C variable

- C variable is a named location in a memory where a program can manipulate the data. This location is used to hold the value of the variable.
- The value of the C variable may get change in the program.
- C variable might be belonging to any of the data type like int, float, char etc.

Rules for naming C variable:

1. Variable name must begin with letter or underscore.
2. Variables are case sensitive
3. They can be constructed with digits, letters.
4. No special symbols are allowed other than underscore.
5. sum, height, _value are some examples for variable name

Declaring & initializing C variable:

- Variables should be declared in the C program before to use.
- Memory space is not allocated for a variable while declaration. It happens only on variable definition.
- Variable initialization means assigning a value to the variable.

S.No	Type	Syntax	Example
1	Variable declaration	data_type variable_name;	int x, y, z; char flat, ch;
2	Variable initialization	data_type variable_name = value;	int x = 50, y = 30; char flag = 'x', ch='l';

There are three types of variables in C program They are,

1. Local variable
2. Global variable
3. Environment variable

1. Example program for local variable in C:

- The scope of local variables will be within the function only.
- These variables are declared within the function and can't be accessed outside the function.
- In the below example, m and n variables are having scope within the main function only. These are not visible to test function.
- Like wise, a and b variables are having scope within the test function only. These are not visible to main function.

```
#include<stdio.h>
void test();
int main()
{
int m = 22, n = 44;
// m, n are local variables of main function
/*m and n variables are having scope
within this main function only.
These are not visible to test funtion.*/
/* If you try to access a and b in this function,
you will get 'a' undeclared and 'b' undeclared error */
printf("\nvalues : m = %d and n = %d", m, n);
```

```
test();
}

void test()
{
int a = 50, b = 80;
// a, b are local variables of test function
/*a and b variables are having scope
within this test function only.
These are not visible to main function.*/
/* If you try to access m and n in this function,
you will get 'm' undeclared and 'n' undeclared
error */printf("\nvalues : a = %d and b = %d", a, b);
}
```

Output:

```
values : m = 22 and n = 44

values : a = 50 and b = 80
```

2. Example program for global variable in C:

- The scope of global variables will be throughout the program. These variables can be accessed from anywhere in the program.
- This variable is defined outside the main function. So that, this variable is visible to main function and all other sub functions.

```
#include<stdio.h>
void test();int m = 22, n = 44;
int a = 50, b = 80;
int main()
{
printf("All variables are accessed from main function");
printf("\nvalues: m=%d:n=%d:a=%d:b=%d", m,n,a,b);
test();
}
void test()
{
printf("\n\nAll variables are accessed from" \
" test function");
printf("\nvalues: m=%d:n=%d:a=%d:b=%d", m,n,a,b);
}
```

Output:

```
All variables are accessed from main function
values : m = 22 : n = 44 : a = 50 : b = 80


All variables are accessed from test function
values : m = 22 : n = 44 : a = 50 : b = 80
```

3. Environment variables in C:

- Environment variable is a variable that will be available for all C applications and C programs.
- We can access these variables from anywhere in a C program without declaring and initializing in an application or C program.
- The inbuilt functions which are used to access, modify and set these environment variables are called environment functions.

- There are 3 functions which are used to access, modify and assign an environment variable in C. They are,

1. setenv()
2. getenv()
3. putenv()

Example program for getenv() function in C:

This function gets the current value of the environment variable. Let us assume that environment variable DIR is assigned to “/usr/bin/test/”.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
printf(“Directory = %s\n”,getenv(“DIR”));
return 0;
}
```

Output:

```
/usr/bin/test/
```

Example program for setenv() function in C:

This function sets the value for environment variable. Let us assume that environment variable “FILE” is to be assigned “/usr/bin/example.c”

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
setenv(“FILE”, “/usr/bin/example.c”,50);
printf(“File = %s\n”, getenv(“FILE”));
return 0;
}
```

Output:

```
File = /usr/bin/example.c
```

Example program for putenv() function in C:

This function modifies the value for environment variable. Below example program shows that how to modify an existing environment variable value.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
setenv(“DIR”, “/usr/bin/example/”,50);
printf(“Directory name before modifying = ” \
“%s\n”, getenv(“DIR”));
putenv(“DIR=/usr/home/”);
printf(“Directory name after modifying = ” \
“%s\n”, getenv(“DIR”));
return 0;
}
```

Output:

```
Directory name before modifying = /usr/bin/example/
Directory name after modifying = /usr/home/
```

Difference between variable declaration & definition in C:

S.no	Variable declaration	Variable definition
1	Declaration tells the compiler about data type and size of the variable.	Definition allocates memory for the variable.
2	Variable can be declared many times in a program.	It can happen only one time for a variable in a program.
3	The assignment of properties and identification to a variable.	Assignments of storage space to a variable.