Programming in C

Data Type:-

Primitive D.T.
↳ Integer, char, float
'..' → Single literal constant
".." → Multi literal constant

Derived
↳ Array, String, pointer
↳ " "

User Define
↳ structure, Union, enum, typedef (alias) (another name)
(collection of dissimilar element)

int = 4 Byte size = 32 bits
↳ Short int → 2B
↳ int → 4B
↳ long int → 8B

$2^{10} \rightarrow 1024$
$2^{11} \rightarrow 2048$
$2^{12} \rightarrow 4096$
$2^{13} = 8192$
$2^{14} = 16384$
$2^{15} = 32768$
$2^{16} = 65536$

Range of

Unsigned no: 0 to $2^n - 1$

2's complement

0000 → 1111+1 = 0000
0111 → 1000+1 = 1001
+7 −1

Sign Magnitude
4 bits
↑ Sign

Sign magnitude: $-2^{n-1}-1$ to $2^{n-1}-1$
1's complement: $-2^{n-1}-1$ to $2^{n-1}-1$
2's complement: -2^{n-1} to $2^{n-1}-1$

If range of signed integer range -2^{n-1} to $2^{n-1}-1$

what is the range of 8 bit signed integer:
B. −128 to 127

what is the decimal value of signed 6 bit no.
100000
2's ↓
100000 → −32
1×2^5 - - - 0×2^0

"%d" → Signed Integer. d→decimal 4 Byte

"%u" → Unsigned u. 4 Byte unsigned short int a = 23;

"%hd" → Short integer 2 Byte

"%hu" → Short unsigned integer 2 Byte.

→ a = −4

"%d" → −4 0100 → 1100 (stored in memory) (2's com.)

"%u" → 12 (signed ↓ unsigned) $1\times2^3+1\times2^2 = 12$

short "%u" → $2^{16}-4$

Formulae.

for 4 bit = $2^4 = 16$.

−4 → %d → −4

%u = 16 − 4 = 12.

→ ch1 = 'a', %c → a %d → 97

ch2 = 'z' %c → z %d → 122

a − z → 97 − 122

A − Z → 65 − 90

'0' − '9' → 48 − 57

Char → 1 Byte

2's complement

It's C: 100011 −29.

011100

+1

011101 → 29

100011 → $-2^5+2+1 = -29$

If after 2's complement MSB changes ∴ no overflow

If MSB not changes ∴ overflow.

★

```
Char ch1 = 65;
```

%c → A

%d → 65

short int a = 32771; → 16 bit range -2^{n-1} to $-2^{n-1}-1$

printf ("%hd", a)

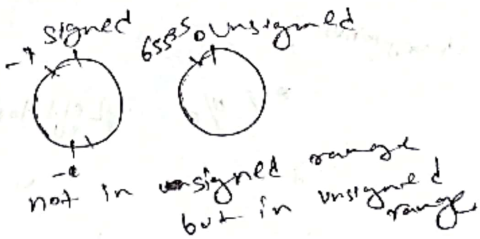
-32768 to 32767

∴ Out of range

$2^{16} \rightarrow 65536$

$-\ 32771$

-32765. Ans.



not in unsigned range but in unsigned range

short int a = −33333; → exceed in '−'.

65536

$-\ |-33331|$

32203 → Ans. '+' no.

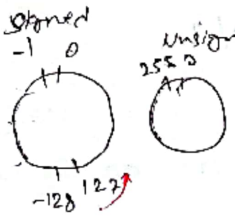
short int a = 65540;

↓ exceeding more than 65535. ∴ 65540 % 65536

↓ '4' → Ans.

char ch = −134;

"%c", ch; →



8 bit → char.

∴ −134 is not in signed range but in unsigned range

∴ $256 - |-134| = 122$ → 'z' Ans.

Variable name can start with a...z, A...Z, _

∴ char Int = 'a'; → is correct as case sensitive

Int is keyword

not Int

Lvalue = R-value

address kind of variable not any constant

20 = a

Lvalue required ∴ Error.

Dividend = Quotient * Divisor + Remainder.

7/3 → Dividend

3 → Divisor.

$-7/3$ Q = −2, R = −1

$7/3$ Q = 2, R = 1

$-7/-3$ Q = 2, R = −1

Divident → '4' ∴ → Remainder '4'

'-' ∴ → " '-'

Precedence & Associativity Rule

Associativity

( ) Highest

Unary minus −

* / % Left to Right

\+ − "

= Right to left Lowest

Relational operator

Left to Right

> <
>= <= } more Precedence

== , != Less precedence

Unary -

−5 + 10

(−5) + 10 −(5+10)

x = 7 * 2.0 / 2 + 10 / 3

= 14.0 / 2 + 10/3

= 7.0 + 10/3

= 7.0 + 3

= 10.0

x = −2 + 11 − 7 * 9 % 6/12;

= (−2) + 11 − 63 % 6/12

= (−2) + 11 − 3 / 12

= (−2) + 11 − 0

= 9

x = 2 * 3 / 4 + 4 / 4 + 8 * − 2 + 5/8

= 6/4 + 4/4 + 8 − 2 + 5/8

= 1 + 1 + 8 − 2 + 0

= 2 + 8 − 2

= 10 − 2

= 8

// ⇒ Single line comment

/* */ → multiple line Comment

x = 4 + 2 % −8

4 + 2 = 6

x = (−2) + 11 − 103 * 2 % 3 / 2

= (−2) + 11 − 206 % 3 / 2

= (−2) + 11 − 2 / 2

= (−2) + 11 − 1

= 9 − 1

= 8

x = 2 * 3 / 4 + 4 / 4 + 5/8 − 8 * 4

= 6/4 + 4/4 + 5/8 − 8 * 4

= 1 + 1 + 1 + 0 − 32

= −30

x = (−3) % 2 * 4 % 3 / 8 + 3

= −1 * 4 % 3 / 8 + 3

= −4 % 3 / 8 + 3

Scope of Variable

Dynamic Scope: [From where the function has be called is the after-function/block.

Lexical Scope:

If declaration of a variable not found within a block the declaration of variable searched in outer block.

```
printf ("%d", 10 == 4<5)
        10 == 1
          ↓
          0
```

4<5<0
→
1<0
↓
0

Logical Operator → AND OR NOT
&& || !

Negation → Unary
minus → Unary

* i = 2.
~(i) = ~(2)
= -(2+1)
= -3

i = -2019
~(10) = -(-2019+1)
= +2018

5 + 5 ! = 10 || 6+4

10 ! = 10 || 10
↓
0 || 10
↓
1

zero in C is 0
Non zero in C is 1.

Short Circuit code: If Logical operator && is applied if first operand is 0 then second operand is Not evaluated. Result = 0

If Logical operation OR, || is applied if first operand is 1 then " " " "
Result = 1

```
int a, x = 5, y = 6
a = ++x || ++y
```
→
x = 6
y = 6

Because Short Circuit code.

x=0, y=6, a.
a = x++ && ++y
(x++ → 0)
x = 1
y = 6

Bitwise Operator:

&
|
negation ~
XOR ^

>> Shift Right
<< Shift Left

int x = 5 ;
int y = 17 ;
int z = x & y

let 1 Byte = 8 bits.

```
  0000 0101
& 0001 0001
-----------
  0000 0001  →  1
```

x | y

```
  0000 0101
  0001 0001
-----------
  0001 0101   = 21
  16 8 4 2 1
```

x ^ y

```
  0000 0101
  0001 0001
-----------
  0001 0100  →  20
```

a = 8 . → 0000 1000

a << 3 → 0100 0000 = 64

Each bit shift Left operation is equal to multiply by 2 if overflow doesn't occur.

and → 1 Byte.

int x = 5 ; → 0000 0101

a = ~x ; → 1111 1010 → Negative value −6

2's

0000 0110 → 6

int a = 8 ; → 0000 1000 → 8 .

a >> 2 → 0000 0010 → 2

1 time Right Shift is divide by 2.

#

```
int b ;
b = printf ("Hello Dosto");
printf ("%d", b);   →  Hello Dosto11

printf ("%d", printf ("ABCD"))  →  ABCD4
```

Control Flow

1) Selection or Conditional or branching.
2) Iterative
3. Jump

```
# int a = 10;
  if (a >= 10)          → No bracket
     a = a+10;          → Part of If
     a = a+5;           → Not part of if block   (always execute)
  printf("%d", a);

25
```

```
if (a < 10)
   a = a+10;   ✗
   a = a+5;    ← execute
printf ...

15
```

```
# int a = 5;        → a = 5 [Not zero]
  if (a = 5) {      As value of a is not zero ∴ It will not be executed
  ---               ∴ Condition true
  ---
  }
  printf("%d", a);
```

```
# int a = 5;
  if (a == 10); {   → terminate if statement
     a = a+10       ] Normal execute
     a = a-5
  }
  printf("%d", a)
```

```
# char a = -15;
  int u;
  u = a>>3
```

8 bit

11110001 → 2's

00111010 (6 5 4 3 2 1 0)

2 | 15
2 | 7 - 1
2 | 3 - 1
2 | 1 - 1
  0

00001111

00001111 → 2's com. → Ans 2

00000011 → 2's complement

11110001

11111000

11111100

11111110

00000010 = -2 (↑ As negative)

For negative no. with shift in Right 1 will be in MSB

In switch case: If break is Not present then after that case every case evaluates to true.

Each list value is called Case.

In switch float/double is error.

printf(" ") returns no. of character.

```
Switch (printf("ABCD"))
    case 4: printf("%d", 4); break.
    case 5: - - -
}
```

ABCD4 Ans.

In for

```
int i=10;
for( ; i<=15; i++) {
    - - -
}
```

Run after the block.

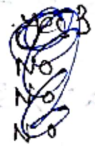
No problem.
if no expression also ';' must present

```
int i;
for( ; i<=15; i++) {- - -}.
```

depend upon garbage value, loop will execute


```
int i=1;
for(; i<=3; printf("yes")){
    printf("No");
    i=i+1;
}
```

3



NoyesNoyesNoyes

☆ expr3 for(expr1; expr2; expr3) always execute after executing all statement within the block.

```
int i=-1;
for(; ++i; i++){
    - - -
}
```

0

As 0 so loop will not run.

false

char ch;

for (ch=1; ch; ch++){ will run 255 times.

}

Gate 2015

```
int i, j, k = 0
j = 2 * 3/4 + 2.0/5 + 8/5;      → 2.0.
k = k - (--j);   → k = 0-(1) → k = -1
for (i=0; i<5; i++){
    switch (i+k){
        case 1:
        case2: printf ("\n%d", i+k);
        Cas 3: printf ("\n%d", i+k);
        default: printf ("\n%d", i+k);
    }
}
```

No. of times printf will execute

i=0
switch = -1 → 1
i=1 ... 0 → 1
i=2 ... 1 → 3
i=3 ... 2 → 3
i=4 ... 3 → 2
= 10

$2*3/4 + 2.0/5 + 8/5$

$6/4 + 0.4 + 1$

$1 + 0.4 + 1 \Rightarrow 2.4$ But j = int.

∴ j = 2.

	int	float
2/5 =	0	0.00 - -
2.0/5 =	0	0.4
8.0/5 =	1	1.6

In case of :

switch (i<<5)

if (i<<5)

} No effect of i just for checking it is evaluated according i<<5 the again i will be in initial value

But if increment then increment performed.

Int a = 2023;

printf("%.d %.d %.d", a++, a++, a++);

3 2 1

→ 10 2021 0

Evaluated from left to right

Sum of $1^2+2^2+3^2 \cdots +10^2 = \frac{n(n+1)(2n+1)}{6}$

$1+2+3+\cdots+n = \frac{n(n+1)}{2}$

Operator	Description	Associativity
() [] . -> ++ --	Parentheses or function call Brackets or array subscript Dot or Member selection operator Arrow operator Postfix increment/decrement	left to right
++ -- + - ! ~ (type) * & sizeof	Prefix increment/decrement Unary plus and minus not operator and bitwise complement type cast Indirection or dereference operator Address of operator Determine size in bytes	right to left
* / %	Multiplication, division and modulus	left to right
+ -	Addition and subtraction	left to right
<< >>	Bitwise left shift and right shift	left to right
< <= > >=	relational less than/less than equal to relational greater than/greater than or equal to	left to right
== !=	Relational equal to or not equal to	left to right
&&	Bitwise AND	left to right
^	Bitwise exclusive OR	left to right
|	Bitwise inclusive OR	left to right
&&	Logical AND	left to right
||	Logical OR	left to right
? :	Ternary operator	right to left
= += -= *= /= %= &= ^= |= = >>=	Assignment operator Addition/subtraction assignment Multiplication/division assignment Modulus and bitwise assignment Bitwise exclusive/inclusive OR assignment	right to left
,	comma operator	left to right

< Back

All C Programming

Consider the following program:

```
#include<stdio.h>
void main()
 {
  int x=-2024;
 printf("%d", ~(x=x+5)) ;
 printf("%d", ~(x+1));
 }
```

The sum of the output values printed is _____.

Previous

Solution

$x = x + 5 \rightarrow x = -2024 + 5 = -2019$

$\sim(x) \rightarrow \sim(-2019) = -(-2019 + 1) = 2018$ $\sim(x + 1) \rightarrow \sim(-2019 + 1) = -(-2018 + 1) = 2017$

Sum of the values printed = 2018 + 2017 = 4035.

Previous