SNS COLLEGE OF TECHNOLOGY

Coimbatore-35.
An Autonomous Institution

COURSE NAME : 19CST101 PROGRAMMING FOR PROBLEM SOLVING

I YEAR/ I SEMESTER

UNIT-II C PROGRAMMING BASICS

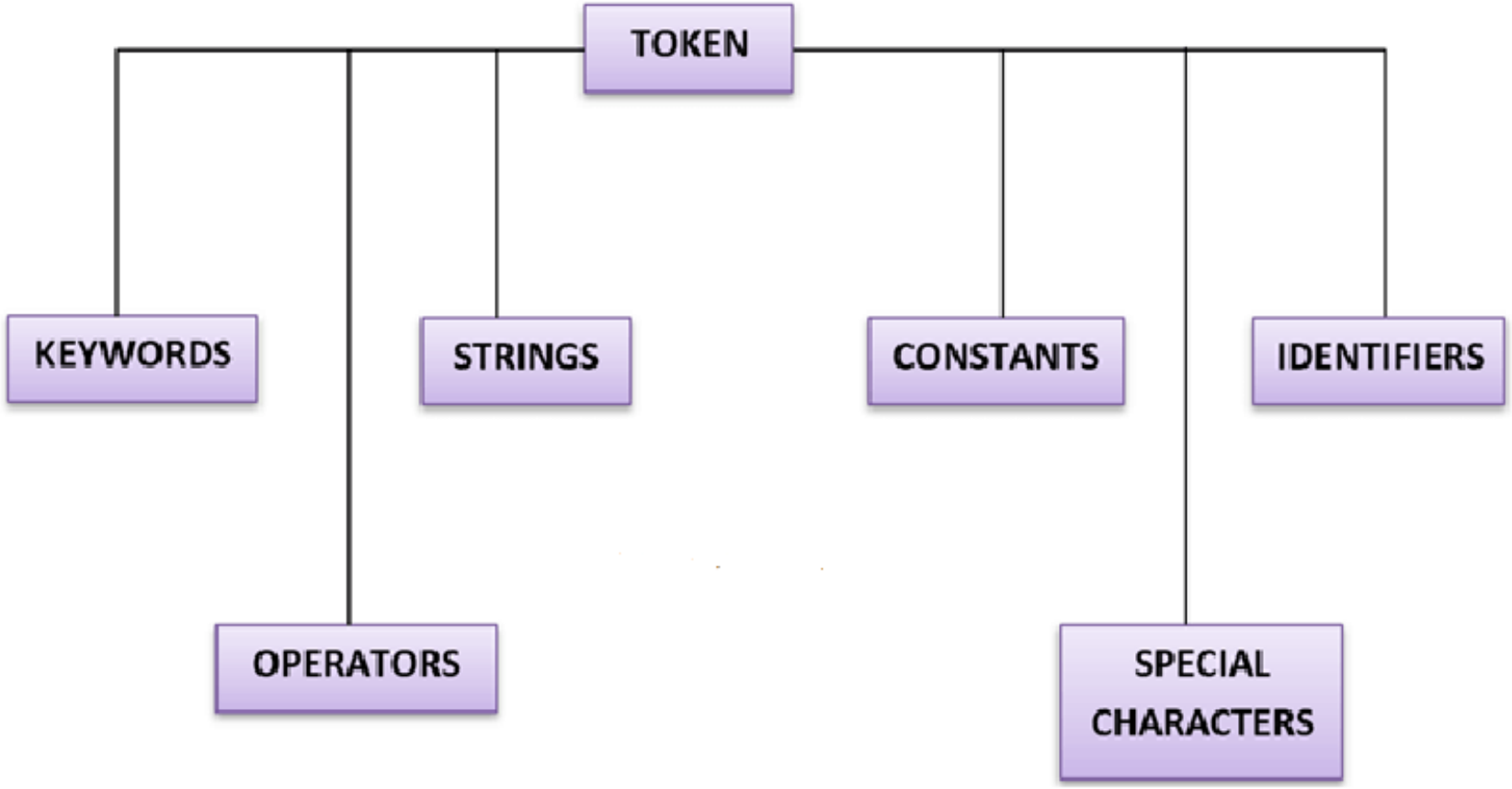
Topic: C Tokens

Mrs. S.R.JANANI
Assistant Professor
Department of Computer Science and Engineering

C Tokens

- In C programs, each word and punctuation is referred to as a token.
- C Tokens are the smallest building block or smallest unit of a C program.
- The compiler breaks a program into the smallest possible units and proceeds to the various stages of the compilation, which is called token.
- C Supports Six Types of Tokens:
  1. Identifiers
  2. Keywords
  3. Constants
  4. Strings
  5. Operators
  6. Special Symbols

C Tokens


TOKEN
KEYWORDS
STRINGS
CONSTANTS
IDENTIFIERS
OPERATORS
SPECIAL CHARACTERS

Character Set

- A character set is a set of alphabets, letters and some special characters that are valid in C language.

Alphabets

```
Uppercase: A B C ..................................... X Y Z
Lowercase: a b c ..................................... x y z
```

- C accepts both lowercase and uppercase alphabets as variables and functions.

Digits

```
0 1 2 3 4 5 6 7 8 9
```

Character Set

Special Characters

Special Characters in C Programming				
,	<	>	.	_
(	)	;	$	:
%	[	]	#	?
'	&	{	}	"
^	!	*	/	|
-	\	~	+	

White Space Characters

- Blank space, newline, horizontal tab, carriage return and form feed.

C Keywords

- In 'C' every word can be either a keyword or an identifier.
- A keyword is a **reserved word**.
- Keywords have fixed meanings, and the meaning cannot be changed.
- You cannot use it as a variable name, constant name, etc.
- They act as a building block of a 'C' program.
- There are a total of 32 keywords (reserved words) in 'C'.
- Keywords are written in lowercase letters.

C Keywords

- For example:

```
int money;
```

Here, `int` is a keyword that indicates `money` is a variable of type `int` (integer).

- Here is a list of all keywords allowed in ANSI C.

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	short	float	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

C Identifiers

- An identifier is nothing but a name assigned to an element in a program.
- Example, name of a variable, function, etc.
- Identifiers are the user-defined names consisting of 'C' standard character set.
- As the name says, identifiers are used to identify a particular element in a program.
- Each identifier must have a unique name.
- For Example:

```
int money;
double accountBalance;
```

Here, `money` and `accountBalance` are identifiers.

C Identifiers

Rules for constructing C identifiers:

- An identifier can only have alphanumeric characters (a-z , A-Z , 0-9) (i.e. letters & digits) and underscore(_) symbol.
- The first character of an identifier should be either an alphabet or an underscore, and then it can be followed by any of the character, digit, or underscore.
- It should not begin with any numerical digit.
- In identifiers, both uppercase and lowercase letters are distinct. Therefore, we can say that identifiers are case sensitive.
- Commas or blank spaces cannot be specified within an identifier.
- Keywords cannot be represented as an identifier.
- The length of the identifiers should not be more than 31 characters.
- Identifiers should be written in such a way that it is meaningful, short, and easy to read.

C Identifiers

- **Example of valid identifiers**

```
total, sum, average, _m _, sum_1, etc.
```

- **Example of invalid identifiers**

```
2sum (starts with a numerical digit)
int (reserved word)
char (reserved word)
m+n (special character, i.e., '+')
```

C Identifiers

Types of identifiers:

- Internal identifier
- External identifier

Internal Identifier

- If the identifier is not used in the external linkage, then it is known as an internal identifier. The internal identifiers can be local variables.

External Identifier

- If the identifier is used in the external linkage, then it is known as an external identifier. The external identifiers can be function names, global variables.

Differences between Keyword and Identifier

Keyword	Identifier
Keyword is a pre-defined word.	The identifier is a user-defined word
It must be written in a lowercase letter.	It can be written in both lowercase and uppercase letters.
Its meaning is pre-defined in the c compiler.	Its meaning is not defined in the c compiler.
It is a combination of alphabetical characters.	It is a combination of alphanumeric characters.
It does not contain the underscore character.	It can contain the underscore character.

Variables in C

- Variables are memory locations(storage area) in the C programming language.
- The primary purpose of variables is to store data in memory for later use.
- Unlike constants which do not change during the program execution, variables value may change during execution.
- If you declare a variable in C, that means you are asking the operating system to reserve a piece of memory with that variable name.

Variable Declaration:

- Syntax

```
type variable_name;
```

or

```
type variable_name, variable_name, variable_name;
```

Variables in C

Variable Declaration and Initialization :

- Example

```
int    width, height=5;
char   letter='A';
float  age, area;
double d;

/* actual initialization */width = 10;
age = 26.5;
```

Variables in C

Variable Assignment:

- A variable assignment is a process of assigning a value to a variable.
- Example

```
int width = 60;
int age = 31;
```

Variables in C

Rules for defining variables:

- A variable name can consist of Capital letters A-Z, lowercase letters a-z, digits 0-9, and the underscore character.
- The first character must be a letter or underscore.
- Blank spaces cannot be used in variable names.
- Special characters like #, $ are not allowed.
- C keywords cannot be used as variable names.
- Variable names are case sensitive.
- Values of the variables can be numeric or alphabetic.
- Variable type can be char, int, float, double, or void.

Variables in C

Types of Variables in C:

- There are many types of variables in c:
  1. Local Variable
  2. Global Variable
  3. Static Variable
  4. Automatic Variable
  5. External Variable

Types of Variables in C

Local Variable

A variable that is declared inside the function or block is called a local variable.

It must be declared at the start of the block.

```
void function1(){
int x=10;//local variable
}
```

You must have to initialize the local variable before it is used.

Types of Variables in C

Global Variable

A variable that is declared outside the function or block is called a global variable. Any function can change the value of the global variable. It is available to all the functions.

It must be declared at the start of the block.

```
int value=20;//global variable
void function1(){
int x=10;//local variable
}
```

Types of Variables in C

Static Variable

A variable that is declared with the static keyword is called static variable.

It retains its value between multiple function calls.

```
void function1(){
int x=10;//local variable
static int y=10;//static variable
x=x+1;
y=y+1;
printf("%d,%d",x,y);
}
```

If you call this function many times, the **local variable will print the same value** for each function call, e.g, 11,11,11 and so on. But the **static variable will print the incremented value** in each function call, e.g. 11, 12, 13 and so on.

Types of Variables in C

Automatic Variable

All variables in C that are declared inside the block, are automatic variables by default. We can explicitly declare an automatic variable using **auto keyword.**

```
void main(){
int x=10;//local variable (also automatic)
auto int y=20;//automatic variable
}
```

Types of Variables in C

External Variable

We can share a variable in multiple C source files by using an external variable. To declare an external variable, you need to use **extern keyword.**

myfile.h

```
extern int x=10;//external variable (also global)
```

program1.c

```
#include "myfile.h"
#include <stdio.h>
void printValue(){
    printf("Global variable: %d", global_variable);
}
```

Variables in C

C Program to Print Value of a Variable

Example:

```
#include<stdio.h>

void main()
{
    /* c program to print value of a variable */    int age = 33;
    printf("I am %d years old.\n", age);
}
```

Program Output:

```
I am 33 years old.
```

C Constants

- Constants are like a variable, except that their value never changes during execution once defined.
- C Constants is the most fundamental and essential part of the C programming language.
- Constants in C are the fixed values that are used in a program, and its value remains the same during the entire execution of the program.
- Constants are also called literals.
- Constants can be any of the data types.
- It is considered best practice to define constants using only upper-case names.

Constant Definition in C

Syntax:

```
const type constant_name;
```

`const keyword` defines a constant in C.

Example:

```
#include<stdio.h>
main()
{
  const int SIDE = 10;
  int area;
  area = SIDE*SIDE;
  printf("The area of the square with side: %d is: %d sq. units"
  , SIDE, area);
}
```

C Constants

Constant Types in C

Constants are categorized into two basic types, and each of these types has its subtypes/categories. These are:

Primary Constants

1. Numeric Constants
   - Integer Constants
   - Real Constants
2. Character Constants
   - Single Character Constants
   - String Constants
   - Backslash Character Constants

C Constants

Integer Constant

It's referring to a sequence of digits. Integers are of three types viz:

1. Decimal Integer
2. Octal Integer
3. Hexadecimal Integer

Example:

15, -265, 0, 99818, +25, 045, 0X6

Real constant

The numbers containing fractional parts like 99.25 are called real or floating points constant.

C Constants

Single Character Constants

It simply contains a single character enclosed within ' and ' (a pair of single quote). It is to be noted that the character '**8**' is not the same as **8**. Character constants have a specific set of integer values known as ASCII values (American Standard Code for Information Interchange).

Example:

'X', '5', ';'

String Constants

These are a sequence of characters enclosed in double quotes, and they may include letters, digits, special characters, and blank spaces. It is again to be noted that "**G**" and '**G**' are different - because "G" represents a string as it is enclosed within a pair of double quotes whereas 'G' represents a single character.

Example:

"Hello!", "2015", "2+1"

C Constants

Backslash character constant

C supports some character constants having a backslash in front of it. The lists of backslash characters have a specific meaning which is known to the compiler. They are also termed as "Escape Sequence".

For Example:

`\t` is used to give a tab

`\n` is used to give a new line

C Constants

Backslash character constant

Constants	Meaning
\a	beep sound
\b	backspace
\f	form feed
\n	new line
\r	carriage return
\t	horizontal tab
\v	vertical tab
\'	single quote
\"	double quote
\\	backslash
\0	null

C Constants

Two ways to define constant in C:

- There are two ways to define constant in C programming.
  - const keyword
  - #define preprocessor

C Constants

1) C const keyword

The const keyword is used to define constant in C programming.

```
const float PI=3.14;
```

Now, the value of PI variable can't be changed.

```
#include<stdio.h>
int main(){
    const float PI=3.14;
    printf("The value of PI is: %f",PI);
    return 0;
}
```

Output:

```
The value of PI is: 3.140000
```

C Constants

If you try to change the the value of PI, it will render compile time error.

```
#include<stdio.h>
int main(){
const float PI=3.14;
PI=4.5;
printf("The value of PI is: %f",PI);
   return 0;
}
```

Output:

```
Compile Time Error: Cannot modify a const object
```

C Constants

2) C #define preprocessor

The #define preprocessor is also used to define constant.

- By using the **#define** pre-processor directive which doesn't use memory for storage and without putting a semicolon character at the end of that statement

```
#include <stdio.h>
#define PI 3.14
int main() {
printf("%f", PI);
return 0;}
```

C Delimiters

- These are the symbols which has some syntactic meaning and has got significance.
- These will not specify any operations.
- These cannot be used for some other purpose.
- C language delimiters list is show below.

SYMBOL	NAME	MEANING
#	Hash	Pre processor directive
,	Comma	Variable delimiter used to separate
:	Colon	Label delimiters
;	Semi colon	Statement delimiters
()	Parenthesis	Used in expressions or in function
{}	Curly braces	Used for blocking c structure
[]	Square braces	Used along with arrays


Thank You!