PDS Lab
Section 16
Autumn-2018

Tutorial 2

Language Elements in C

What are the different elements you can find in the following program?

```
# include< stdio.h>

# define PI_4_BY_3 4. 1887902048

 double radius= 10;

  double volOfSphere ( double r)
{
      return PI_4_BY_ 3 * r * r * r;
}

 main ()
{
     double volume;
    volume =volOfSphere(radius);
     printf(" Radius= %lf ,  volume= %lf.\n" , radius,volume);
}
```

The C language alphabet

- Uppercase letters ‘A’ to ‘Z’
- Lowercase letters ‘a’ to ‘z’
- Digits ‘0’ to ‘9’

- C special characters:

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

- White space character in C

\b	blank space
\t	horizontal tab
\v	vertical tab
\r	carriage return
\f	form feed
\n	new line
\\	Back slash
\’	Single quote
\"	Double quote
\?	Question mark
\0	Null
\a	Alarm (bell)

ASCII	Symbol	ASCII	Symbol	ASCII	Symbol	ASCII	Symbol
0	NUL	16	DLE	32	(space)	48	0
1	SOH	17	DC1	33	!	49	1
2	STX	18	DC2	34	"	50	2
3	ETX	19	DC3	35	#	51	3
4	EOT	20	DC4	36	$	52	4
5	ENQ	21	NAK	37	%	53	5
6	ACK	22	SYN	38	&	54	6
7	BEL	23	ETB	39	'	55	7
8	BS	24	CAN	40	(	56	8
9	TAB	25	EM	41	)	57	9
10	LF	26	SUB	42	*	58	:
11	VT	27	ESC	43	+	59	;
12	FF	28	FS	44	,	60	<
13	CR	29	GS	45	-	61	=
14	SO	30	RS	46	.	62	>
15	SI	31	US	47	/	63	?

ASCII	Symbol	ASCII	Symbol	ASCII	Symbol	ASCII	Symbol
64	@	80	P	96	`	112	p
65	A	81	Q	97	a	113	q
66	B	82	R	98	b	114	r
67	C	83	S	99	c	115	s
68	D	84	T	100	d	116	t
69	E	85	U	101	e	117	u
70	F	86	V	102	f	118	v
71	G	87	W	103	g	119	w
72	H	88	X	104	h	120	x
73	I	89	Y	105	i	121	y
74	J	90	Z	106	j	122	z
75	K	91	[	107	k	123	{
76	L	92	\	108	l	124	|
77	M	93	]	109	m	125	}
78	N	94	^	110	n	126	~
79	O	95	_	111	o	127	

C language recognizes total 256 ASCII codes; other 128 ASCII codes are for extended characters' symbols

- Keywords
  - Keywords are those words whose meaning is already defined by Compiler; also called "reserved words" and cannot be used in identifier declaration
  - There are 32 keywords in C

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

C is a case-sensitive programming language!

Declaration of Variables

Which one of the following is a valid name of a C variable?

2ab_c

Switch

xy#1

"rst"

x y

case

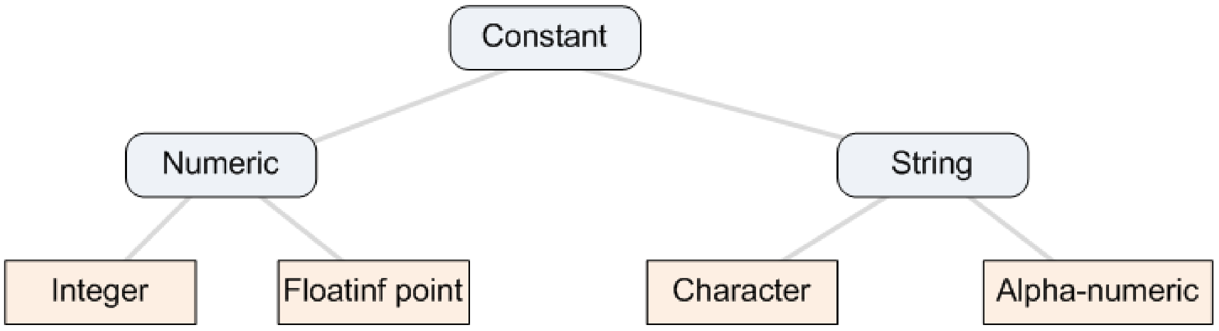
Rules

- Names given to various program elements (variables, constants, functions, etc.)
- May consist of letters, digits and the underscore ('_') character, with no space between.
- Blank and comma are not allowed.
- First character must be an alphabet or underscore.
- An identifier can be arbitrary long.
- Identifier should not be a reserved word.

Note: C is a case sensitive programming language

- 'area', 'AREA' and 'Area' are all different.

Different types of Constants



12345	+596	-137
3.141414	2147483647	– 2147483648
23000000	2.3e7	3.45e23
0.123e-12	1.7E+308	
‘a’	“a”	“IIT Kharagpur”
“14CS10003”		

x = 3.1441 y = ‘a’ name = “Debasis” ?

Data Types in C

Data Types

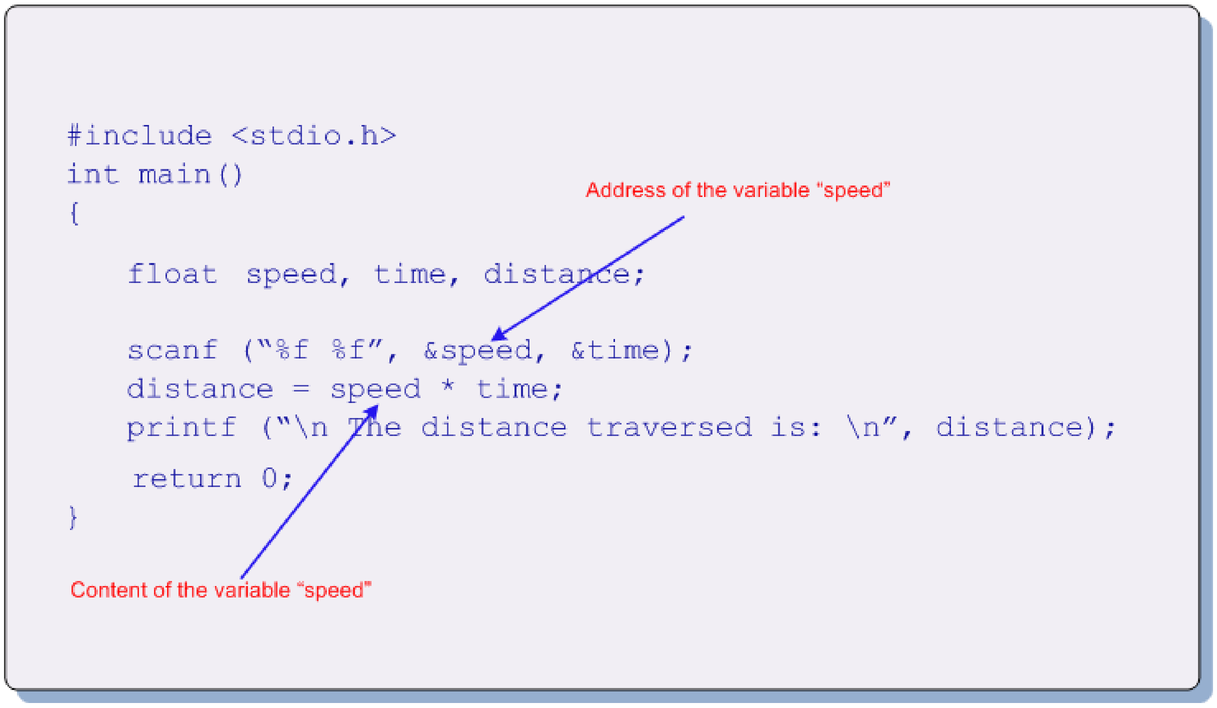
Type	Storage size (in byte)	Value range
char	1	-128 to 127 or 0 to 255
unsigned char	1	0 to 255
signed char	1	-128 to 127
int	2 or 4	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647
unsigned int	2 or 4	0 to 65,535 or 0 to 4,294,967,295
short	2	-32,768 to 32,767
unsigned short	2	0 to 65,535
long	4	-2,147,483,648 to 2,147,483,647
unsigned long	4	0 to 4,294,967,295

float	4	1.2E-38 to 3.4E+38	6 decimal places
double	8	2.3E-308 to 1.7E+308	15 decimal places
long double	10	3.4E-4932 to 1.1E+4932	19 decimal places

What is the meaning?

scanf (“%c%d%f”, &x, &y, &z);

printf (“%c %c %f”, x, y, z);


Address of the variable “speed”
Content of the variable “speed”

```
#include <stdio.h>
int main()
{
    float  speed, time, distance;

    scanf ("%f %f", &speed, &time);
    distance = speed * time;
    printf ("\n The distance traversed is: \n", distance);
     return 0;
}
```

Assignment in C

- Used to assign values to variables, using the assignment operator (=).

- General syntax:
  variable_name = expression;

Examples:

```
velocity = 20;

b = 15; temp = 12.5;

A = A + 10;

v = u + f * t;

s = u * t + 0.5 * f * t * t;
```

- Assignment during declaration

```
int speed = 30;

char flag = ‘y’;
```

- Multiple variable assignment

```
a = b = c = 5;
flag1 = flag2 = ‘y’;
speed = flow = 20.0;
```

- In addition to = operator, C has a set of **shorthand** assignment operators of the form
-

  var_name op = expression;

This is equivalent to

    var_name = var_name op expression;

Examples

$x \mathrel{+}= y+1; \rightarrow x = x + (y+1);$

$x \mathrel{-}= y \rightarrow x = x-y;$

$a \mathrel{*}= a; \rightarrow a = a*a;$

$m \mathrel{\%}= n; \rightarrow m = m\%n;$

Examples:

Given m = 0.1kg, c = 3.0e8 m/sec, then find the energy that will be converted.

$$e = mc^2$$

Calculate *T* given a value of *l* and *g* using the formula

$$T = 2\pi\sqrt{\frac{l}{g}}$$

Operators in C

Arithmetic Operators

Relational Operators

Logical Operators

Arithmetic Operators

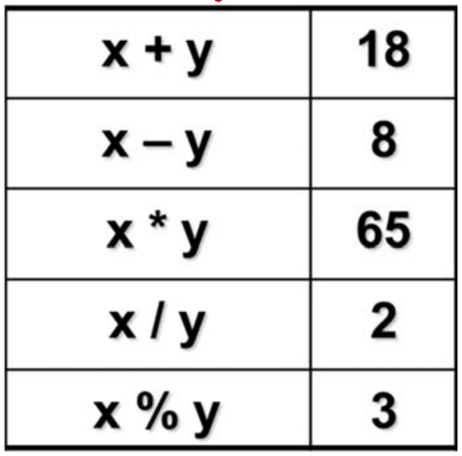
Addition:	+
Subtraction:	-
Multiplication:	*
Division:	/
Modulus:	%

Examples:

```
distance = rate * time ;

netIncome = income - tax ;

speed = distance / time ;

area = PI * radius * radius;

y = a * x * x + b*x + c;

quotient = dividend / divisor;

remain = dividend % divisor;
```

Example

x = 13; y = 5;



x + y	18
x – y	8
x * y	65
x / y	2
x % y	3

Increment and Decrement Operators

Increment operator ++

It adds 1 to its operand

++x; (prefix operator)
x++; (postfix operator)

These are equivalent to x = x + 1;

y = ++x; is equivalent to y = x + 1;

Note:
y = ++x; and y = x++; are different.
++x increments x **before** its value is used, while
x++ increments x **after** its value has been used.

x = 5;	**x**	**y**
y = ++x;	6	6
y = x++;	6	5

Decrement operator --

It subtracts 1 from its operand

--x; (prefix operator)
x--; (postfix operator)

These are equivalent to x = x - 1;

Note: y = x--; is not same as y = --x;

Note: increment (++) and decrement (--) operators are only applicable to variables (integer).

Examples:

(i + j)++; is illegal! This is because (i+j) is not an integer variable name.

Suppose, a = 10, b = 5; Following two in sequence, if executed

c = ++a – b will result c = 6;

c = b-- + a will result c = 16;

Evaluate the following expressions:

3+-5*-2 10

10 - 5 - 7 / 4 * 4

3 > 5 – 2

3 + 5%2 - 1

Relational Operators

<	is less than
>	is greater than
<=	is less than or equal to
>=	is greater than or equal to
==	is equal to
!=	is not equal to

Example:

$a + b > c - d$ is the same as $(a+b) > (c-d)$

Sample code segment in C

```
 if  (x > y)
      printf (“%d is larger\n”, x);
else
      printf (“%d is larger\n”, y);
```

Logical Operators

There are two logical operators in C (also called logical connectives).

&&	→	Logical AND
| |	→	Logical OR
!	→	Logical NOT

What they do?

- They act upon operands that are themselves logical expressions.
- The individual logical expressions get combined into more complex conditions that are true or false.

Example

(a > b) && (c < d) || ((a-b) != (c-d))
results TRUE if a = 5, b = 2, c = 1 and d = 4

Associativity and Precedence of Operators

Operator	Associativity	Precedence
( )	Left to Right	1
- (unary)	Right to Left	2
--, ++		
!, ~		
*, /, %	Left to Right	3
+, -	Left to Right	4
<<, >>	Left to Right	5
<, <=, >, >=	Left to Right	6
== , !=	Left to Right	7
&	Left to Right	8
^	Left to Right	9
|	Left to Right	10
&&	Left to Right	11
||	Left to Right	12
?:	Right to Left	13

Examples:

v = u + f * t; → v = u+(f*t);

X = x * y / z → X = (x*y)/z

A = a + b – c * d / e → A = ((a+b)-((c*d)/e))

A = -b * c + d % e → A = (((-b)*c)+(d%e))

Example:

a + b * c – d / e → a + (b * c) – (d / e)

a * – b + d % e – f → a * (– b) + (d % e) – f

a – b + c + d → (((a – b) + c) + d)

x * y * z → ((x * y) * z)

a + b + c * d * e → (a + b) + ((c * d) * e)

Integer arithmetic

- When the operands in an arithmetic expression are integers, the expression is called integer expression, and the operation is called integer arithmetic.
- Integer arithmetic always yields integer values.

- Operators applicable
  - All arithmetic operators
  - All logical operators
  - All relational operators
  - All increment and decrement operators
  - All bit-wise operators

Real Arithmetic

- Arithmetic operations involving only real or floating-point operands.
- Since floating-point values are rounded to the number of significant digits permissible, the final value is an approximation of the final result.

Examples

1.0 / 3.0 * 3.0 will have the value 0.99999 and not 1.0

a = 22.0/7.0*7*7 = (((22.0/7.0)*7)*7) = 153.86

b = 22*7/7*7 = (((22*7)/7)*7) = 154

Mixed-mode Arithmetic

- If either operand is of the real type, then only real arithmetic is performed, and the result is a real number.

25 / 10 → 2
25 / 10.0 → 2.5

- C language permits mixing of constants and variables of different types in an expression
- During evaluation it adheres to very strict rules of type conversion
  - If operands are of different types, the lower type is automatically converted to the higher type before the operation proceeds LOWER int < long < float < double HIGHER
  - char and short are automatically converted to int.
  - If one operand is unsigned, then other is converted to unsigned and the result is in unsigned
  - **float** is automatically converted to **double**
  - If one operand is double, then other is converted to double and the result is in double
  - If one operand is long, then the other operand is converted to long

Type casting

- C language allows to force a type conversion, which is different than the automatic type conversion

- The syntax for such a **type casting** is
  (type_name) expression;

Examples

int a = 4, b = 5; float x; double y;

x = (float) a / b; *// division is done in floating point mode,* x = 0.8
a = (int) x / b; *// Result is converted to integer by truncation,* a = 0
y = (char) b / a; *// It may report wrong type conversion*

Assume that variables a and b have data type `int` and variable c and d have data type float. Also, a = 9, b = 8, c = 16.0, and d = 6.0. For each question write the value assigned to the variable z. Data type of z is float.

z = a + c / 4 * d / 3 + b;

z = c + a / 4 * b / 3 + d ;

z = (int) c / a * b / 3 ;

z = a / b * b % 5 % 3 *c;

What will be the output in the following C Programs?

Program #1

```
#include <stdio.h>
int main ()
{
   int n;
   scanf("%d",&n);
   printf("%d\n",1/n);
   return 0;
}
```

Program #2

```
#include <stdio.h>
int main ()
{
   int n;
   scanf("%d",&n);
   printf("%f\n",1/n);
   return 0;
}
```

Program #3

```
#include <stdio.h>
int main ()
{
   int n;
   scanf("%d",&n);
   printf("%f\n",1.0/n);
   return 0;
}
```

Program #4

```
#include <stdio.h>
int main ()
{
   int n; float x;
   scanf("%d",&n);
   x = (float)1/n;
   printf("%f\n",x);
   return 0;
}
```

Important links:

http://cse.iitkgp.ac.in/~dsamanta/courses/pds/index.html