C Array – অ্যারে

এর আগের প্রোগ্রাম গুলোতে আমরা দুই একটা ভ্যারিয়েবল নিয়ে কাজ করেছি। 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।

১	২	৩	৪	৫	…	n

এখন আমরা উপরের পাঁচটি প্রাইম নাম্বার আমাদের প্রোগ্রামে রাখার জন্য আমরা ৫টি খোপ যুক্ত একটি বাক্স নিব। এবং প্রতিটি খোপে একটি করে প্রাইম নাম্বার নিব। নিচের মত করেঃ

২	৩	৫	৭	১১

৫টার পরিবর্তে এভাবে মাত্র একটা অ্যারেতে অসংখ্য প্রাইম নাম্বার আমরা রাখতে পারি। আমরা যখন অ্যারেকে বলব, এক নং খোপে কি আছে, তখন তা আমাদের ২ রিটার্ণ করবে। তেমনি যদি বলি ৫ নং খোপে কি আছে, তখন সে ১১ রিটার্ণ করবে।

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]

ডেটা টাইপে বলে দিতে হয় অ্যারেতে আমরা কি ধরনের ডেটা রাখব। যদি ইণ্টিজার 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।

যে কো একটা নাম দিতে হবে অ্যারের। এরপর সাইজ। সাইজ বলতে অ্যারেতে কয়টা ইলিম্যান্ট আমরা রাখব। যেমন পাঁচটা প্রাইম নাম্বার রাখার জন্য আমরা নিচের মত করে একটা অ্যারে লিখতে পারিঃ

int prime[5];

এখানে আমরা prime নামে ৫টি ইলিম্যান্ট এর একটা অ্যারে তৈরি করেছি। মানে এর মধ্যে আমরা ৫টি ইন্টিজার নাম্বার রাখব। 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 ইলিম্যান্ট গুলো বলে দিতে পারিঃ

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

যাকে বলা অ্যারে ইনিশিয়ালাইজ। অ্যারেটি তৈরি করার সময় আমরা তার ইলিম্যান্ট বা উপাদান গুলোও বলে দিয়েছি। অ্যারের উপাদান 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 ইলিম্যান্ট গুলো বলে দিলে অ্যারে সাইজ না বললেও সমস্যা হবে না। যেমনঃ

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

অ্যারে এক্সেস করাঃ

অ্যারের কোন উপাদান 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 উপাদান মানে ০ তম ইন্ডেক্সে রয়েছে 2। দ্বিতীয় উপাদান বা ১তম ইন্ডেক্সে রয়েছে ৩। এভাবে চতুর্থ উপাদান বা ৩ তম ইন্ডেক্সে রয়েছে ১১।

আমরা যদি লিখিঃ prime[0], এটি আমাদের ২ 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};

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

    return 0;
}
```

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

এর আগে আমরা লুপ শিখে এসেছি। আমরা যদি অ্যারের সব গুলো ইলিমেন্ট 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;
}
```

এখানে যদি আমাদের অ্যারেতে একশটিও ভ্যারিয়েবল 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);

    for(i=0; i<6; i++){
        printf("Array element %d is: %d\n", i , Number[i]  );

    }


    return 0;

}
```

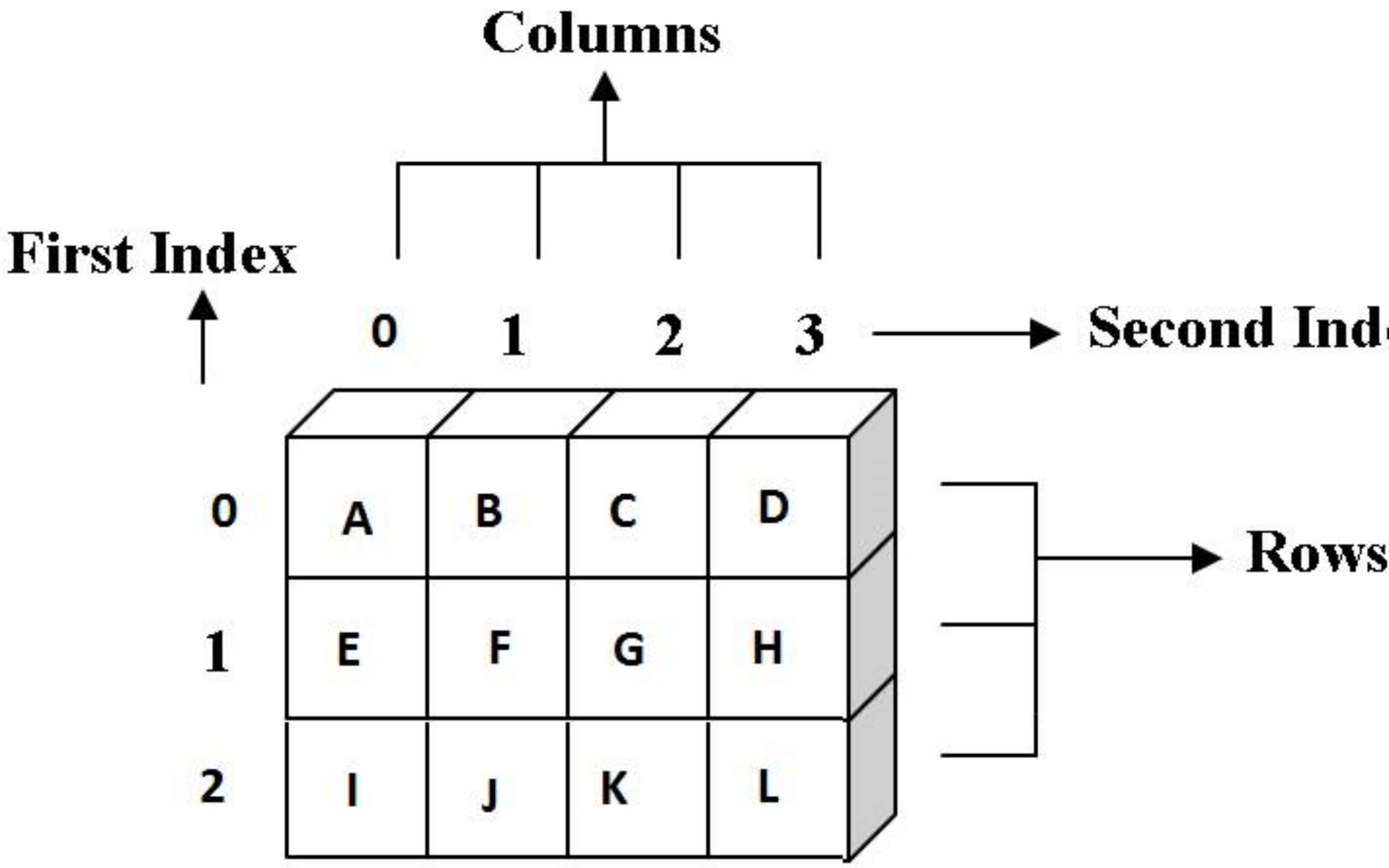
2D অ্যারে

আমরা এতক্ষণ যে অ্যারে নিয়ে আলোচনা করেছি, তা ছিল ওয়ান ডাইমেনশনাল অ্যারে। অ্যারে মাল্টি ডাইমেশনাল হতে পারে। যেমন 2**D**, 3**D ইত্যাদি।** Two Dimensional array নিচের মত।

অ্যারে ডিক্লেয়ার করা হয় এভাবেঃ

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 ইলিম্যান্ট রাখা যাবে। এখন i যদি ৫ হয়, j যদি ৬ হয়, তাহলে ঐ অ্যারেতে মোট ৩০টি ইলিম্যান্ট রাখা যাবে। নিচের ছবিটি দেখিঃ



Two-dimensional array with 12 elements

এটি একটি Two Dimensional Character Array যার উপাদান সংখ্যা ১২। মনে করি Array টির নাম Character এবং এটিকে লিখতে হবে Character[3][4] এখানে প্রথমটা হচ্ছে row সংখ্যা, দ্বিতীয়টি হচ্ছে Column সংখ্যা। অর্থাৎ এখানে ৩টি রো রয়েছে এবং ৪টি কলাম রয়েছে, সর্বোমোট উপাদাম সংখ্যা হচ্ছে ১২।

প্রথম উপাদান অর্থাৎ উপরের Character নামক Two Dimensional Array থেকেঃ

প্রথম উপাদানটি পেতে হলে আমাদের লিখতে হবে Character[0][0] এবং তা হচ্ছে A

দ্বিতীয় উপাদানটি পেতে হলে আমাদের লিখতে হবে Character[0][1] এবং তা হচ্ছে B

তৃতীয় উপাদানটি পেতে হলে আমাদের লিখতে হবে Character[0][2] এবং তা হচ্ছে C

চতুর্থ উপাদানটি পেতে হলে আমাদের লিখতে হবে Character[0][3] এবং তা হচ্ছে D

পঞ্চম উপাদানটি পেতে হলে আমাদের লিখতে হবে Character[1][0] এবং তা হচ্ছে E
ষষ্ঠ উপাদানটি পেতে হলে আমাদের লিখতে হবে Character[1][1] এবং তা হচ্ছে F
সপ্তম উপাদানটি পেতে হলে আমাদের লিখতে হবে Character[1][2] এবং তা হচ্ছে G
এভাবে আমরা পরের উপাদান গুলোও পেতে পারি।

কোন কিছু রাখতে হলে এ ভাবে একই ধাপ অনুসরন করতে হবে।
প্রথম ইনডেক্সে একটা Character রাখব, তার জন্য লিখতে হবে Character[0][0]
আগেই বলেছি Array Index শূন্য থেকে শুরু হয়।
দ্বিতীয় ইনডেক্সে একটা Character রাখার জন্য লিখতে হবে Character[0][1]
এভাবে বাকি গুলো রাখতে হবে।

উপরের উদাহরন কঠিন মনে হলে আরো সহজে চিন্তা করি। যেমন int num [2][2] নামে একটা অ্যারে নিব আমরা। এটিতে আমরা সর্বোচ্চ ৪টা ইলিমেন্ট রাখতে পারব। যেমন আমরা একটা ইন্টিজার অ্যারে নিলামঃ

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

কম্পিউটার এটিকে নিচের মত করে রাখবেঃ

1	2
3	4

```
int[0][0] = 1;
int[0][1] = 2
int[1][0] = 3;
int[1][1] = 4;
```

এখন যদি আমরা int[0][1] এ কি রয়েছে, তা দেখি, আমরা দেখব 2 রয়েছে। যা আমরা এভাবেও লিখতে পারিঃ

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

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

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

প্রোগ্রামটি এটিকে নিচের মত করে রাখবেঃ

1	2	3
4	5	6
7	8	9

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;
```

যা আমরা এভাবেও ইনিশিয়ালাইজ করতে পারিঃ

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

```

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

```
#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]);
        }
    }
}
```

যা রান করে দেখলে নিচের মত আউটপুট পাবেনঃ

```
value of num[0] [0] : 10
value of num[0] [1] : 20
value of num[1] [0] : 30
value of num[1] [1] : 40
```

নিচের প্রোগ্রামটি দেখুনঃ

```
#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।

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

```
#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
        }
    }
    return 0;
}
```

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

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

```
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 পারোঃ

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

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

return 0;

}



<strong>ফাংশনে অ্যারে পাস করাঃ</strong>

<strong> </strong>

<strong> </strong>

আমরা চাইলে একটা অ্যারে একটা ফাংশনে আর্গুমেন্ট হিসেবে পাস করতে পারি। নিচের প্রে


#include <stdio.h>

int getSum(int arr[]) {

```

```
  int sum = 0;
  int i;

   for (i = 0; i < 5; ++i) {
      sum += arr[i];
   }
   return sum;
}


int main () {

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

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

   return 0;
}
```