# ICT PATHSHALA WITH KAUSHIK SIR

**Engr. Kaushik Saha**

**B.Sc in Electrical & Electronics Engineering (EEE)**

**Teaching Asst. at Islamic University**

**<u>Campus</u> : STUDY ZONE , Gate NO:7, Block:K, Halishahar.**

**Mobile: 01832221610**

## সি প্রোগ্রামিং এ অ্যারে

অ্যারে সম্পর্কে জানার আগ পর্যন্ত আমরা সাধারণত দুই একটা ভ্যারিয়েবল নিয়ে কাজ করেছি। কিন্তু বাস্তব জীবনে আমাদের এমন এমন প্রোগ্রাম লিখতে হতে পারে, যেখানে একের অধিক ভ্যারিয়েবল নিয়ে কাজ করতে হবে। একই টাইপের একের অধিক ভ্যারিয়েবল নিয়ে কাজ করার জন্য প্রোগ্রামিং এ আমরা অ্যারে ব্যবহার করি। যেমন প্রথম পাঁচটা প্রাইম নাম্বার হচ্ছে 2, 3, 5, 7, 11। এখন আমরা এই পাঁচটা নাম্বার আমাদের প্রোগ্রামে ব্যবহার করব। আমরা যদি সাধারণ ভ্যারিয়েবল এর মাধ্যমে ব্যবহার করতে চাই, তাহলে পাঁচটা ভ্যারিয়েবল ডিক্লেয়ার করতে হবে। কিন্তু অ্যারে ব্যবহার করে আমরা একটা ভ্যারিয়েবল ব্যবহার করেই এই পাঁচটা প্রাইম নাম্বার প্রোগ্রামে ব্যবহার করতে পারি।

অ্যারেকে মনে করতে পারেন একটা বাক্স এর মত, যার মধ্যে অনেক গুলো ছোট ছোট খোপ থাকতে পারে নিচের টেবিলটি দেখুন।

| | | | | |
|---|---|---|---|---|
| | | | | |

এখন আমরা প্রথম পাঁচটি প্রাইম নাম্বার প্রোগ্রামে রাখার জন্য আমরা ৫টি খোপ যুক্ত একটি অ্যারে নিব। অ্যারের প্রতিটি খোপে একটি করে প্রাইম নাম্বার রাখবে। তাহলে অ্যারেটি দেখাবে নিচের মত করেঃ

| | | | | |
|---|---|---|---|---|
| **2** | **3** | **5** | **7** | **11** |

অ্যারের প্রতিটা খোপকে বলা হয় ইনডেক্স। আর সি প্রোগ্রামিং এ এই ইনডেক্স গণনা শুরু হয় 0 থেকে। অ্যারেতে প্রতিটা ডেটা আইটেম যখন কম্পিউটার মেমরিতে সংরক্ষণ করে, তা মূলত একটা মেমরি লোকেশনে রাখে। নির্দিষ্ট সময় কম্পিউটার মেমরির যে লোকেশন খালি থাকবে, ঐ লোকেশনেই ডেটা গুলো সংরক্ষণ করবে। নিচের ছবিটি লক্ষ্য করিঃ

| মেমরি এড্রেস | 10001 | 10002 | 10003 | 10004 | 10005 |
|---|---|---|---|---|---|
| **ডেটা আইটেম** | 2 | 3 | 5 | 7 | 11 |
| **অ্যারে ইনডেক্স** | 0 | 1 | 2 | 3 | 4 |

এখানে আমরা 5 সাইজের একটা অ্যারে নিয়ে সেখানে প্রথম পাঁচটি প্রাইম নাম্বার রেখেছি। যার 0 ইনডেক্সে রয়েছে 2। যা অ্যারের প্রথম আইটেম। কোন কোন লেখক যাকে ইলিম্যান্টও বলে। এই অ্যারের 1 নং ইনডেক্সে রয়েছে 3। যা হচ্ছে অ্যারের দ্বিতীয় আইটেম। এভাবে 4 নং ইনডেক্স বা পঞ্চম এবং শেষ আইটেম হচ্ছে 11। উপরের অ্যারেতে আমরা মাত্র ৫টা আইটেম রেখেছি। ৫টার পরিবর্তে এভাবে একটা অ্যারেতে অসংখ্য আইটেম রাখতে পারি।

**অ্যারে ডিক্লেয়ার করাঃ**

অ্যারে ব্যবহার করার জন্য প্রথমে অ্যারে ডিক্লেয়ার করতে হয়। অ্যারে অন্যান্য সাধারণ ভ্যরিয়েবলের মতই ডিক্লেয়ার করা হয়। যেমন

```
data_type array_name[size]
```

ভ্যারিয়েবলের মত অ্যারেরও একটা নাম দিতে হয়। ডেটা টাইপে বলে দিতে হয় অ্যারেতে আমরা কি ধরনের ডেটা রাখব। যদি ইন্টিজার রাখি, তাহলে হবে int, যদি ক্যারেক্টার রাখি তাহলে বলে দিতে হবে char। ইন্টিজার অ্যারেতে কারেক্টার রাখা যাবে না। আবার ইন্টিজার অ্যারেতে ফ্লোটিং পয়েন্টও রাখা যাবে না। তবে ফ্লোটিং পয়েন্ট অ্যারেতে আমরা চাইলে ইন্টিজার ভ্যালু রাখতে পারব।

নামের পাশা পাশি অ্যারের সাইজ বলে দিতে হয়। সাইজ বলতে অ্যারেতে কয়টা আইটেম আমরা রাখতে চাচ্ছি, তা। যেমন পাঁচটা প্রাইম নাম্বার রাখার জন্য আমরা নিচের মত করে একটা অ্যারে লিখতে পারিঃ

```
int prime[5];
```

এখানে আমরা prime নামে ৫টি আইটেম এর একটা অ্যারে তৈরি করেছি। মানে এর মধ্যে আমরা সর্বোচ্চ ৫টি ইন্টিজার নাম্বার রাখতে পারব। কম রাখলে সমস্যা নেই। তবে ডিক্লেয়ার করা সাইজের বেশি আইটেম রাখা যায় না। ইন্টিজারের পাশাপশি এভাবে আমরা অন্য ডেটা টাইপের জন্যও অ্যারে তৈরি করতে পারি নিচের মত করেঃ

```
float gpa[100];
char name[30];
```

**অ্যারে ইনিশিয়ালাইজঃ**

আমরা ভ্যারিয়েবল তৈরির সময় যেমন একটা ভ্যালু এসাইন করে দিতে পারি, অ্যারে তৈরির সময়ও এর আইটেম গুলো বলে দিতে পারিঃ

*int prime[5] = {2, 3, 5, 7, 11};*

যাকে বলা অ্যারে ইনিশিয়ালাইজ। অ্যারেটি তৈরি করার সময় আমরা তার ডেটা আইটেম গুলোও বলে দিয়েছি। অ্যারের আইটেম একটার থেকে আরেকটার মধ্যে পার্থ্যকের জন্য কমা ব্যবহার করতে হয়। আর ভ্যালু গুলো

লেখা হয় দ্বিতীয় ব্র্যাকেটের মধ্যে।

উপরে আমরা এক লাইনে অ্যারে ইনিশিয়ালাইজ করেছি। আমরা চাইলে আলাদা আলাদা ভাবে ইনিশিয়ালাইজ করতে পারি, নিচের মত করেঃ

```
int prime[5];
prime[0] = 2;
prime[1] = 3;
prime[2] = 5;
prime[3] = 7;
prime[4] = 11;
```

**ফ্লোটিং পয়েন্ট অ্যারেঃ**

```
float gpa[4] = {3.5, 3.8, 3.9, 2.9};
```

অ্যারে ডিক্লেয়ারের সময়  এর আইটেম গুলো বলে দিলে অ্যারে সাইজ না বললেও সমস্যা হবে না। যেমনঃ

```
float gpa[] = {3.5, 3.8, 3.9, 2.9};
```

**অ্যারে এক্সেস করাঃ**

অ্যারে থেকে কোন আইটেম বের করাকে বলে অ্যারে এক্সেস। অ্যারের ইন্ডেক্সিং শুরু হয় ০ থেকে।  নিচের অ্যারেটি দেখিঃ

```
int prime[5] = {2, 3, 5, 7, 11}
```

এই অ্যারেটির প্রথম আইটেম মানে ০ তম ইন্ডেক্সে রয়েছে 2। দ্বিতীয় আইটেম বা ১তম ইন্ডেক্সে রয়েছে ৩। এভাবে চতুর্থ আইটেম বা ৩ তম ইন্ডেক্সে রয়েছে ১১।

আমরা যদি লিখিঃ prime[0], এটি আমাদের 2 রিটার্ণ করবে। যদি লিখি prime[1] এটি রিটার্ণ করবে 3। prime[3] লিখলে রিটার্ণ করবে 11। নিচের প্রোগ্রামটি দেখিঃ

```
#include <stdio.h>
int main()
{
  int prime[5] = {2, 3, 5, 7, 11};
```

PRESENTED BY ICT PATHSHALA WITH KAUSHIK SIR

```
   printf("%d",prime[0]);

   return 0;

}
```

উপরের প্রোগ্রামে আমরা অ্যারের একটা আইটেম শুধু প্রিন্ট করেছি। আপনি এখন printf ফাংশনের ভেতরে লেখা prime[0] এর পরিবর্তে 1, 2, 3 ইত্যাদি লিখে দেখুন । দেখবেন একবার এক একটা ইনডেক্সের আইটেম প্রিন্ট করবে।

এর আগে আমরা লুপ শিখে এসেছি। আমরা যদি অ্যারের সব গুলো আইটেম এক্সেস করতে চাই, তাহলে লুপ ব্যবহার করতে পারি। যেমন এভাবেঃ

```
#include <stdio.h>

int main()

{

  int i;

  int prime[5] = {2, 3, 5, 7, 11};

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

  printf("%d \n",prime[i]);

  return 0;

}
```

PRESENTED BY
ICT PATHSHALA
WITH KAUSHIK SIR

এখানে যদি আমাদের অ্যারেতে একশ আইটেম থাকে, আমরা সহজেই তা এক্সেস করতে পারব। শুধু i<5 এর পরিবর্তে i<100 লিখলেই হবে।

আচ্ছা আমরা এখন একটা প্রোগ্রাম চিন্তা করি যেটা ইউজার থেকে ৬টা নাম্বার নিবে এবং পরে তা যোগ করে রেজাল্ট আমাদের দেখাবে। এটা সাধারন পদ্ধতিতে করতে গেলে আমাদের আগে ৬টা ভ্যারিয়েবল নিতে হত, তারপর সেগুলোকে যোগ করতে হত তারপর যোগফল দেখাতে হতো।

এখন আমরা কত সহজেই এ জিনিসটা করতে পারব মাত্র একটি ভ্যারিয়েবল নিয়েঃ

```
#include <stdio.h>

int main()
```

```
{
  int number[6];
  int i, result=0;

  for(i=0; i<6; i++){
    printf("Enter %d no Number:\n", i+1);
    scanf("%d", &number[i]);
    result +=number[i];
  }

  printf("Result is: %d", result);
  return 0;
}
```

আমরা চাইলে কোন কোন নাম্বার গুলো ব্যবহারকারী ইনপুট দিয়েছে, সেগুলো প্রিন্টও করুতে পারিঃ

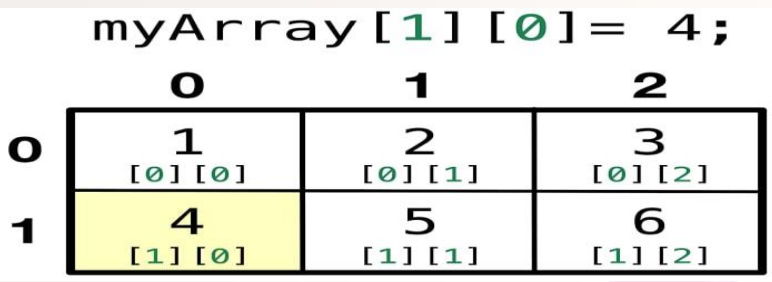
```
#include <stdio.h>
int main() {
  int number[6];
  int i, result=0;
  for(i=0; i<6; i++){
    printf("Enter %d no Number:\n", i+1);
    scanf("%d", &number[i]);
    result +=number[i];
  }
  printf("Result is: %d \n", result);
```

PRESENTED BY ICT PATHSHALA WITH KAUSHIK SIR

```
  for(i=0; i<6; i++){

    printf("Array element %d is: %d\n", i , number[i]  );

  }

  return 0;

}
```

**2D অ্যারে**

আমরা এতক্ষণ যে অ্যারে নিয়ে আলোচনা করেছি, তা ছিল ওয়ান ডাইমেনশনাল অ্যারে। অ্যারে মাল্টি ডাইমেশনাল হতে পারে। যেমন 2D, 3D ইত্যাদি। টু ডাইমেনশনাল অ্যারে দেখতে নিচের মত:




PRESENTED BY

**2D অ্যারে ডিক্লেয়ার করা হয় এভাবেঃ**

```
1 data_type array_name[i][j];
```

এখানে i এবং j এর মান যে কোন সংখ্যা হতে পারে। অ্যারের সাইজ হবে i*jটি। মানে অ্যারেটিতে i*jটি আইটেম রাখা যাবে। এখন i যদি 4 হয়, j যদি 4 হয়, তাহলে ঐ অ্যারেতে মোট 16টি আইটেম রাখা যাবে। এখানে মূলত 2D অ্যারের ইনডেক্স গুলো দেখিয়েছি। প্রথম আইটেম এক্সেস করতে হলে লিখতে হবে [0][0]। দ্বিতীয় আইটেম এক্সেস করতে হলে লিখতে হবে [0][1] এভাবে।

এবার আমরা একটা ক্যারেক্টার অ্যারে দেখি alfabet নামে। যার সাইজ হচ্ছে 4*4। এর মানে এর মধ্যে মোট ১৬টি আইটেম রাখা যাবে। এখন এই alfabet অ্যারে নিচের মত ইনিশিয়ালাইজ করিঃ

```
char alfabet[4][4] = {'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I',
    'J', 'K', 'L', 'M', 'N', 'O', 'P'};
```

alfabet[4][4] এর ক্ষেত্রে এখানে প্রথম হচ্ছে row সংখ্যা, দ্বিতীয়টি হচ্ছে Column সংখ্যা। অর্থাৎ এখানে ৪টি রো রয়েছে এবং ৪টি কলাম রয়েছে, সর্বোমোট আইটেমসংখ্যা হচ্ছে ১৬।

উপরের alfabet নামক অ্যারে থেকেঃ

- প্রথম আইটেম পেতে হলে আমাদের লিখতে হবে alfabet[0][0] এবং তা হচ্ছে A

- দ্বিতীয় আইটেম পেতে হলে আমাদের লিখতে হবে alfabet[0][1] এবং তা হচ্ছে B
- তৃতীয় আইটেম পেতে হলে আমাদের লিখতে হবে alfabet[0][2] এবং তা হচ্ছে C

এভাবে বাকি গুলো।

একটা উদাহরণ দেখিঃ

```
#include <stdio.h>

int main() {

  char alfabet[4][4] = {'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I',

    'J', 'K', 'L', 'M', 'N', 'O', 'P'};

  printf("%c \n",alfabet[0][0]);

  return 0;

}
```

উপরের প্রোগ্রাম রান করলে আউটপুট পাবো A। এখন যদি alfabet[3][3] লিখে রান করে দেখি, তাহলে দেখব আউটপুট দিচ্ছে P।

2D অ্যারেতে কোন কিছু রাখতে হলে এভাবে একই ধাপ অনুসরন করতে হবে। যেমনঃ

প্রথম ইনডেক্সে একটা আইটেম রাখব, তার জন্য লিখতে হবেঃ

```
alfabet[0][0] = 'Z';
```

দ্বিতীয় ইনডেক্সে একটা আইটেম রাখার জন্য লিখতে হবে:

```
alfabet[0][1] = 'Y';
```

এভাবে বাকি গুলো।

উপরের উদাহরন কঠিন মনে হলে আরো সহজে চিন্তা করি। যেমন int num [2][2] নামে একটা অ্যারে নিব আমরা। এটিতে আমরা সর্বোচ্চ 4টা আইটেম রাখতে পারব। যেমন আমরা একটা ইন্টিজার অ্যারে নিলামঃ

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

int num [2][2] = {1, 2, 3, 4}

কম্পিউটার এটিকে নিচের মত করে রাখবেঃ

PRESENTED BY ICT PATHSHALA WITH KAUSHIK SIR

| | |
|---|---|
| 1 | 2 |
| 3 | 4 |

যা আমরা int num [2][2] = {1, 2, 3, 4} এর পরিবর্তে এভাবেও এসাইন করতে পারবঃ

```
int num[0][0] = 1;

int num[0][1] = 2

int num[1][0] = 3;

int num[1][1] = 4;
```

অথবা প্রতিটা রো আলাদা আলাদা করেও এসাইন করতে পারি নিচের মত করে:

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

সম্পূর্ণ প্রোগ্রামঃ

```
#include <stdio.h>

int main() {
   int num[2][2] = {
      {1,2},
      {3,4}
   };
   printf("%d \n",num[0][1]);
   return 0;
}
```

যা আউটপুট দিবেঃ 2

আবার int num [3][3] নামে একটা অ্যারে নিলে আমরা সর্বোচ্চ ৯টা আইটেম রাখতে পারব। নিচের মত করেঃ

```
int num [3][3] = {1, 2, 3, 4, 5, 6, 7, 8, 9};
```

প্রোগ্রামটি মেমরিতে নিচের মত করে রাখবেঃ

অর্থাৎ

```
int[0][0] = 1;
int[0][1] = 2;
int[0][2] = 3;

int[1][0] = 4;
int[1][1] = 5;
int[1][2] = 6'

int[2][0] = 7;
int[2][1] = 8;
int[2][2] = 9;
```

PRESENTED BY
ICT
PATHSHALA
WITH KAUSHIK SIR

যা আমরা এভাবেও ইনিশিয়ালাইজ করতে পারিঃ

```
int num[3][3] = {
    {1,2,3},
    {4,5,6},
    {7,8,9}
};
```

সম্পূর্ণ প্রোগ্রামঃ

```c
#include <stdio.h>
int main() {
  int num[3][3] = {
    {1,2,3},
    {4,5,6},
    {7,8,9}
  };
  printf("%d \n",num[0][1]);
  return 0;
}
```

প্রোগ্রামে আমরা দুইটি for লুপ ব্যবহার করে সহজেই 2d অ্যারে ব্যবহার করতে পারি। নিচের প্রোগ্রামটি দেখুনঃ

```c
#include <stdio.h>
int main()
{
  int i,j;
  int num[2][2] = {10,20,30,40};
  for (i=0;i<2;i++)
  {
    for (j=0;j<2;j++)
    {
      printf("value of num[%d] [%d] : %d\n",i,j,num[i][j]);
    }
  }
}
```

PRESENTED BY ICT PATHSHALA WITH KAUSHIK SIR

যা রান করে দেখলে নিচের মত আউটপুট পাবেনঃ

1value of num[0] [0] : 10

2value of num[0] [1] : 20

3value of num[1] [0] : 30

4value of num[1] [1] : 40

একই প্রোগ্রাম আমরা নিচের মত করেও লিখতে পারিঃ

```
#include<stdio.h>
int main()
{
   int i,j;
   int num[2][2] = {
      {10,20},
      {30,40}
      };
   for (i=0;i<2;i++)
   {
      for (j=0;j<2;j++)
      {
         printf("value of num[%d] [%d] : %d\n",i,j,num[i][j]);
      }
   }
}
```

এখানে অ্যারে ইনিশিয়ালাইজ অন্য ভাবে করেছি। রান করে দেখলে একই আউটপুট পাবো।

PRESENTED BY
ICT PATHSHALA
WITH KAUSHIK SIR

উপরের প্রোগ্রামে আমরা একটা স্ট্যাটিক 2D অ্যারে নিয়ে কাজ করেছি। এবার আমরা ব্যবহাররী থেকে ইনপুট নিয়ে অ্যারেতে ডেটা রাখবঃ

```
#include<stdio.h>
int main()
{
  int i,j;

  int num[2][2];
  // reading value
  for (i=0;i<2;i++)
  {
    for (j=0;j<2;j++)
    {
      printf("Enter value of num[%d] [%d]:",i,j);
      scanf("%d", &num[i][j]);
    }
  }

  // printing value
  for (i=0;i<2;i++)
  {
    for (j=0;j<2;j++)
    {
      printf("You entered value of num[%d] [%d] : %d\n",i,j,num[i][j]);
    }
```

```
  }
  return 0;
}
```

এখানে আমরা 2*2 সাইজে একটা অ্যারে নিয়েছি। আপনি চাইলে যে কোন সাইজের অ্যারে নিয়ে কাজ করতে পারেন।

অ্যারে ইনিশিয়ালাইজ করার সময় যদি আমরা কোন ভ্যালু না দিয়ে থাকি, তাহলে সেটা নাল অথা শূন্য দিয়ে পূর্ণ হবে। নিচের অ্যারে ইনিশিয়ালাইজটা দেখিঃ

```
int num[3][3] = {
  {1,2},
  {4,5,6},
  {7,8}
};
```

আমরা 3*3 সাইজের একটা অ্যারে নিয়েছি। কিন্তু সবগুলোতে ভ্যালু এসাইন করি নি। যে সব পজিশনে আমরা কোন ভ্যালু এসাইন করিনি, সে সব পজিশনে শূন্য পাবো। নিচের প্রোগ্রামটি রান করে দেখতে পারিঃ

```
#include<stdio.h>
int main()
{
  int i,j;
 int num[3][3] = {
    {1,2},
    {4,5,6},
    {7,8}
  };
  for (i=0;i<3;i++)
  {
```

PRESENTED BY ICT PATHSHALA WITH KAUSHIK SIR

```
    for (j=0;j<3;j++)

    {


       printf("value of num[%d] [%d] : %d\n",i,j,num[i][j]);

    }

  }
```

**ফাংশনে অ্যারে পাস করাঃ**

আমরা চাইলে একটা অ্যারেকে কোন ফাংশনের আর্গুমেন্ট হিসেবে পাস করতে পারি। নিচের প্রোগ্রামটি দেখিঃ

```
#include<stdio.h>

int getSum(int arr[]) {

 int sum = 0;

 int i;

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

    sum =sum+ arr[i];

  }

  return sum;

}

int main () {


  int balance[5] = {25, 2, 3, 17, 50};


  printf( "Sum is: %d ", getSum(balance) );


  return 0;

}
```

প্রোগ্রামটি রান করলে আউটপুট পাবোঃ Sum is: 97

এখানে আমরা getSum নামে একটা ফাংশন লিখেছি। যার একটি প্যারামিটার হিসেবে রয়েছে একটি ইন্টিজারঅ্যারে।

**Made by**

**Engr. Kaushik Saha**

**Teaching Asst. at Islamic University**