Simple Interest

```
#include<stdio.h>
#include<conio.h>

int main()
{
    float p,t,r,I;
printf("Enter Principle amount:");
 Scanf("%f",&p);

printf("Enter Time in years:");
 Scanf("%f", &t);

printf("Enter Rate:");
 Scanf("%f",&r);

 I = (P*T*r)/100;

printf("The interest is %f",I);
 getch();
 return 0;
}
```

output:

```
Enter Principle amount 2000
Enter Time in years: 2
Enter Rate : 0.2
The interest is: 8
```

ax^2+bx+c

Date
Page

Q. WAP to find roots of quadratic Equation:

```
#include <stdio.h>
#include <conio.h>
#include<math.h>
    int main(){
        int a,b,c;
        float r1, r2;
printf("Enter value of a,b and c:");
Scanf("%d%d%d", &a, &b, &c);

    r1 = (-b + sqrt(b*b-4*a*c))/2*a;
    r2 = (-b - sqrt(b*b-4*a*c))/2*a;

printf("r1 = %f. and r2 = %f", r1, r2);
        getch();
        return 0;
    }
```

return 0;

Date
Page

* Defining Constants:
These are two ways to define constant in C programming.

a. Using # define preprocessor [Symbolic constant]
b. Using const keyword [Variable Constant]

** Using # define preprocessor [Symbolic constant]
Syntax
define SYMBOL_NAME value ← structure
Example:
define PIE 3.14

- Symbolic names: Symbolic name are usually written Upper case
- # define statement mustn't end with semi-colon.
- A space is required between # define and symbol name and between symbol name and value.

Date
Page

// Program to calculate circumference & area of circle using Symbolic constant.

```
# include <stdio.h>
# include <conio.h>
# define PIE 3.14
Int main(){
    float r, area, circumference;
printf("Enter radius of circle:");
Scanf("%of", &r);

area = PIE *r*r;

circumference = 2* P.IE *r

Printf("Area is: %of", area);
Printf("\n Circumference is: %of", circumference);
getch();
return 0;
}
```

```
Enter radius of circle : 7
Area is 3.14
Circumference is 6.28
```

OTP

// Program to calculate Volume & area of sphere using Symbolic constant.

```
#include<stdio.h>
#include<conio.h>
#define PIE 3.14

int main()
{
    float r, area, volume;
printf("Enter radius of sphere:");
 Scanf("%f", &r);
area = 4 * PIE * r * r;
Volume = 4/3 * PIE * r * r * r;
          0
Printf("Area is : %f", area);
Printf("\n Volume is: %f", Volume);
getch();
return 0;
}
```

```
Enter radius of sphere: 1
Volume is: 4.1866
area is: 12.56
```

O/P

b Using const keyword

Syntax : Const datatype variable_name = value;
Example : Const int full_marks = 100;

II Program to calculate percentage of a student.

```
# include <stdio.h>
# include <conio.h>
  int main() {
      int obt_marks;
      float percent;
   Const int full_marks = 100;

  Printf ("Enter obtained marks: ");
  Scanf ("%d" & obt-marks);
  percent = (obt_marks/full marks)*100;
  printf ("Percentage: %f", percent);
   getch();
  return 0;
   }
```

Enter obtained marks : 90
Percentage is : 90

o/p

Date
Page

* **Type Casting:** Type Casting is the process of coverting one data type into another data type. C language provides two ways for type casting.

a. Implicit Type casting.
b. Explicit Type casting.

Implicit Type casting

It is the process of converting a data value from one type to another type automatically by compiler.

For Example:

```
// Eample of Implicit type casting.
#include <stdio.h>
#include <conio.h>
int main(){
    int x = 10;
    char y = 'a';
    float z;

x = x+y ; //y is implicitly converted to integer.
Printf ("Value of x: %d", x);
z = x+1.0 ; //x is implicitly converted in float type.
printf ("Value of z: %f", z);
getch();
return 0;
```

```
value of x: 107
value of z: 108.000000
```
O/P

Date
Page

Explicit Type Casting.

It is the process of converting one data type into another data type by the user according to the requirement.

// Example of Explicit type casting.

```
# include <stdio.h>
# include <conio.h>
  int main ()
      int a=2, b=10
      float c;
   c= a/b;

Printf ("value before Typecasting: %f", c);
   c= (float) a/b;

printf ("\n value after type casting: %f", c);
   getch();
   return 0;
   }
```

Output:
value before type casting: 0.000000
value after type casting: 0.200000

Date
Page

* Escape Sequence :

In C, backslash (\) is known as escape character because it causes skip from a normal interpretation of a string. So that the character is recognized having a special meaning. A blackslash followed by one or more character is known as Escape Sequence.

Escape Sequence	Meaning
\n	new line
\t	tab (4 spaces)
\a	beep sound
\'	Single Quote
\"	double Quote
\\	Display Backslash.
\0	Null character

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

printf("You \n are \n learning \"C\" language");
print f("\n\" best of luck \"");
```

```
You
are
learning "C" language
"Best of luck"
```

Date
Page

* Formatted & Unformatted Input/Output :

C language provides us console input/output functions which is used to:
i. read input from keyboard by the user accessing the console.
ii. Display the output to the user at the console.
* There are two types of console input/output functions.

1. Formatted Input/Output:
These functions are used to take one or more input from the user. While calling formatted input/Output functions, We must use format specifier in them.

Scanf() → Used to read one or more input from user at console.
print f() → Used to display one or more values to user at console.

2. Unformatted Input/Output.
These function are used to read a single input/display single output. while calling any of the unformatted input/output functions, we don't use format specifier.

Date
Page

getch() Syntax: variable_name = getch();	• Used to read single character from keyboard but doesn't display that character on console & immediately returned without pressing Enter (conio.h)
getche() Syntax Variable_name = getche();	• Used to read single character from keyboard and display that character on console and immediately returned without pressing Enter (conio.h)
~~getch()~~ getchar. Syntax Variable_name = ~~getch()~~ getchar()	• reads one character until and unless enter key is pressed (stdio.h)
putchar() Syntax: putchar(Variable_name) or putchar('A')	• Display single character by putting character display or by passing variable that has stored character. (stdio).
gets() Syntax: gets(string_variable) or	• Used to input string. (stdio.h).
~~Putchar~~ Puts("Programming"); (G). Syntax: Puts(string-variable) or Puts("Programming");	• Used to display string. (stdio.h)