



# GOVERNMENT OF ANDHRA PRADESH COMMISSIONERATE OF COLLEGIATE EDUCATION



## Scope of Variables Programming in C Computer Science

**Mr. P Raghavender Reddy** M.Sc, M.Tech

Govt. College for Men (A), Kadapa

Email. Id : prr712@gmail.com

Learn more at <http://ccelms.ap.gov.in>



# Contents

---



- ❖ What is an Variable?
  - ❖ Where Variables are Declared?
  - ❖ What is Scope of a Variable?
  - ❖ Types of Scopes
  - ❖ Example Programs
-



# Learning Objects

---



- ❖ Understand the need of a variable in a program
  - ❖ Know the different regions in a program for declaring a variable
  - ❖ Understand the accessibility or visibility region of a variable in a program
  - ❖ Declare the variables in different place based their use of region
-



# What is a Variable?

- **Variable** is a named memory location that have a type
- Before using a variable for computation, it has to be
  - **Declare** – name an object (gives a symbolic name)
  - **Define** – create an object (allocate memory)
  - **Initialize** – assign data or store data
- With one exception (**extern variable**), a variable is declared and defined at the same time.
- **Single syntax** for declaration and definition of a variable.



# Creation of Variable

## Syntax for Variable Declaration & Definition

**Data\_Type Variable\_List ;**

**Examples:** char code; int roll\_no; double area, side;

## Syntax for Variable Initialization

**Variable\_name = Expression ;**

**Examples:** code = 'B'; roll\_no = 532; area = side\*side;

## Syntax for Variable Declaration, Definition & Initialization

**Data\_Type Variable\_name = Expression ;**

**Examples:** char code = 'B'; double area, side=10.5;



# Creation of Variable

## Data Type

- Memory Size
- Range of Values
- Set of Operations

## Name of a Variable

- An identifier

## Name of Location

## Value

## Declaration

`char code = 'B' ;`

## Initialization

- Assigned a value
- Stored a value
- Use an expression

## Memory Location

**code**

**B**

0x2FA5

Address



# Where Variables are Declared?

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

```
void fact(int p);
```

```
int f = 1;
```

```
int main()
```

```
{
```

```
    int n=5;
```

```
    fact(n);
```

```
    printf(" %d! = %d", n, f);
```

```
    return 0;
```

```
}
```

```
void fact(int m)
```

```
{
```

```
    for(int i=1; i<=m; i++)
```

```
        f = f * i;
```

```
}
```

In a function prototype

Outside of all functions

Inside of main function block

In a function Definition

Inside of for loop block





# Types Variables

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

```
void fact(int p);  
int f = 1;
```

Global (External) variable

```
int main()
```

```
{
```

```
    int n=5;
```

```
    fact(n);
```

```
    printf(" %d! = %d", n, f);
```

```
    return 0;
```

```
}
```

Local (Automatic, Internal) variables

```
void fact(int m)
```

```
{
```

```
    for(int i=1; i<=m; i++)
```

```
        f = f * i;
```

```
}
```

for block

main block

fact block





# What is a Scope of a Variable?



- Scope of variable is a region (set of statements) of program over which the variable is accessible or visible.
- Variable visible within its scope & invisible outside its scope
- C defines 5 scopes that determines the visibility of a variable
  1. Block Scope
  2. Function Prototype Scope
  3. Function Scope
  4. Program Scope
  5. File Scope



# Block Scope



## ➤ Variables declared

- **Inside { } block**
- **In for loop**
- **In function header**

Have block scope

## ➤ Block scope variables are accessible or visible

- **From beginning of "{" or declaration**
- **To the end of "}" or loop**

Scope of **n**

Scope of **m**

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

```
void fact(int p);  
int f = 1;
```

```
int main()  
{
```

```
    int n=5;  
    fact(n);  
    printf("\n %d! = %d", n, f);  
    return 0;  
}
```

```
void fact(int m)  
{
```

```
    for(int i=1; i<=m; i++)  
        f = f * i;
```

Scope of **i**



# Function Prototype Scope

## ➤ Variables declared

- In function prototype

Have function prototype scope

function prototype or function declaration

## ➤ Function prototype scope variables are accessible or visible

- Within function prototype

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

```
void fact(int p);
```

```
int f = 1;
```

```
int main()
```

```
{
```

```
    int n=5;
```

```
    fact(n);
```

```
    printf("\n %d! = %d", n, f);
```

```
    return 0;
```

```
}
```

```
void fact(int m)
```

```
{
```

```
    for(int i=1; i<=m; i++)
```

```
        f = f * i;
```

```
}
```



# Function Scope



- Labels declared with "goto" have function scope

- Labels are accessible or visible
  - From beginning of "{"
  - To the end of "}"

Scope of **loop** label

```
#include<stdio.h>

int main()
{
    printf("Start");
    loop:
    printf("Start");
    printf("Start");
    goto loop;
    printf("Start");
    return 0;
}
```



# Program Scope & File Scope



- Variables declared outside of all functions have program scope
- Accessible or visible throughout the execution of program
- Variables declared outside of all functions with "static" storage class have file scope
- Accessible or visible throughout the entire of file in which its declared

```
#include<stdio.h>

void fact(int p);
int i ;    // i has program scope
static int f =1; // f has file scope

int main()
{
    int n=5;    fact(n);
    printf("\n %d! = %d", n, f);
    return 0;
}

void fact(int m)
{
    for(int i=1; i<=m; i++)
        f = f * i;
}
```



# Summary



Scope	<ul style="list-style-type: none"><li>▪ a region (set of statements) of program over which the variable is accessible or visible.</li></ul>
Block	<ul style="list-style-type: none"><li>▪ Declared inside a block, a loop, or in a function definition</li><li>▪ Accessible from beginning "{" to end "}" of a block.</li></ul>
Function Prototype	<ul style="list-style-type: none"><li>▪ Declared in a function prototype.</li><li>▪ Accessible within a function prototype.</li></ul>
Function	<ul style="list-style-type: none"><li>▪ Only "goto" label has function scope.</li><li>▪ Accessible from beginning "{" to end "}" of a function.</li></ul>
Program	<ul style="list-style-type: none"><li>▪ Declared outside of all functions.</li><li>▪ Accessible throughout execution of a program.</li></ul>
File	<ul style="list-style-type: none"><li>▪ Declared outside of all functions with "static" keyword.</li><li>▪ Accessible throughout the entire file.</li></ul>



# References



- E Balagurusamy – Programming in ANSIC – Tata McGraw-Hill
- Reema Thareja – Introduction to C Programming – Oxford University Press
- Pradip Dey, Manas Ghosh – Programming in C – Oxford University Press
- Brain W Kernighan, Dennis M Ritchie – The 'C' Programming Language – Pearson
- Jeri R Hanly, Elliot B Koffman – Problem Solving and Program Design in C – Pearson





---

# Thank You



P Raghavender Reddy

[pr712@gmail.com](mailto:pr712@gmail.com)

**9441762630**

---