greatlearning
Power Ahead

Introduction to C

greatlearning
Power Ahead

Agenda

- Introduction
- C Syntax
- Comments
- Tokens
- Variables
- User Input/Ouput
- Data Types
- Operators
- Math
- Boolean
- Conditional Statements
- Switch Case
- Loops
- Array
- Functions

greatlearning
Power Ahead

Introduction

- Developed by Dennis Ritchie.
- Robust low level programming level language.
- Procedural language which follows structure.
- Supports various operating systems and hardware platforms.
- Many compilers are available for executing programs written in 'C'.
- Executable creation from a C -program is a two step process:
    1) Compilation
    2) Linking

C Syntax

Various Components in a C program

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

void main(){

printf("Hello World");

getch();
}
```

Preprocessor directive

Execution begins here

Pushing output to console

Termination after any key is pressed

Comments in C

What?

Comments are lines ignored by the compiler while compilation.

Why?

To make notes, provide details etc.

How?

Single Line --- // This is a single line comment

Multi Line -- /*This is a multi-line comment */

Tokens

C tokens are smallest individual units in a program.
C supports following tokens:

- Identifiers- user defined names/abbreviations
- Keywords - words which are specific to C language
- Constants - values which don't change
- Strings - sequence of characters
- Operators - act on operands and generates output
- Special Symbols - used for preprocessor directives (#)

Variables

What?

Memory references whose value changes during execution

Why?

Makes memory handling easy

Make reuse of same memory locations for storing different values Makes referencing of values easier

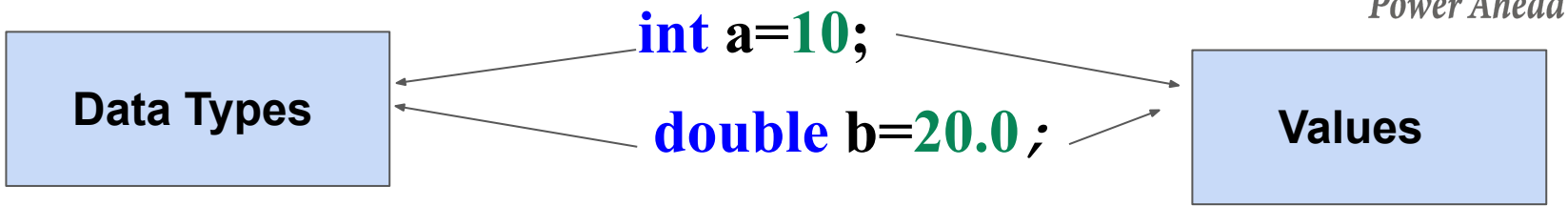
Syntax:

Initialization

Data_type [variable_name];

Assignment

[variable_name]=value;




```
#include <stdio.h>
 #include <conio.h>
 int main()
{
   int a=10;  double b=20.0;
   printf("%d + %lf = %lf",a,b,a+b);
   return 0;
   getch();
}
```

greatlearning
Power Ahead

User Input/Output

C supports the following functions for user input/output.

- printf() and scanf()
- getchar() and putchar()
- gets() and puts()

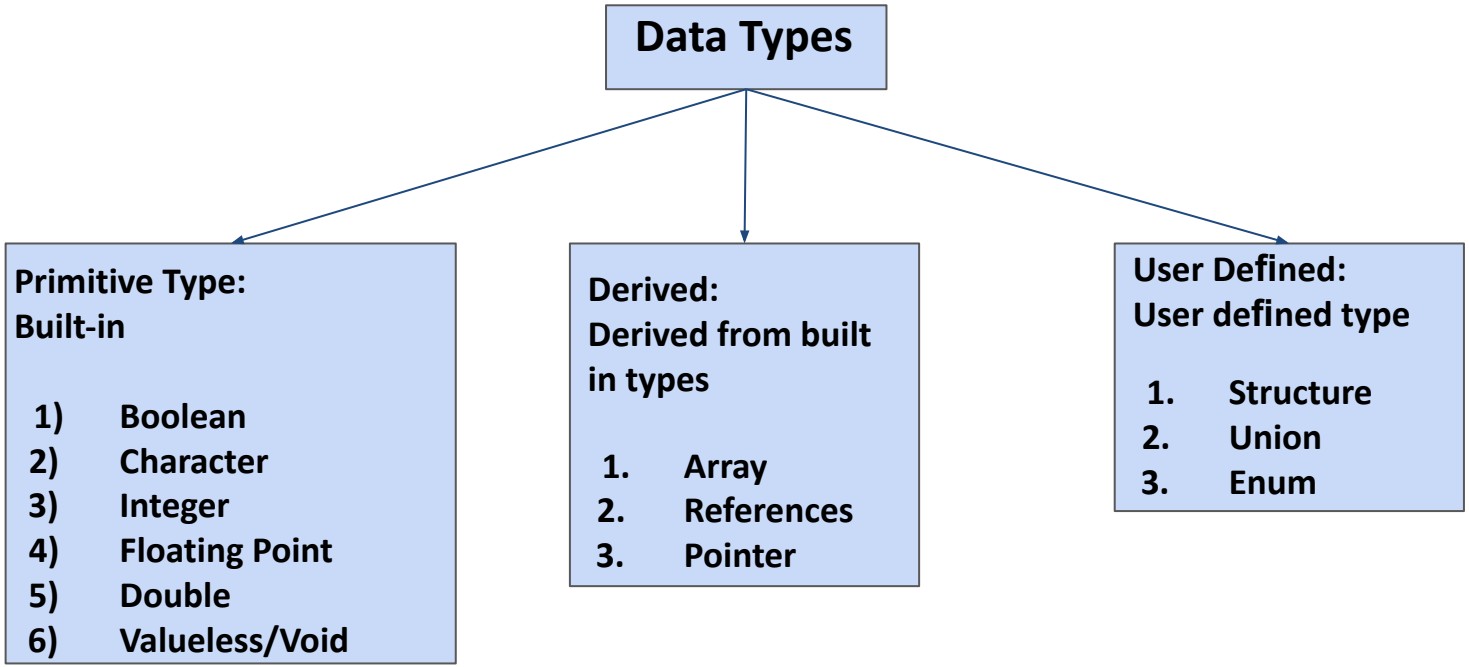
greatlearning
Power Ahead

Format Strings

This informs the scanf() function, what type of input to expect and printf() is used to give a heads up to the compiler, what type of output to expect.

%d	Integer
%f	Floating point
%c	Character
%s	String

C Data Types



Operators

C supports primarily 5 types of operators :

1. **Arithmetic Operators**
   - Used for mathematical and arithmetic operators.
   - There are two types of arithmetic operators:
   - Unary Operators -
     - Operators that operates or works with a single operand are unary operators.
     - For example: (++ , - -)
   - Binary Operators -
     - Operators that operates or works with two operands are binary operators.
     - For example: (+ , – , * , /)

Operators

2. **Relational Operator**
   - Used for comparison of the values of two operands.
   - For example, checking for equality of operands, whether an operand is greater than the other operand or not etc.
   - Some of the relational operators are (==, >= , <=).

3. **Logical Operator**
   - Used to combine two or more conditions/constraints.
   - Result of the operation of a logical operator is a boolean value either true or false.
   - Some logical operators are (AND(&&),OR(||),NOT(!)).

Operators

4. **Bitwise operator**
   - Used to perform bit-level operations on the operands.
   - Operators are first converted to bit-level and then the calculation is performed on the operands.
   - Mathematical operations such as addition, subtraction, multiplication etc can be performed at bit-level for faster processing.
5. **Assignment Operator**
   - Used for value assignment to a variable.
   - Left side operand of the assignment operator is a variable and right side operand of the assignment operator is a value.
   - The value on the right side must be of the same data-type of variable on the left side otherwise the compiler will raise an error.
   - Example: +=, -=,*=,/=

Strings

Strings are used to store sequence of characters as information.Strings can be used by importing the ***"string"*** library.

- **Concatenation**

Here Great and Learning are two strings which are concatenated.

String Functions

String functions are accessible by importing <string.h>.

- strlen() Find length of string
- strcpy() Copy one string to other
- strcat() Join two strings
- strcmp() Compare two strings
- strlwr() Converts string to lowercase
- strupr() Converts string to uppercase

String Functions Demo

greatlearning
Power Ahead

```
#include<stdio.h>
#include<string.h>
 int main()
{
  char s1[30];
   char s2[30];
   gets(s1);
   gets(s2);

  printf("%ld \n",strlen(s1));
  printf("%s \n",strcat(s1,s2));
  printf("%ld \n",strcmp(s1,s2));
  printf("%s \n",strlwr(s1));
  printf("%s \n",strupr(s1));
  printf("%s \n",strcpy(s1,s2));
  return 0;
  }
```

Math Library

- C supports large number of mathematical functions.
- These functions are for various mathematical calculations.
- These functions can be directly used in programs.
- We need to include <math.h> to use these functions which is a C library.

Math Library Demo

greatlearning
Power Ahead

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
int main(){
   short int a = 100;
   int b = -1000;
   long int c = 1300;
   float d= 230.47;
   double e = 200.347;
   printf("sqrt(si): %d \n",sqrt(a));
   printf("pow(li, 3): %d \n",pow(b, 3));
   printf("sin(d): %d \n",sin(c));
   printf("abs(i) : %d \n",abs(d));
   printf("floor(d): %d \n",floor(e));
   printf("sqrt(f): %d \n",sqrt(d));
   printf("pow(d, 2): %d \n",pow(d, 2));
   return 0;
}
```

greatlearning
Power Ahead

Boolean

- The Boolean data type is used to declare a variable whose value will be set as true (1) or false (0).
- To declare such a value, you use the **bool** keyword.
- The variable can then be initialized with the starting value.
- A Boolean constant is used to check the state of a variable, an expression, or a function, as true or false.

Boolean Demo

```
#include <stdio.h>

int main()

{

   bool machineIsWorking = true;

   printf("Since this machine is working, its value is %d \n", machineIsWorking);

   machineIsWorking= false;

   printf("The machine has stopped operating. \n Now its value is %d \n", machineIsWorking);

   return 0;

}
```

greatlearning
Power Ahead

Conditional Statements

- Conditional statements are also known as selection statements.
- They are used to make decisions based on a given condition.
- If the condition evaluates to True, a set of statements is executed, otherwise another set of statements is executed.
- Conditional statements evaluate to a boolean value.

- **The if Statement:**

  Selection and execution of statement(s) based on a given condition is done by the if statement. If the condition evaluates to True then a given set of statement(s) is executed. However, if the condition evaluates to False, then the given set of statements is skipped and the program control passes to the statement following the if statement.

  Syntax:

```
if (condition)
{
 statement 1;
 statement 2;
 statement 3;
}
```

- **The if-else Statement:**
  The if - else statement causes one of the two possible statement(s) to execute, depending upon the outcome of the condition.

```
if (condition)     //if part
{
  statement1;
  statement2;
}
else   //else part
 statement 3;
```

Here, the if-else statement comprises two parts, namely, if and else. If the condition is True the if part is executed. However, if the condition is False, the else part is executed.

- **Nested if-else Statement:**
  A nested if-else statement contains one or more if-else statements. The if else can be nested in three different ways, which are discussed here.

  The if - else statement is nested within the if part.

```
if (condition)
{
   Statement 1;
   if(condition2)
        Statement 2;
   else
        Statement 3;
}
else
        Statement 4;
```

Example : A code segment to determine the largest of three numbers

```
if (a>b){
        if (a>c)
        printf(“a is largest”);
}
else{
        if (b>c)
        // nested if-else statement within else
        printf(“b is largest”);
        else
        printf(“c is largest”);
}
```

greatlearning
Power Ahead

Switch Case

- The switch statement works as a multiway branch statement.
- It’s a modified form of if-else.
- It’s a common alternative to the if statement when you want to get multiple results.
- Default condition gets executed when no conditions are met.

greatlearning
Power Ahead

Switch Case Demo

```
#include <stdio.h>
int main() {
    int e = 1;
    switch (e) {
    case 1:  break;
    case 2:  break;
    default:  break;
   }
return 0;
}
```

Loops

Loops are used to re-execute a part of code a given number of times, depending upon the condition specified.

Entry Controlled

The condition is checked each-time before entering the loop. If the condition is satisfied then only the loop body gets executed. The loop body does not get executed if the condition is false in the first iteration.

Entry controlled loops are of following types:

1) **For loop**

Syntax:

```
for(i=0;i<n;i++){
        //do something
}
```

2) **While Loop**

Syntax:

```
while(condition is True){
        //do something
}
```

Array

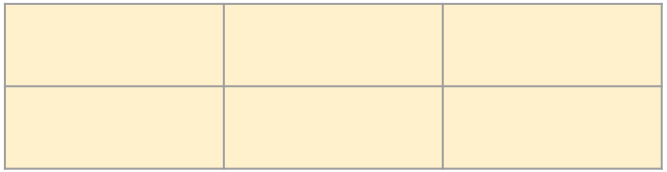
- An array is a collection of elements of similar data type which are stored in contiguous memory locations.
- Every element in an array has a specific index.
- Size of array in C is fixed at the time of definition.

Array's are of two types:

1. Single dimensional

1x4

1. Multi-dimensional



2 X 3

greatlearning
Power Ahead

Definition:

An array can be defined in the following manner in C:

Data_type <name_of_array>[<size_of_array>]

Size of an array must be an integer as it shows the number of elements in an array which cannot be a real number.

Syntax:

```
int a[10];          //single dimensional array
double b[10][10];   //double - dimensional array
```

greatlearning
Power Ahead

Array initialization:

1) Static Initialization

```
int a[10]={0,1,2,3,4,5,6,7,8,9,10}
char c[5]={'h','e','l','l','o'}
```

2) Dynamic Initialization

```
int a[10]; //array is created with garbage value
for (int i=0;i<n;i++)
        cin>>a[i];
```

greatlearning
Power Ahead

Pointers in C

- Pointers are variables which store the address of a variable.
- They have data type just like variables, for example an integer type pointer can hold the address of an integer variable and an character type pointer can hold the address of char variable.
- Example:

```
int a =10;  int
*p=&a;
```

Pointers Demo

```
#include <stdio.h>

int main () {

        int a = 20;  int *p;
        p = &a;
        printf("Address of a : %x\n", &a );
        printf("Content of p: %x\n", p );
        printf("Content of *p: %d\n", *p );
        return 0;
}
```

greatlearning
Power Ahead

Functions

- Functions are blocks of code which are used to perform specific tasks .
- In C a function needs to be declared before it’s used.
- Functions have a function definition , function body and a return type.
- Functions with return type except void needs to return a value at the end.
- Functions with return type void do not return any value.

Function Demo

```
#include <stdio.h>
int add(int x,int y){
    return x+y;
}

int main() {
        printf("%d",add(3,4));
        return 0;
}
```

greatlearning
Power Ahead

Thank You