Program

Ramesh's basic salary is input through the keyboard. His dearness allowance is 40% of basic salary, and house rent allowance is 20% of basic salary. Write a program to calculate his gross salary?

Ramesh's basic salary is input through the keyboard. His dearness allowance is 40% of basic salary, and house rent allowance is 20% of basic salary. Write a program to calculate his gross salary?

Algorithm

Step 1: Input BS.

/* BS is a variable that represents Basic Salary */

Step 2: Compute DA

/* DA is a variable that represents Dearness Allowance */

DA = 40 * BS /100 ;

Step 3: Compute RA

/* RA is a variable that represents Rent Allowance */

RA = 20 * BS /100;

Step 4: Calculate GS

/* GS is a variable that represents Gross Salary */

GS = BS + DA + RA

Step 5: PRINT GS

Step 6: End

C Instructions

- Type Declaration Instruction
- Arithmetic Instruction
- Control Instruction

Type Declaration Instruction

The type declaration statement is written at the beginning of main( ) function.

```
int bas ;
float rs, grosssal ;
char name, code ;
```

There are several subtle variations of the type declaration instruction.

a) `int i = 10, j = 25 ;`
`float a = 1.5, b = 1.99 + 2.4 * 1.44 ;`

b) `int i = 10, j = 25 ;`
is same as
`int j = 25, i = 10 ;`

However,
`float a = 1.5, b = a + 3.1 ;`
is alright, but
`float b = a + 3.1, a = 1.5 ;`
is not.

c) The following statements would work

```
int a, b, c, d ;
a = b = c = 10 ;
```

However, the following statement would not work

```
int a = b = c = d = 10 ;
```

Arithmetic Instruction

- A C arithmetic instruction consists of a variable name on the left hand side of = and variable names & constants on the right hand side of =.
- The variables and constants appearing on the right hand side of = are connected by arithmetic operators like **+, -, *,** and **/**.
- Ex.:

```
int ad ;
float kot, deta, alpha, beta, gamma ;
ad = 3200 ;
kot = 0.0056 ;
deta = alpha * beta / gamma + 3.2 * 2 / 5 ;
```

Arithmetic Instruction

- **Integer mode arithmetic statement**

  Ex.: int i, king, issac, noteit ;

  i = i + 1 ;

  king = issac * 234 + noteit - 7689 ;

- **Real mode arithmetic statement**

  Ex.: float qbee, antink, si, prin, anoy, roi ;

  qbee = antink + 23.123 / 4.5 * 0.3442 ;

  si = prin * anoy * roi / 100.0 ;

- **Mixed mode arithmetic statement**

  Ex.: float si, prin, anoy, , avg,num ;

  int a, b, c, roi ;

  si = prin * anoy * roi / 100.0 ;

  avg = (a + b + c + num) / 4 ;

Points to Remember

a) C allows only one variable on left-hand side of **=**.

Ex. **z = k * l** is legal, whereas **k * l = z** is illegal.

b) Modulus operator (%)

10 % 2 yields 0.

c) Modulus operator (**%**) cannot be applied on a float.

d) On using % the sign of the remainder is always same as the sign of the numerator.

Ex. –5 % 2 yields –1, whereas, 5 % -2 yields 1.

Points to Remember

e) An arithmetic instruction is used for storing character constants in character variables .

Ex:

```
char a, b, d ;
a = 'F' ;
b = 'G' ;
d = '+' ;
```

Value	Char	Value	Char	Value	Char	Value	Char	Value	Char	Value	Char
0		22	▬	44	,	66	B	88	X	110	n
1	☺	23	↕	45	-	67	C	89	Y	111	o
2	☻	24	↑	46	.	68	D	90	Z	112	p
3	♥	25	↓	47	/	69	E	91	[	113	q
4	♦	26	→	48	0	70	F	92	\	114	r
5	♣	27	←	49	1	71	G	93	]	115	s
6	♠	28	⌐	50	2	72	H	94	^	116	t
7	•	29	↔	51	3	73	I	95		117	u
8	◘	30	▲	52	4	74	J	96	`	118	v
9	○	31	▼	53	5	75	K	97	a	119	w
10	◙	32		54	6	76	L	98	b	120	x
11	♂	33	!	55	7	77	M	99	c	121	y
12	♀	34	"	56	8	78	N	100	d	122	z
13	♪	35	#	57	9	79	O	101	e	123	{
14	♫	36	$	58	:	80	P	102	f	124	|
15	☼	37	%	59	;	81	Q	103	g	125	}
16	►	38	&	60	<	82	R	104	h	126	~
17	◄	39	'	61	=	83	S	105	i	127	ᴹʜ
18	↕	40	(	62	>	84	T	106	j	128	Ç
19	‼	41	)	63	?	85	U	107	k	129	ü
20	¶	42	*	64	@	86	V	108	l	130	é
21	§	43	+	65	A	87	W	109	m	131	â

Points to Remember

f) Arithmetic operations can be performed on **int**s, **float**s and **char**s.

```
char x, y ;
int z ;
x = 'a' ;
y = 'b' ;
z = x + y ;
```

g) No operator is assumed to be present. It must be written explicitly. In the following example, the multiplication operator after b must be explicitly written.

```
a = c.d.b(xy)                  // usual arithmetic statement
b = c * d * b * ( x * y )      //Correct  C statement
```

Points to Remember

h) Unlike other high level languages, there is no operator for performing exponentiation operation. Thus following statements are invalid.

```
a = 3 ** 2 ;
b = 3 ^ 2 ;
```

- Correct way:

```
#include <math.h>
main( )
{
  int a ;
  a = pow ( 3, 2 ) ;
  printf ( “%d”, a ) ;
}
```

Integer and Float Conversions

- An arithmetic operation between an integer and integer always yields an integer result.
- An operation between a real and real always yields a real result.
- An operation between an integer and real always yields a real result. In this operation the integer is first promoted to a real and then the operation is performed. Hence the result is real.

Integer and Float Conversions

Operation	Result	Operation	Result
5 / 2	2	2 / 5	0
5.0 / 2	2.5	2.0 / 5	0.4
5 / 2.0	2.5	2 / 5.0	0.4
5.0 / 2.0	2.5	2.0 / 5.0	0.4

Type Conversion in Assignments

```
int i ;
float b ;
i = 3.5 ;
b = 30 ;
```

```
#include<stdio.h>
int main()
{
   int i ;
   float b ;
   i = 3.5 ;
   b = 30 ;
   printf(“i = %d and b=%f", i, b);
}
```

Output:
i = 3 and b = 30.000000

program fragment

- float a, b, c ;
- int s ;
- s = a * b * c / 100 + 32 / 4 - 3 * 1.1 ;

```
int k;
float a;
```

Arithmetic Instruction	Result	Arithmetic Instruction	Result
k = 2 / 9	0	a = 2 / 9	0.0
k = 2.0 / 9	0	a = 2.0 / 9	0.2222
k = 2 / 9.0	0	a = 2 / 9.0	0.2222
k = 2.0 / 9.0	0	a = 2.0 / 9.0	0.2222
k = 9 / 2	4	a = 9 / 2	4.0
k = 9.0 / 2	4	a = 9.0 / 2	4.5
k = 9 / 2.0	4	a = 9 / 2.0	4.5
k = 9.0 / 2.0	4	a = 9.0 / 2.0	4.5

```
int k;
float a;
```

Arithmetic Instruction	Result	Arithmetic Instruction	Result
k = 2 / 9	0	a = 2 / 9	0.0
k = 2.0 / 9	0	a = 2.0 / 9	0.2222
k = 2 / 9.0	0	a = 2 / 9.0	0.2222
k = 2.0 / 9.0	0	a = 2.0 / 9.0	0.2222
k = 9 / 2	4	a = 9 / 2	4.0
k = 9.0 / 2	4	a = 9.0 / 2	4.5
k = 9 / 2.0	4	a = 9 / 2.0	4.5
k = 9.0 / 2.0	4	a = 9.0 / 2.0	4.5

Point out the errors, if any, in the following C statements:

(a) int = 314.562 * 150 ;
(b) name = ‘Ajay’ ;
(c) varchar = ‘3’ ;
(d) 3.14 * r * r * h = vol_of_cyl ;
(e) k = (a * b) (c + (2.5a + b) (d + e) ;
(f) m_inst = rate of interest * amount in rs ;
(g) si = principal * rateofinterest * numberofyears / 100 ;
(h) area = 3.14 * r ** 2 ;
(i) volume = 3.14 * r ^ 2 * h ;
(j) k = ((a * b) + c) (2.5 * a + b) ;
(k) a = b = 3 = 4 ;
(l) count = count + 1 ;
(m) date = '2 Mar 04' ;

Point out the errors, if any, in the following C statements:

(a) int = 314.562 * 150 ;
(b) name = ‘Ajay’ ;
(c) varchar = ‘3’ ;
(d) 3.14 * r * r * h = vol_of_cyl ;
(e) k = (a * b) (c + (2.5a + b) (d + e) ;
(f) m_inst = rate of interest * amount in rs ;
(g) si = principal * rateofinterest * numberofyears / 100 ;
(h) area = 3.14 * r ** 2 ;
(i) volume = 3.14 * r ^ 2 * h ;
(j) k = ((a * b) + c) (2.5 * a + b) ;
(k) a = b = 3 = 4 ;
(l) count = count + 1 ;
(m) date = '2 Mar 04' ;

a) int total = 314.562 * 150 ;
(b) name = ‘A’ ;
(c) varchar = ‘3’ ;
(d) vol_of_cyl = 3.14 * r * r * h;
(e) k = (a * b)* (c + (2.5*a + b) *(d + e) ;
(f) m_inst = rate_of_interest * amount_in_rs ;
(g) si = principal * rateofinterest * numberofyears / 100 ;
(h) area = 3.14 * pow(r , 2) ;
(i) volume = 3.14 *pow(r, 2) * h ;
(j) k = ((a * b) + c)* (2.5 * a + b) ;
(k) a = b = 3 ;
(l) count = count + 1 ;
(m) date = '2' ;