Extra

1. Time count.

2. Random Number.

3. Constant

4. Creating macro.

5. Enum basic problem

6. Bitwise not

7. Shifting right or left

8. Bitwise and

9. Bitwise or

10. Bitwise x-or (exclusive or )

11. Determine even and odd using bitwise operation

12. Uppercase and lowercase using bitwise operator.

13. কোনো একটি সংখ্যা ২ এর পাওয়ার কিনা তা নির্নয়।

14. কোনো ধনাত্নক পূনসংখ্যার বাইনারি রুপে মোট কয়টি বিট 1

15. দুটি সংখ্যা সমান কিনা তা নির্নয়।

1. Time count.

#include <stdio.h>

#include <time.h>

void fun(int x, int n)

{

x = n \* 2;

}

int main()

{

int i, j, x, n;

clock\_t starttime, endtime;

double timeelapsed;

starttime = clock();

n = 12345678;

for (i = 0; i < 1000000000; i++) {

for (j = 0; j < 10; j++) {

x = n \* 2;

}

}

endtime = clock();

timeelapsed = (double)(endtime - starttime) / CLOCKS\_PER\_SEC;

printf("Time : %lf seconds\n", timeelapsed);

starttime = clock();

n = 12345678;

for (i = 0; i < 1000000000; i++) {

for (j = 0; j < 10; j++) {

fun(x, n);

}

}

endtime = clock();

timeelapsed = (double)(endtime - starttime) / CLOCKS\_PER\_SEC;

printf("Time : %lf seconds\n", timeelapsed);

return 0;

}

Output:

Time : 12.627000 seconds

Time : 21.584000 seconds

2. Random Number.

Output:

2233

20874

28795

11503

23403

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int main()

{

time\_t t;

srand((unsigned)time(&t));

for (int i = 0; i < 5; i++) {

printf("%d\n", rand());

}

return 0;

}

* কোডটি রান করলে প্রতেকবার ভিন্ন ভিন্ন আউটপুট আসবে। Random number এর জন্যে stdlib.h হেডার ফাইলের ভেতরে rand() ফাংশনটি ব্যবহার করতে হবে।
* srand function-এ এমন সংখ্যা পাঠাতে হবে যেটি একেকবার একেকরকম হবে। আমরা যদি কম্পিউটার এর বর্তমান সময়(time(&t)) পাঠাই তাহলে কিন্তু একেকবার একেক সংখ্যা পাঠানোর কাজটি হয়ে যাবে।

3(1). Constant

Output: a = 50, b = 60

#include <stdio.h>

#define p 50

#define q 60;

int main()

{

int a = p;

int b = q

printf("a = %d, b = %d\n", a, b);

}

q - কে ডিফাইন করেছি 60; (সেমিকোলন সহ) তাই কম্পাইল হওয়ার সময় q এর বদলে 60; বসে যাবে। ফলে কোনো এরর হবে না।

3(2). Constant

Output:

Value of n = 1

Value of PI = 3.141593

#include <stdio.h>

#include <math.h>

#define n 1

#define PI (2 \* acos(0))

int main()

{

printf("Value of n = %d\n", n);

printf("Value of PI = %lf\n", PI);

}

PI-এর মান বের করার জন্য সবচেয়ে ভালো পদ্বতি এটা।

4(1). Creating macro.

#include <stdio.h>

#define MAX(a, b) (a>b? a:b)

int main()

{

int a = 10, b = 20;

double x = 5.543, y = 3.1245;

printf("Maximum of %d and %d is = %d\n", a, b, MAX(a, b));

printf("Maximum of %lf and %lf is = %lf\n", x, y, MAX(x, y));

}

Output:

Maximum of 10 and 20 is = 20

Maximum of 5.543000 and 3.124500 is = 5.543000

* Here we use ternary operator.
* Note: This macro is worked both int and double data type.
* It is good programing practice to use capital letter for constant(like MAX).

4(2). Creating macro.

Output:

Before swapping : a = 10, b = 20

After swapping : a = 20, b = 10

#include <stdio.h>

#define SWAP(a, b) { \

a ^= b; \

b ^= a; \

a ^= b; \

}

int main()

{

int a = 10, b = 20;

printf("Before swapping : ");

printf("a = %d, b = %d\n", a, b);

SWAP(a, b);

printf("After swapping : ");

printf("a = %d, b = %d\n", a, b);

}

If we want to make multiple line macro then we must follow this above procedure. And must use backslash sign(\) at the end of every statement.

5(1). Enum basic problem

#include <stdio.h>

enum color {red, green, blue};

int main()

{

enum color selectedcolor;

int num;

printf("Enter num 1 for red, 2 for green and 3 for blue : ");

scanf("%d", &num);

if (num == 1) selectedcolor = red;

else if (num == 2) selectedcolor = green;

else if (num == 3) selectedcolor = blue;

printf("Selected color : %d", selectedcolor);

}

Output:

Enter num 1 for red, 2 for green and 3 for blue : 3

Selected color : 2

This output looks like array index which is start from 0. If we enter 3 then the index of 3 is (3-1) = 2 and this is the output of selected color.

return 0 কথাটি চাইলে বাদ দেয়া যাবে। যদি এই লাইনটি না লিখা হয়

ডিফল্ট হিসাবে main() ফাংশন এক্সিকিউশনের শেষে 0 রির্টান করবে।

5(2). Enum basic problem

Output:

Value : 0

Value : 1

Value : 2

Value : 3

#include <stdio.h>

enum color {No\_color, red, green, blue};

int main()

{

printf("Value : %d\n", No\_color);

printf("Value : %d\n", red);

printf("Value : %d\n", green);

printf("Value : %d\n", blue);

}

5(3). Enum basic problem

#include <stdio.h>

enum color { No\_color = 0, red = 40, green = 44, blue = 80 };

int main()

{

printf("Value : %d\n", No\_color);

printf("Value : %d\n", red);

printf("Value : %d\n", green);

printf("Value : %d\n", blue);

}

Output:

Value : 0

Value : 40

Value : 44

Value : 80

5(4). Enum basic problem

#include <stdio.h>

enum day { saturday, sunday, monday, tuesday, wednesday, thrusday, friday };

int main()

{

enum day today;

today = saturday;

switch (today)

{

case saturday:

printf("Today is saturday\n");

break;

case sunday:

printf("Today is sunday\n");

break;

case monday:

printf("Today is monday\n");

break;

case tuesday:

printf("Today is tuesday\n");

break;

case wednesday:

printf("Today is wednesday\n");

break;

case thrusday:

printf("Today is thrusday\n");

break;

case friday:

printf("Today is friday\n");

break;

default:

printf("Enter a valid day\n");

break;

}

}

Output:

Today is Saturday

6. Bitwise not

#include <stdio.h>

int main()

{

char a, b;

a = 0;

b = ~a;

printf("a = %d, b = %d\n", a, b);

a = 1;

b = ~a;

printf("a = %d, b = %d\n", a, b);

}

Output:

a = 0, b = -1

a = 1, b = -2

NOTE:

When a = 0 then b = -1 ,cause

The binary value of 0 is = 00000000

If we reverse it then the value is = 11111111 (This is the value of -1)

So, that’s why when a = 0 then b = -1.

Similarly, when a = 1 then b = -2, cause

The binary value of 1 is = 00000001

If we reverse if then the value is = 11111110 (This is the value of -2)

So, that’s why when a = 1 then b = -2

The main work of bitwise not (~) operator is reverse the number.

(Like that: 0 to 1 and 1 to 0)

Remember: This is not the same as logical not(!) operator.

7. Shifting right or left

#include <stdio.h>

int main()

{

int n, x, m;

while (1)

{

/\*The program will be closed when you enter 0\*/

printf("Please enter your number : ");

scanf("%d", &n);

if (n == 0)

break;

printf("How many bits you want to shift left? ");

scanf("%d", &x);

m = n << x;

printf("Result : %d\n", m);

}

}

Output:

Please enter your number : 9

How many bits you want to shift left ? 1

Result : 18

Please enter your number : 2

How many bits you want to shift left ? 1

Result : 4

Please enter your number : 2

How many bits you want to shift left ? 2

Result : 8

Please enter your number : 0

NOTE:

<< : shift left

>> : shift right

8. Bitwise and

Output: 5 & 6 = 4

#include <stdio.h>

NOTE:

5 = 00000101

6 = 00000110

--------------

5&6 = 00000100

int main()

{

int n1 = 5, n2 = 6, n3;

n3 = n1 & n2;

printf("%d & %d = %d\n", n1, n2, n3);

}

5 এর একদম ডানদিকের বিটের সাথে 6 এর একদম ডানদিকের বিটের মধ্যে অপারেশন চালাবে।

9. Bitwise or

Output: 5 | 6 = 7

#include <stdio.h>

int main()

{

int n1 = 5, n2 = 6, n3;

n3 = n1 | n2;

printf("%d | %d = %d\n", n1, n2, n3);

}

আগেরটার মতো চিন্তা করো।

10. Bitwise xor(exclusive or)

Output: 5 ^ 6 = 3

#include <stdio.h>

**1 0 output 1**

**1 1 output 0**

**0 0 output 0**

int main()

{

int n1 = 5, n2 = 6, n3;

n3 = n1 ^ n2;

printf("%d ^ %d = %d\n", n1, n2, n3);

}

11. Determine even and odd using bitwise operation

Output:

4

This is even number

#include <stdio.h>

int main()

{

int n;

scanf("%d", &n);

if (n & 1) printf("This is odd number\n");

else printf("This is even number\n");

}

NOTE:

4 & 1 = 00000100 & 00000001 = 00000000 = 0 (even)

5 & 1 = 00000101 & 00000001 = 00000001 = 1 (odd)

12. Uppercase and lowercase using bitwise operator.

#include <stdio.h>

char to\_upper(char ch)

{

return ch & 95;

}

char to\_lower(char ch)

{

return ch | 32;

}

int main()

{

char \*str = "aBcdefghijklmNOPQrstuvwxYZ";

for (int i = 0; i < 26; i++)

{

printf("Uppercase : %c\n", to\_upper(str[i]));

printf("Lowercase : %c\n", to\_lower(str[i]));

}

}

Output:

Uppercase: A

Lowercase : a

Uppercase : B

Lowercase : b

Uppercase : C

Lowercase : c

Uppercase : D

Lowercase : d

-------------

-------------

NOTE:

return ch & 95; Why we use this?

Look, first character of the str is a

a = 01100001

95 = 01011111

-------------------

a&95 = 01000001 (This is the value of A or 65)

13. কোনো একটি সংখ্যা ২ এর পাওয়ার কিনা তা নির্নয়।

#include <stdio.h>

int main()

{

int n;

scanf("%d", &n);

if (n > 0 && (n & (n - 1)) == 0)

printf("Yes, %d is a power of 2\n", n);

else

printf("No, %d is not a power of 2\n", n);

}

Output:

16

Yes, 16 is a power of 2

NOTE:

Look 1, 2, 4, 8, 16 this is the number which is the power of 2.

The binary value of this number are:

1 = 1, 2 = 10, 4 = 100

মাত্র একটি বিট হলো ১

এখন যেই বিট ১ সেটাকে ০ করে দিয়ে তার ডান পাশের সবগুলো বিটকে যদি ১

করে দেই তাহলে কি আসে?

যেমনঃ 4 = 100 --> 4 = 011 (এটা হলো ৩ এর বাইনারি মান)

এই ৪ এবং ৩ এর মধ্যে bitwise and operation চালালে ফলাফল হবে ০

তাহলে n(4) এর মান যদি ০ না হয় এবং n&(n-1) এর মান যদি ০ হয় তাহলে

n(4) হবে ২ এর পাওয়ার।

14. কোনো ধনাত্নক পূনসংখ্যার বাইনারি রুপে মোট কয়টি বিট 1

Output:

4

Number of 1 : 1

#include <stdio.h>

int main()

{

int n, i;

scanf("%d", &n);

int count = 0;

while (n)

{

if (n & 1) {

count++;

}

n = n >> 1;

}

printf("Number of 1 : %d\n", count);

}

NOTE: The binary value of 4 is 100

So, the number of 1 is : 1

আমরা যদি ডানদিক থেকে ২য় বিটটি কি ০ নাকি ১ সেটি নির্নয় করতে যাই

তাহলে ১ কে এক ঘর left shift করে তার সঙ্গে সংখ্যাটির& করলেই হবে।

#include <stdio.h>

int main()

Output:

7

Number of 1 : 3

{

int n = 32;

int num, i;

int count = 0;

scanf("%d", &num);

for (i = 0; i < n; i++)

{

if (num & (1 << i))

count++

}

printf("Number of 1 : %d\n", count);

}

int n = sizeof(int) \* CHAR\_BIT; #include <limits.h>

15. দুটি সংখ্যা সমান কিনা তা নির্নয়।

Output:

4 4

Numbers are equal

#include <stdio.h>

int main()

{

int n, m;

scanf("%d %d", &n, &m);

if (n ^ m)

printf("Numbers are not equal\n");

else

printf("Numbers are equal\n");

}

NOTE:

x-or এর ক্ষেত্রে নিয়ম হচ্ছে যদি সংখ্যা দুটি সমান হয় তাহলে আউটপুট ০

অ্যারেতে একটি সংখ্যা কেবল একবারই আছে।

বাকি সংখ্যা গুলো দুই বার করে আছে। যে সংখ্যাটি একবার আছে সেটি নির্নয় করতে হবে।

#include <stdio.h>

int main()

{

int array[] = { 1, 2, 3, 4, 1, 2, 3 };

int i, n = 7;

int result = array[0];

for (i = 1; i < n; i++)

{

result = result ^ array[i];

}

printf("Result is : %d\n", result);

}

Output: Result is : 4

আমরা যদি এক্সর করি তাহলে কিন্তু ফলাফল হবে যে সংখ্যাটি একবার আছে সেটি।

কারন যেগুলো দুই বার করে আছে সেগুলোতে এক্সর চালালে ফলাফল হবে ০