Introduction to C Programming
Module - 25

বাইনারি সংখ্যা পদ্ধতি কাকে বলে? (Binary Number System)

যে সংখ্যা পদ্ধতিতে সংখ্যা গণনা করার জন্য ২টি অঙ্ক বা প্রতীক ব্যবহৃত হয় তাকে বাইনারি সংখ্যা পদ্ধতি বলে। যেমন-(১০১০)২। বাইনারি সংখ্যা পদ্ধতিতে যেহেতু ০ এবং ১ এই দুইটি প্রতিক বা চিহ্ন ব্যবহার করা হয় তাই এর বেজ বা ভিত্তি হচ্ছে ২। ইংল্যান্ডের গণিতবিদ জর্জ বুল বাইনারি সংখ্যা পদ্ধতি উদ্ভাবন করেন। বাইনারি সংখ্যা পদ্ধতি সবচেয়ে সরলতম সংখ্যা পদ্ধতি। বাইনারি সংখ্যা পদ্ধতির ০ এবং ১ এই দুটি মৌলিক চিহ্নকে বিট বলে এবং আট বিটের গ্রুপ নিয়ে গঠিত হয় একটি বাইট।লজিক লেভেল ০ এবং ১ বাইনারি সংখ্যা পদ্ধতির সাথে সামঞ্জন্যপূর্ণ। তাই কম্পিউটার বা সকল ইলেক্ট্রনিক্স ডিভাইসে বাইনারি সংখ্যা পদ্ধতি ব্যবহৃত হয়।

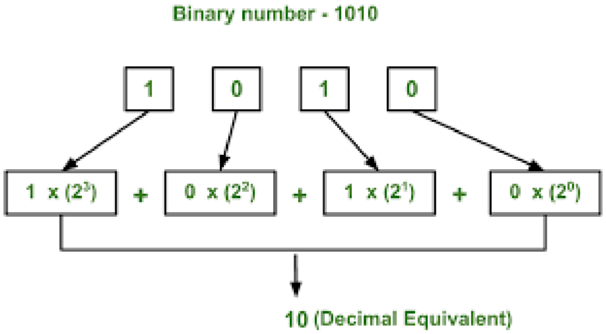
বাইনারি সংখ্যা পদ্ধতির বৈশিষ্ট্য (Characteristics of binary number system)

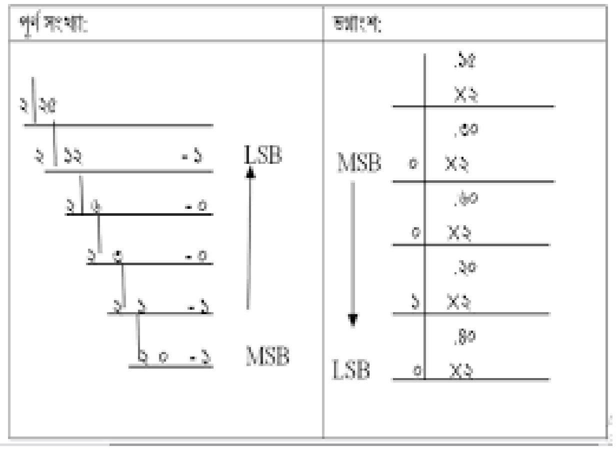
- বাইনারি নাম্বার সিস্টেমের বেস বা ভিত্তি হচ্ছে ২।
- এ পদ্ধতিতে ০ এবং ১ মোট ২টি মৌলিক অঙ্ক আছে।
- বাইনারি সংখ্যার মাধ্যমে কম্পিউটারের সমস্ত যোগ বিয়োগ ও অন্যান্য কার্যাদি সম্পন্ন করা হয়।

ডেসিমাল সংখ্যা পদ্ধতি-Decimal Number System

ডেসি অর্থ ১০ (দশ)। যে সংখ্যা পদ্ধতিতে সংখ্যা গণনা করার জন্য ১০ (দশ) টি অঙ্ক বা প্রতীক ব্যবহার করা হয় তাকে ডেসিমাল সংখ্যা পদ্ধতি বলে। মানুষ সাধারণত গণনার কাজে ডেসিমাল সংখ্যা পদ্ধতি ব্যবহার করে থাকে। প্রচলিত রয়েছে মানুষের ডেসিমাল সংখ্যা পদ্ধতি ব্যবহার করার কারণ হলো মানুষের দুই হাতের মোট আঙ্গুলের সংখ্যা ১০ (দশ)। ডেসিমাল সংখ্যা পদ্ধতিতে ব্যবহৃত প্রতীক বা অঙ্ক (ডিজিট) গুলো হলো 0,1,2,3,4,5,6,7,8 এবং 9। যেমন— $(497)_{10}$, $(46.89)_{10}$, $(1011)_{10}$ ইত্যাদি।

Binary to Decimal -





বাইনারি টু ডেসিমেল রুপান্তরের নিয়মাবালিঃ

```
1234.5678
| | | | └┴┴┴── ভগ্নাংশ
| | | | └────────── রেডিক্স পয়েন্ট (দশমিক)
└┴┴┴─────────── পূর্নসংখ্যা
```

পূর্ন সংখ্যার ক্ষেত্রে নিচের ধাপ গুলো অনুসরণ করতে হয়ঃ

1. পূর্ন সংখ্যাটিকে যে সংখ্যা পদ্ধতিতে নেয়া হবে তার ভিত্তি 2 দিয়ে ভাগ করতে হবে এবং ভাগশেষ আলাদা করে লিখে রাখতে হবে।
2. উপরের ধাপে প্রাপ্ত ভাগফলকে আবার ভিত্তি দিয়ে ভাগ করতে হবে এবং ভাগশেষ সংরক্ষন করতে হবে।
3. দুই নং ধাপটি চলতে থাকবে যতক্ষন পর্যন্ত ভাগশেষ শূন্য না আসে।
4. সংরক্ষতি ভাগশেষ গুলো সর্বশেষে পাওয়া ভাগশেষ থেকে প্রথম ভাগশেষ পর্যন্ত পর পর বাম থেকে ডানে দিকে লিখলে রুপান্তরিত সংখ্যাটি পাওয়া যাবে।

ভগ্নাংশের ক্ষেত্রে নিচের ধাপ গুলো অনুসরণ করতে হয়ঃ

1. ভগ্নাংশটিকে যে সংখ্যা পদ্ধতিতে নেয়া হবে তার ভিত্তি 2 দিয়ে গুন করতে হবে এবং পূর্ন অংশটিকে আলাদা করে সংরক্ষন করতে হবে।
2. উপরের ধাপে প্রাপ্ত ভগ্নাংশটিকে আবার ভিত্তি দিয়ে গুন করতে হবে এবং পূর্ন অংশটি সংরক্ষন করতে হবে।
3. দুই নং ধাপটি চলতে থাকবে যতক্ষন পর্যন্ত গুনফল পূন্য সংখ্যা আসে, অর্থাৎ ভগ্নাশ শূন্য না আসে।
4. সংরক্ষতি পূর্ন সংখ্যা গুলো সর্বপ্রথমে পাওয়া পূর্ন সংখ্যা থেকে সর্বশেষে পাওয়া পূর্ণ সংখ্যা পর্যন্ত পর পর বাম থেকে ডানে দিকে লিখলে রুপান্তরিত সংখ্যাটি পাওয়া যাবে।

বাইনারি টু ডেসিমেল(Binary to Decimal) সংখ্যা পদ্ধতিতে রুপান্তর করার একটি উদাহরন নিচে দেয়া হল। দ্রষ্টব্যঃ বাইনারি সংখ্যা পদ্ধতির ভিত্তি 2।

দশমিক সংখ্যা (123.456)10 এর বাইনারি মান কত?

পূর্ণ সংখ্যা
প্রশ্নের সংখ্যাটির পূর্ন সংখ্যা হচ্ছে 123। এবার 123 কে বাইনারির ভিত্তি 2 দিয়ে ভাগ করতে থাকব যতক্ষন পর্যন্ত ভাগফল শূণ্য না হয়। এবং প্রতিবার ভাগশেষ গুলো লিখে রাখব।

2|123 ভাগশেষ
—┼——————————
2|61   1

123 কে 2 দিয়ে ভাগ করলে ভাগফল 61 এবং ভাগশেষ 1।

2|123 ভাগশেষ
—┼——————————
2|61   1
—┼——————————
2|30   1

61 কে 2 দিয়ে ভাগ করলে ভাগফল 30 এবং ভাগশেষ 1।
এভাবে ভাগফল শূন্য না হওয়া পর্যন্ত করতে হবে।

2|123 ভাগশেষ
—┼——————————
2|61   1   ↑
—┼—————————— |
2|30   1 |

```
─┼──────────────│
2│15   0       │
─┼──────────────│
2│7    1       │
─┼──────────────│
2│3    1       │
─┼──────────────│
2│1    1       │
─┼──────────────│
│0     1       │
```

শেষ ধাপে 1 কে 2 দিয়ে ভাগ করলে ভাগফল হবে 0, কারন 2 দিয়ে 1 কে আর ভাগ যাবে না। ভাগশেষ হবে 1।
এবার নিচে থেকে উপরের দিকে ভাগশেষ গুলো পাশাপাশি লিখলে ফলাফল আসবেঃ
(123)10 = (1111011)2
ভগ্নাংশ
প্রশ্নের সংখ্যাটির ভগ্নাংশ হচ্ছে .456। এখন প্রতিবার ভগ্নাংশকে বাইনারির ভিত্তি 2 দিয়ে গুন করতে থাকব, যতক্ষন গুনফল শূন্য না আসে এবং পূর্ণ অংশ গুলোকে সংরক্ষন করব।

```
        │.456
        │  ×2
  ─┼────────
  0│.912
```

.456 কে 2 দিয়ে গুন করলে পাওয়া যায় 0.912। 0 কে সংরক্ষন করতে হবে এবং .912 কে আবার 2 দিয়ে গুন করতে হবে।

```
        │.456
        │  ×2
  ─┼────────
  0│.912
        │  ×2
  ─┼────────
  1│.824
```

.912 কে 2 দিয়ে গুন করলে পাওয়া যায় 1.824। 1 কে সংরক্ষন করতে হবে এবং .824 কে আবার 2 দিয়ে গুন করতে হবে।
এভাবে যতক্ষন ভগ্নাংশ 0 না আসবে ততক্ষন গুন করে যেতে হবে।

```
        │.456
        │  ×2
  ─┼────────
│  0│.912
│   │  ×2
│  ─┼────────
│  1│.824
│   │  ×2
│  ─┼────────
│  1│.648
```

```
|   |  ×2
|  ─┼───────
|  1|.296
|   |  ×2
|  ─┼───────
↓  0|.592
```

…

…

যেহেতু ৫টা উত্তর বের করার পরও গুনফল পূর্নসংখ্যা আসছে না, অর্থাৎ ভগ্নাংশ শূন্য আসছে না তাই আর না করলেও চলবে।

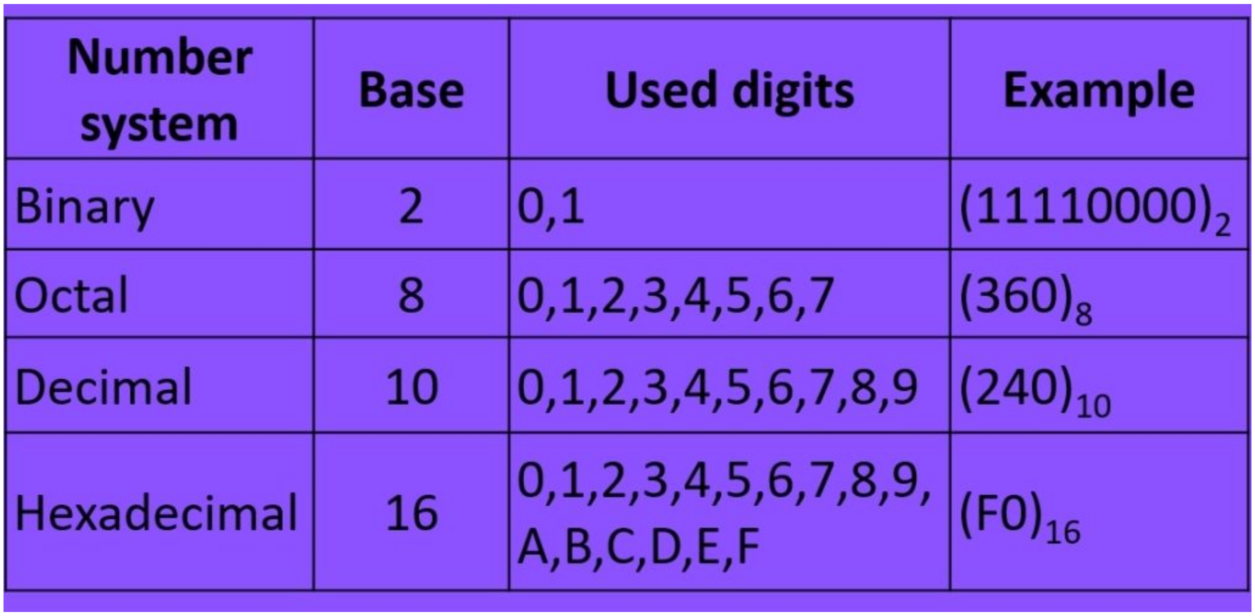
উপর থেকে নিচের দিকে পূর্ন সংখ্যা গুলো পাশাপাশি লিখলে ফলাফল আসবেঃ

(.456)10 = (.01110…)2

যেহেতু আরো গুন করা যাবে তাই ফলাফলের শেষে “…” দেয়া হয়েছে। গুনফল পূর্ন সংখ্যা পাওয়া গেলে এটি দিতে হবে না।

এখন পূর্ণসংখ্যা এবং ভগ্নাংশ একত্রে লিখলে চূড়ান্ত ফলাফল হবেঃ

(123.456)10 = (1111011.01110…)2



Number system	Base	Used digits	Example
Binary	2	0,1	$(11110000)_2$
Octal	8	0,1,2,3,4,5,6,7	$(360)_8$
Decimal	10	0,1,2,3,4,5,6,7,8,9	$(240)_{10}$
Hexadecimal	16	0,1,2,3,4,5,6,7,8,9, A,B,C,D,E,F	$(F0)_{16}$

খুবই গুরুত্বপূর্ন দুইটা আর্টিকেল -

1. http://subeen.com/%E0%A6%95%E0%A6%AE%E0%A7%8D%E0%A6%AA%E0%A6%BF%E0%A6%89%E0%A6%9F%E0%A6%BE%E0%A6%B0%E0%A7%87%E0%A6%B0-%E0%A6%AE%E0%A7%87%E0%A6%AE%E0%A7%8B%E0%A6%B0%E0%A6%BF
2. http://cpbook.subeen.com/2011/08/binary-number-system.html

বিটওয়াইজ অপারেটর

অপারেটর আছে দুই ধরণের, লজিকাল এবং বিটওয়াইজ। লজিকাল অপারেটর গুলো হল &&, || এবং !… আর বিটওয়াইজ অপারেটরগুলো হল সেই সব অপারেটর যারা বিটের উপর কাজ করে। কম্পিউটারে সবকিছুই সংরক্ষিত থাকে দ্বিমিক সংখ্যা হিসেবে। কম্পিউটার ০ এবং ১ ছাড়া কিছুই বুঝে না। এই প্রতিটি ০ কিংবা ১ হল এক একটি বিট। বিটওয়াইজ অপারেটরগুলো এইসব ০ এবং ১-এর প্রতিটির উপর আলাদা আলাদা ভাবে কাজ করে।

বিটওয়াইজ অপারেটরগুলো বেশ গুরুত্বপূর্ণ, কারণ গুণ-ভাগের বদলে এরা সরাসরি বিটের উপর কাজ করায় রানটাইম কমে যায়। এই অপারেটরগুলো শুধুমাত্র integer-এর উপর কাজ করবে, কোনো float কিংবা double-এর জন্য না। কারণ float কিংবা double সম্পূর্ণ ভিন্নভাবে মেমরিতে সংরক্ষিত থাকে।

এ ধরণের অপারেটর আছে ছয়টাঃ
(১) & (AND)
(২) | (OR)
(৩) ^ (XOR অথবা, Exclusive OR)
(৪) << (LEFT SHIFT)
(৫) >> (RIGHT SHIFT)
(৬) ~ (NOT অথবা, Complement)

এদের মধ্যে প্রথম ৫ টা হল বাইনারি অপারেটর, যারা দুইটি সংখ্যার উপর কাজ করবে। আর শেষেরটা হল Unary অপারেটর, যা একটি সংখ্যার উপরই কাজ করবে।

প্রথমেই আসি **AND-**এর কাছে**,**
বিট-ওয়াইজ AND(কোডে লিখবো &) অনেকটা লজিকাল AND (যাকে আমরা লিখি &&)-এর মতই কাজ করে। শুধু পার্থক্যটা হল এটি প্রতিটা বিটের উপর আলাদা আলাদাভাবে কাজ করে। ০ মানে হল মিথ্যা আর ১ মানে সত্য। তাহলে এই অপারেটর ১ রিটার্ন করবে শুধু তখনই যখন দুইটা সংখ্যার একই পজিশনের বিট ১ হবে, নাহলে এটি রিটার্ন করবে ০।

একটি উদাহারণ দিয়ে বুঝানোর চেষ্টা করা যাক।
ধর তোমার কাছে দুইটি ৮ সাইজের দুইটা unsigned ভ্যারিয়েবল আছে। সাইজ ৮ মানে যার বিট রয়েছে ৮ টি।

ধরা যাক, সংখ্যা দু'টি ৬৯ (a) এবং ৪২ (b)। এদেরকে বাইনারিতে রূপান্তর করলে পাবেঃ
69 = 01000101 /* প্রথম বিটটি শূন্য রয়ে গেছে, তার মানে এই না যে সেটা না লিখলেও হবে! */
42 = 00101010
এখন আমরা যদি এদের উপর বিটওয়াইজ AND চালিয়ে দিয়ে ভ্যালুটা AND নামের একটি ভ্যারিয়েবলে রাখতে চাই, তাহলে আমরা লিখবঃ

```
AND = a & b;
```

এটিকে আমরা এমনভাবেও লিখতে পারিঃ x = 42 & 69
কোনটা আগে, কোনটা পরে, সেটা কোনো ব্যাপার না!
এখন এই বিটওয়াইজ AND চালিয়ে দেওয়ার ফলে যা হবে তা হলঃ

```
   69 = 0100 0101
& 42 = 0010 1010
_________________
          0000 0000
```

Oops! শূণ্য হয়ে গেল, কারণ এদের কোনো বিটেই দুই সংখ্যাতেই শূন্য ছিল না! চল এবার অন্য দুইটা বাইনারি সংখ্যা নিয়ে চেষ্টা করা যাক! আমি খুব অলস তাই ডেসিম্যাল নাম্বার নিয়ে তা আবার বাইনারি করে দেখতে চাচ্ছি না বিটে মিল আছে কি না!

```
  1010 1100
&0110 0100

__________
 0010 0100
```

এবার এখানে কি হল সেটা বুঝার চেষ্টা করা যাক।
সংখ্যা দুইটার প্রথম বিটে একটাতে আছে ১, আরেকটাতে শূন্য, তাহলে রিটার্ন করবে ০।
দ্বিতীয় বিটেও একই কাহিনী। তৃতীয় বিটে দুইটা নাম্বারেই আছে ১, তাহলে রিটার্নও করবে ১!
বাকি বিটগুলা নিজে ব্যাখ্যা করে ফেল। হোমওয়ার্ক!

এবার পালা বিটওয়াইজ **OR-**এর**!**
আমরা && আর ||-এর পার্থক্য তো ইতোমধ্যেই জানি। তাহলে এখন বিটওয়াইজ AND আর OR-এর মধ্যে পার্থক্যও বুঝতে পারার কথা।

বিটওয়াইজ OR চালানোর জন্য আমরা লিখবোঃ

```
OR = a | b;
```

AND অপারেটর 1 রিটার্ন করতো ওই বিটে দুইটা সংখ্যাতেই 1 থাকলে। আর OR অপারেটর একটু দয়ালু। তাই সে ওই বিটটিতে দুইটা সংখ্যার যে কোনো একটাতেই 1 থাকলে রিটার্ন করে দিবে 1! আমাদের আগের উদাহারণটি দেখা যাক।

```
  1010 1100
|0110 0100

__________
 1110 1100
```

এখানে প্রথম বিটে দ্বিতীয় সংখ্যাতে 0 থাকলেও প্রথমটিতে 1 আছে, তাই রিটার্ন করবে 1…
আবার চতুর্থটিতে দুইটা সংখ্যাতেই শূন্য, তাই রিটার্নও করবে শূন্য।

এবার যাব আমরা **Exclusive OR-**এর কাছে**,**
বিটওয়াইজ OR-এর দূরসম্পর্কের আত্মীয় হল বিটওয়াইজ XOR. তবে এর মাথায় একটু সমস্যা। এটি দুইটা সংখ্যার যেকোনো একটিতে একই বিটে 1 থাকলে 1 রিটার্ন করে ঠিকই। কিন্তু সংখ্যার দু’টির দুইটিতেই 1 থাকলেই রিটার্ন করে 0!

আর XOR করার জন্য লিখতে হয় এভাবেঃ

```
XOR = a ^ b;
```

আমাদের আগের সেই দুইটা বাইনারি নাম্বারের কাছে ফিরে যাই আবার!

```
  1010 1100
^0110 0100

__________
 1100 1000
```

এখানে প্রথম, দ্বিতীয়, তৃতীয় আর পঞ্চম বিটে যেকোনো একটিতে 1 আছে, তাই রিটার্ন করেছে 1. আর ষষ্ঠ বিটে দুইটাতেই আছে 1, তাই রিটার্ন করে দিয়েছে 0. আর বাকি বিটগুলাতে তো কোনোটাতেই 1 নাই, ওগুলাতে রিটার্ন অবশ্যই 0!

এবার দুই ভাই **Left Shift** আর **Right Shift-**এর পালা**!**
এই দুইটা অপারেটরগুলার কাজ খুবই সহজ সরল ধরণের। এরা যেকোনো একটি সংখ্যার পুরা বিট প্যাটার্নকে নির্দিষ্ট সংখ্যক ঘর ডানে বা বামে সরিয়ে দেয়!

Right Shift লিখতে হয় এভাবেঃ

```
daaneshor = a >> numberofshifts;
```

numberofshifts হল তুমি যতবার শিফট করতে চাও, সেটা। ধরা যাক এই উদাহারনে এর মান 7. তাহলে,
প্রথম বার শিফট করে আমরা পাবঃ 01010101
দ্বিতীয় বার শিফট করে আমরা পাবঃ 00101010
তৃতীয় বার শিফট করে আমরা পাবঃ 00010101
চতুর্থ বার শিফট করে আমরা পাবঃ 00001010
বাকিগুলা হোমওয়ার্ক!

আবার Left Shift করার জন্য লিখতে হয়ঃ

```
baayeplastic = a << numberofshifts;
```

প্রথম বার শিফট করে আমরা পাবঃ 01101110
/* একটা জিনিস খেয়াল কর, যেহেতু আমরা ডাটা টাইপের সাইজ রেখেছিলাম ৮, তাই এখানে সবচেয়ে বামের একটি বিটে 1 ছিল, যা লেফট শিফট করার পর হারিয়ে যাচ্ছে। এটা খুবই চিন্তার বিষয়। তাই লেফট শিফট ব্যবহার করার সময় বিশেষভাবে খেয়াল রাখতে হবে ডাটা টাইপের সাইজের উপর! */
দ্বিতীয় বার শিফট করে আমরা পাবঃ 11011100
তৃতীয় বার শিফট করে আমরা পাবঃ 10111000
চতুর্থ বার শিফট করে আমরা পাবঃ 01110000

সবশেষে **NOT**
আগেই বলেছি এটা একটা Unary অপারেটর। অর্থাৎ এর জন্য আগের গুলার মত দুইটা সংখ্যা লাগবে না। NOT করার জন্য লিখতে হয় এভাবেঃ

```
NOT = ~69;
```

এই অপারেটর যা করে তা হল, বিটে 0 থাকলে 1 করে দেয় আর 1 থাকলে তাকে 0 করে দেয়! তাহলে আমরা 69-এর বাইনারি রিপ্রেজেন্টেশন 01000101-এর কমপ্লিমেন্ট কি হয় তা দেখি।
প্রথম বিটের 0 হয়ে যাবে 1
দ্বিতীয় বিটের 1 হয়ে যাবে 0
শেষ পর্যন্ত আমরা যে সংখ্যাটা পাব, তা হল 10111010, যার ডেসিম্যাল রিপ্রেজেন্টেশন হল 186!