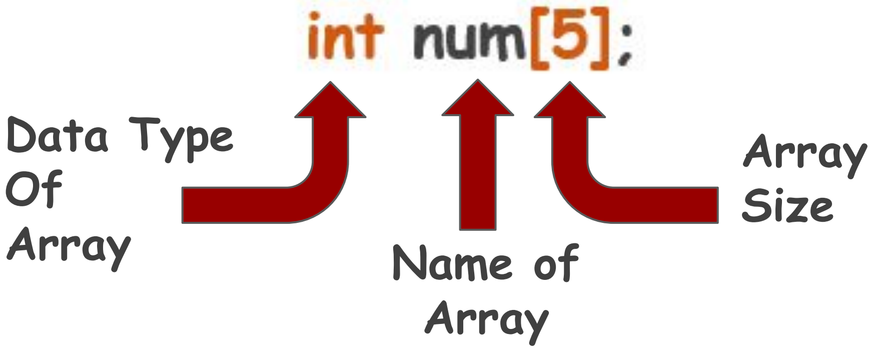
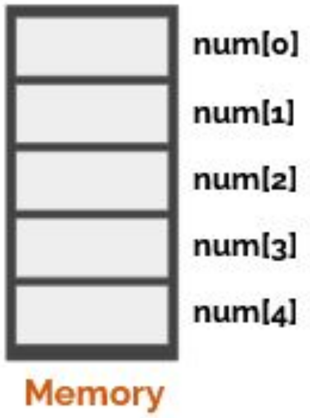
Parallel Arrays

Review: Arrays in C++

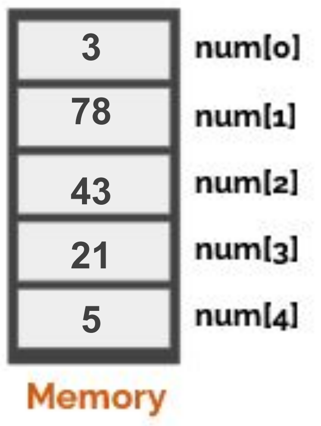
A one-dimensional array is an array in which the components are arranged in a list form. In C++, the array index starts at 0.





Independent Arrays

Previously, all the arrays (int, float or char i.e., string) were independent and the data within that array did not depend on any other data.



Dependent Arrays

Sometimes, we have to deal with different types of data that are related to other data.
Suppose, we have to save the following information for a set of students.

Name	Roll Number
Jack	2312
John	1111
Ibrahim	2121

Dependent Arrays

How to store this data in arrays. Should we declare **single array** or **multiple arrays** ?

Name	Roll Number
Jack	2312
John	1111
Ibrahim	2121

Dependent Arrays

We can not use single array because the data in the arrays are of different types

string int

Name	Roll Number
Jack	2312
John	1111
Ibrahim	2121

Dependent Arrays

These two are different types of data both semantically and syntactically. Semantically means Name and Roll number are separate information. Syntactically means the data type of both data is different.

Name	Roll Number
Jack	2312
John	1111
Ibrahim	2121

Dependent Arrays

Whenever we come across such scenarios when the data is correlated but represent different information, we have to use two separate arrays.

Name	Roll Number
Jack	2312
John	1111
Ibrahim	2121

Dependent Arrays

We can use 2 separate arrays but **how to link** both information in the arrays?

Name	Roll Number
Jack	2312
John	1111
Ibrahim	2121

Dependent Arrays

We can use two separate arrays and store related data at the same index of arrays.

Index	Name	Roll Number
0	Jack	2312
1	John	1111
2	Ibrahim	2121

Dependent Arrays

```
string Names[3] = {"Jack", "John", "Ibrahim"};
int IDs[3] = {2312, 1111, 2121};
```

Index	Name	Roll Number
0	Jack	2312
1	John	1111
2	Ibrahim	2121

Dependent Arrays

```
string Names[3] = {"Jack", "John", "Ibrahim"};
int IDs[3] = {2312, 1111, 2121};
```

Name Jack is on 0 index and his roll number is also on 0 index

Index	Name	Roll Number
0	Jack	2312
1	John	1111
2	Ibrahim	2121

Dependent Arrays

```
string Names[3] = {"Jack", "John", "Ibrahim"};
int IDs[3] = {2312, 1111, 2121};
```

Name John is on 1 index and his roll number is also on 1 index

Index	Name	Roll Number
0	Jack	2312
1	John	1111
2	Ibrahim	2121

Parallel Arrays

```
string Names[3] = {"Jack", "John", "Ibrahim"};
int IDs[3] = {2312, 1111, 2121};
```

These arrays hold related information at same indexes, therefore these arrays are called Parallel Arrays.

Index	Name	Roll Number
0	Jack	2312
1	John	1111
2	Ibrahim	2121

Parallel Arrays

```
string Names[3] = {"Jack", "John", "Ibrahim"};
int IDs[3] = {2312, 1111, 2121};
```

What If we want to store GPA of the student as well?

Index	Name	Roll Number
0	Jack	2312
1	John	1111
2	Ibrahim	2121

Parallel Arrays

```
string Names[3] = {"Jack", "John", "Ibrahim"};
int IDs[3] = {2312, 1111, 2121};
```

What If we want to store GPA of the student as well?

Index	Name	Roll Number	GPA
0	Jack	2312	3.9
1	John	1111	3.2
2	Ibrahim	2121	3.4

Parallel Arrays

```
string Names[3] = {"Jack", "John", "Ibrahim"};
int IDs[3] = {2312, 1111, 2121};
float GPA[3] = {3.9, 3.2, 3.4};
```

Index	Name	Roll Number	GPA
0	Jack	2312	3.9
1	John	1111	3.2
2	Ibrahim	2121	3.4

Working Example

Write a C++ program that takes the data of the students (Name, Roll Number, and GPA) as input from the user and then displays that data in a table format on the console.

Solution

There are two ways to take input from the user in this case.

Method 1: Ask the user how many records he/she wants to enter before taking input.
Method 2: Keep on taking input until user says he/she doesn't want to enter more.

Method 1

First ask the user **how many records** he/she wants to enter.
Then run your loop according to the user input.

```
#include <iostream>
using namespace std;
main()
{
    string Names[100];
    int IDs[100];
    float Gpa[100];
    int count;
    cout << "How many records you want to enter: ";
    cin >> count;
    for (int idx = 0; idx < count; idx = idx + 1) //For Taking Input
     {
      }
}
```

Method 1

First ask the user **how many records** he/she wants to enter.
Then run your loop according to the user input and take input.

```
#include <iostream>
using namespace std;
main()
{
    string Names[100];
    int IDs[100];
    float Gpa[100];
    int count;
    cout << "How many records you want to enter: ";
    cin >> count;
    for (int idx = 0; idx < count; idx = idx + 1) //For Taking Input
    {
        cout << "Name: ";
        cin >> Names[idx];
        cout << "Roll Number: ";
        cin >> IDs[idx];
        cout << "GPA: ";
        cin >> Gpa[idx];
    }

}
```

Method 1

First ask the user **how many records** he/she wants to enter.
Then run your loop according to the user input and take input.
Then display the output.

```
#include <iostream>
using namespace std;
main()
{
    string Names[100];
    int IDs[100];
    float Gpa[100];
    int count;
    cout << "How many records you want to enter: ";
    cin >> count;
    for (int idx = 0; idx < count; idx = idx + 1) //For Taking Input
    {
        cout << "Name: ";
        cin >> Names[idx];
        cout << "Roll Number: ";
        cin >> IDs[idx];
        cout << "GPA: ";
        cin >> Gpa[idx];
    }
    cout << "Name" << "\t" << "ID" << "\t" << "GPA" << endl;
    for (int idx = 0; idx < count; idx = idx + 1)   //For Displaying Output
    {
        cout << Names[idx] << "\t" << IDs[idx] << "\t" << Gpa[idx] << endl;
    }
}
```

Method 2

Take input from the user and then **ask** does he/she wants to **enter more records**?

```cpp
#include <iostream>
using namespace std;
main()
{
    string Names[100];
    int IDs[100];
    float Gpa[100];
    int count = 0;
    bool takeInput = true;
    while (takeInput == true) //For Taking Input
    {
        cout << "Name: ";
        cin >> Names[count];
        cout << "Roll Number: ";
        cin >> IDs[count];
        cout << "GPA: ";
        cin >> Gpa[count];
        cout << "If you want to enter another record press 1 otherwise 0: ";
        cin >> takeInput;
        count = count + 1;
    }
}
```

Method 2

Take input from the user and then **ask** does he/she wants to **enter more records**? If he doesn't then display the output.

```cpp
#include <iostream>
using namespace std;
main()
{
    string Names[100];
    int IDs[100];
    float Gpa[100];
    int count = 0;
    bool takeInput = true;
    while (takeInput == true) //For Taking Input
    {
        cout << "Name: ";
        cin >> Names[count];
        cout << "Roll Number: ";
        cin >> IDs[count];
        cout << "GPA: ";
        cin >> Gpa[count];
        cout << "If you want to enter another record press 1 otherwise 0: ";
        cin >> takeInput;
        count = count + 1;
    }
    cout << "Name" << "\t" << "ID" << "\t" << "GPA" << endl;
    for (int idx = 0; idx < count; idx = idx + 1)   //For Displaying Output
    {
        cout << Names[idx] << "\t" << IDs[idx] << "\t" << Gpa[idx] << endl;
    }
}
```

Learning Objective

Declare, initialize and use Parallel arrays to solve real world problems that needs relatively large amount of data.

Conclusion

- To deal with correlated data, Parallel Arrays are used to hold the related information at the same indexes.

Self Assessment

Write a program that helps a maker of Pasta keep track of sales for five different types of Pasta:

Mild, Medium, Sweet, Hot, Zesty

The program should use two parallel 5-element arrays:

- an array of strings that holds the five pasta types
- an array of integers that holds the number of pastas sold during the past month for each pasta type.

The pasta names should be stored using an initialization list at the time the name array is created.

Self Assessment

The program should prompt the user to enter the number of pastas sold for each type.

Once this sales data has been entered, the program should **produce a report that displays**:

- sales for each pasta type
- total sales
- the name of the highest selling product
- the name of the lowest selling products

Input Validation: Do not accept negative values for the number of pastas sold.