Arrays

Scalar Variables

We call simple variable as a scalar variable because it can contain only one value at a time.

```
#include<iostream>
using namespace std;

main(){
    int number = 15;
    float float_data = 6.5;
    string word = "Hello";
    char character = 'A';
}
```

Scalar Variables

We call simple variable as a scalar variable because it can contain only one value at a time.

```
#include<iostream>
using namespace std;

main(){
    int number = 15;
    float float_data = 6.5;
    string word = "Hello";
    char character = 'A';
}
```

Scalar Variables

We call simple variable as a scalar variable because it can contain only one value at a time.

```
#include<iostream>
using namespace std;

main(){
    int number = 15;
    float float_data = 6.5;
    string word = "Hello";
    char character = 'A';
}
```

Scalar Variables

We call simple variable as a scalar variable because it can contain only one value at a time.

```
#include<iostream>
using namespace std;

main(){
    int number = 15;
    float float_data = 6.5;
    string word = "Hello";
    char character = 'A';
}
```

Scalar Variables

We need to store the Mid Term marks of PF for all the students of CS-2021.

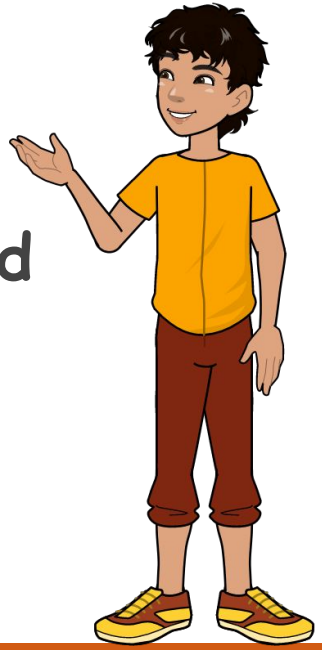
How many Variables we will need?

Scalar Variables

We need to store the Mid Term marks of PF for all the students of CS-2021.

How many Variables we will need?

There are total 220 students enrolled in PF



Scalar Variables

We need to store the Mid Term marks of PF for all the students of CS-2021.

```
#include <iostream>
using namespace std;
main()
{
    float stu1;
    float stu2;
    float stu3;
    .
    .
    .
    float stu220;
}
```

Scalar Variables are Not Sufficient

Therefore, the scalar variables are not sufficient when we need to store a large number of records into the memory.

```
#include <iostream>
using namespace std;
main()
{
    float stu1;
    float stu2;
    float stu3;
    .
    .
    .
    float stu220;
}
```

Scalar Variables are Not Sufficient

To store 220 numbers in RAM, we need to declare 220 variables that is not practically feasible for a programmer.

The requirement can increase from 220 to 1000 or 10000 or even more so it shall get more difficult to declare these.

Alternate Solution

However, we can store multiple values using single variable with the help of Arrays.

Arrays: How to declare?

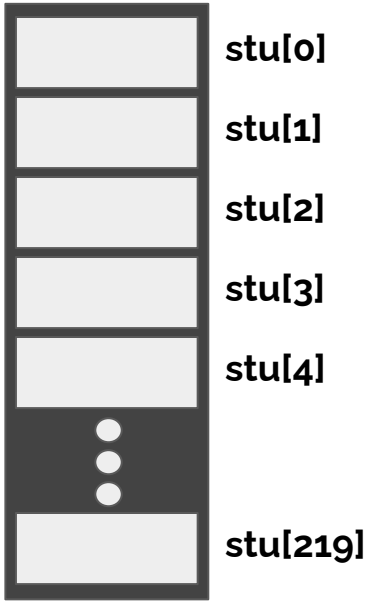
We can declare single variable that can hold 220 values.

```
float stu[220];
```

Arrays: Memory representation

We can declare single variable that can hold 220 values.

```
float stu[220];
```



Arrays:

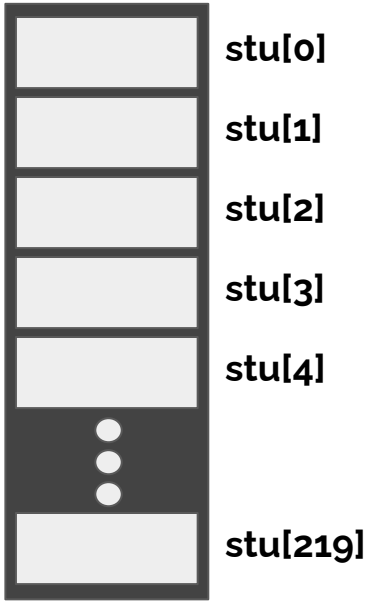
A one-dimensional array is an array in which the components are arranged in a list form.

```
float stu[220];
```

Data Type Of Array

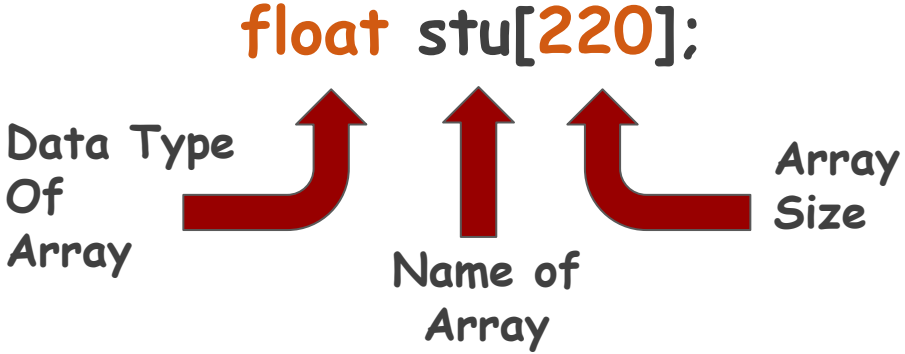
Name of Array

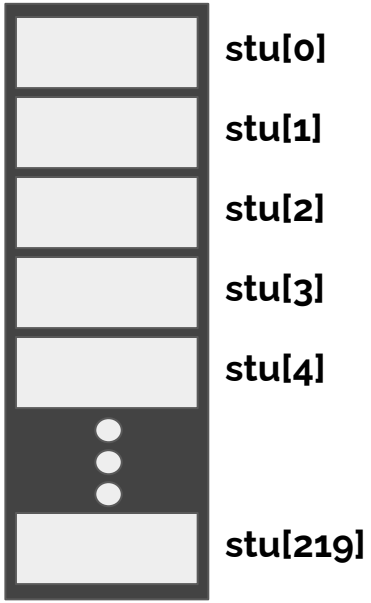
Array Size



Arrays: Index starts at 0

In C++, the array index starts at 0



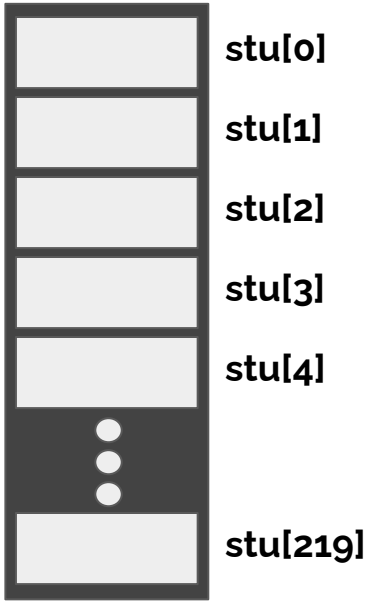


Arrays: How to store values?

```
float stu[220];
```

The Assignment Statement:

```
stu[0] = 19;
```



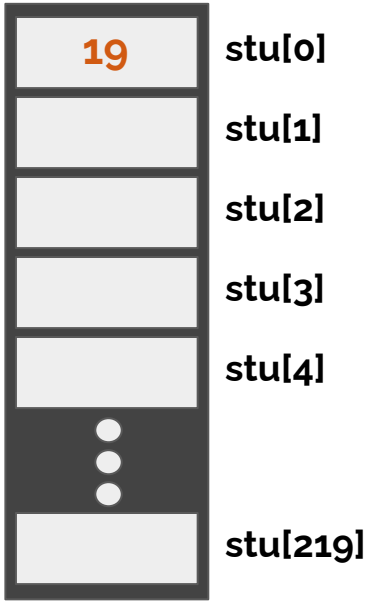
Arrays: How to store values?

```
float stu[220];
```

The Assignment Statement:

```
stu[0] = 19;
```

stores 19 in stu[0], which is the 1st component of the array.



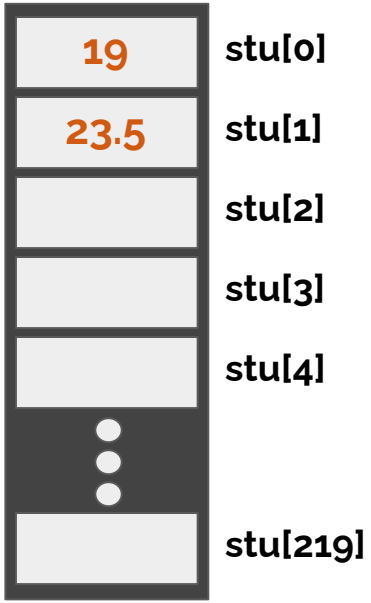
Arrays: How to store values?

```
float stu[220];
```

The Assignment Statement:

```
stu[0] = 19;
stu[1] = 23.5;
```

stores 23.5 in stu[1], which is the 2nd component of the array.



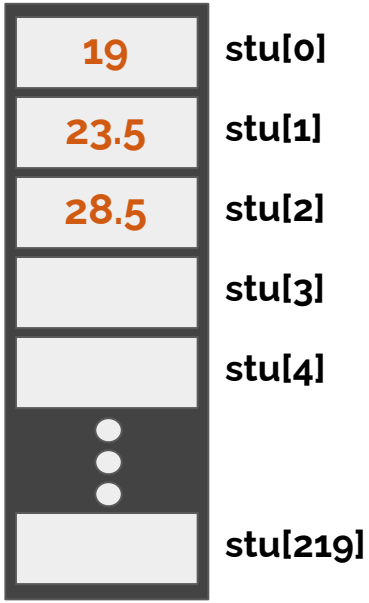
Arrays: How to store values?

```
float stu[220];
```

The Assignment Statement:

```
stu[0] = 19;
stu[1] = 23.5;
stu[2] = 28.5;
```

stores 28.5 in stu[2], which is the 3rd component of the array.

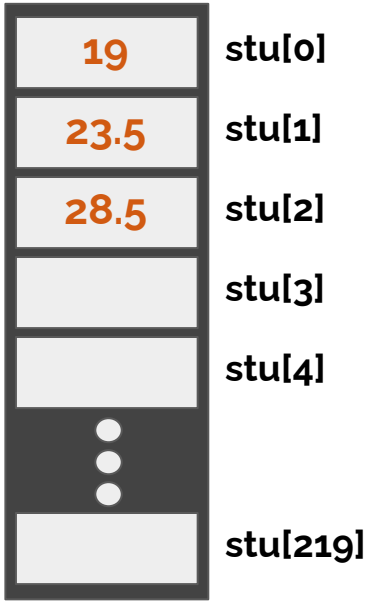


Arrays: How to access values?

```
float stu[220];
```

Just like variables, array elements are accessed in the same manner.

```
cout << stu[0];
```



Arrays: How to access values?

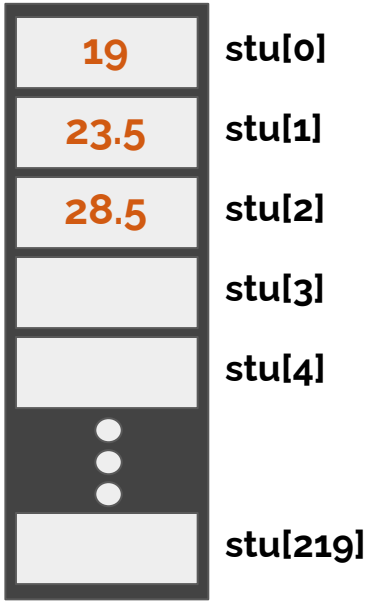
```
float stu[220];
```

Just like variables, array elements are accessed in the same manner.

```
cout << stu[0];
```

```
C:\C++>c++ program.cpp -o program.exe

C:\C++>program.exe
19
C:\C++>
```

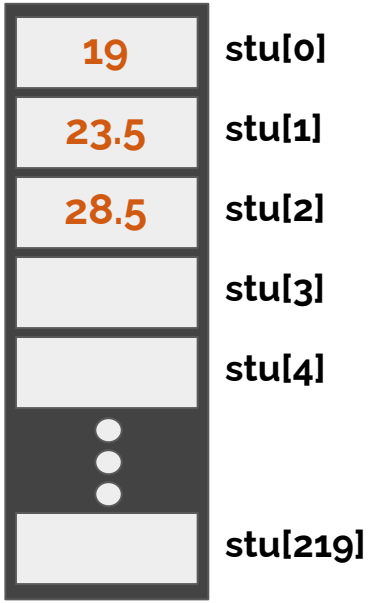


Arrays: How to access values?

```
float stu[220];
```

Just like variables, array elements are accessed in the same manner.

```
cout << stu[1];
```



Arrays: How to access values?

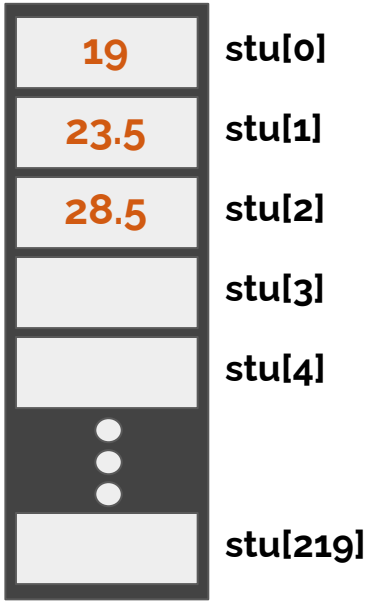
```
float stu[220];
```

Just like variables, array elements are accessed in the same manner.

```
cout << stu[1];
```

```
C:\C++>c++ program.cpp -o program.exe

C:\C++>program.exe
23.5
C:\C++>
```

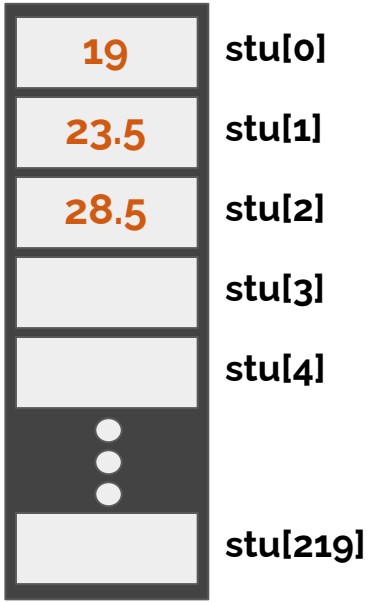


Arrays: Take input from the user

```
float stu[220];
```

Just like variables, arrays can also be initialized by the user in the following manner.

```
cout << "Enter Value: "
cin >> stu[3];
```



Arrays: Take input from the user

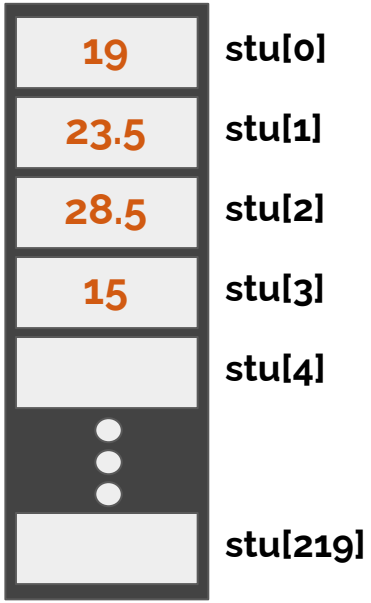
```
float stu[220];
```

Just like variables, arrays can also be initialized by the user in the following manner.

```
cout << "Enter Value: "
cin >> stu[3];
```

```
C:\C++>c++ program.cpp -o program.exe

C:\C++>program.exe
Enter Value: 15
```

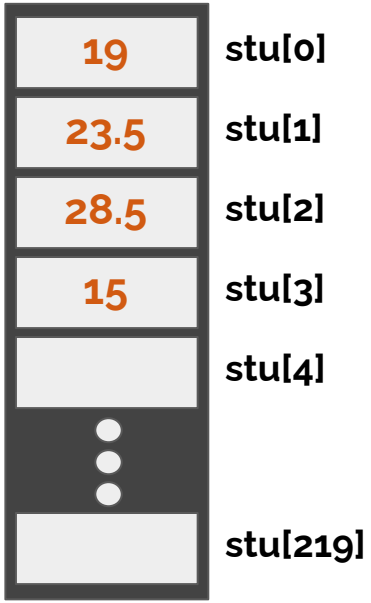


Arrays: Take input from the user

```
float stu[220];
```

Just like variables, arrays can also be initialized by the user in the following manner.

```
cout << "Enter Value: "
cin >> stu[4];
```



Arrays: Take input from the user

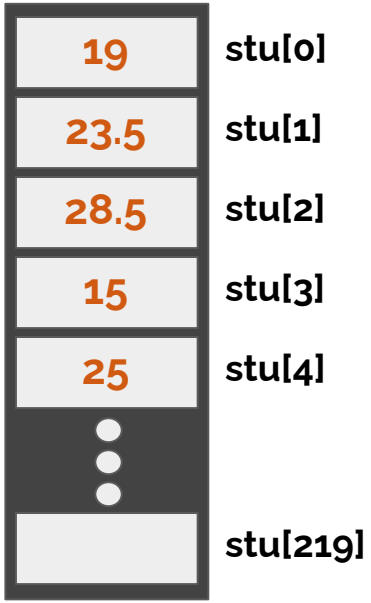
```
float stu[220];
```

Just like variables, arrays can also be initialized by the user in the following manner.

```
cout << "Enter Value: "
cin >> stu[4];
```

```
C:\C++>c++ program.cpp -o program.exe

C:\C++>program.exe
Enter Value: 25
```



Arrays:

Just like variables, arrays can also be initialized by the user in the following manner.

```
#include <iostream>
using namespace std;
main()
{
    float stu[220];
    cout << "Enter 1st student Marks: ";
    cin >> stu[0];
    cout << "Enter 2nd student Marks: ";
    cin >> stu[1];
    cout << "Enter 3rd student Marks: ";
    cin >> stu[2];
    .
    .
    .
    cout << "Enter 220th student Marks: ";
    cin >> stu[219];
}
```

Arrays:

Name of the array remains the same, only the indexes values change.

```
#include <iostream>
using namespace std;
main()
{
    float stu[220];
    cout << "Enter 1st student Marks: ";
    cin >> stu[0];
    cout << "Enter 2nd student Marks: ";
    cin >> stu[1];
    cout << "Enter 3rd student Marks: ";
    cin >> stu[2];
    .
    .
    .
    cout << "Enter 220th student Marks: ";
    cin >> stu[219];
}
```

Arrays:

We are accessing each index of the array separately.

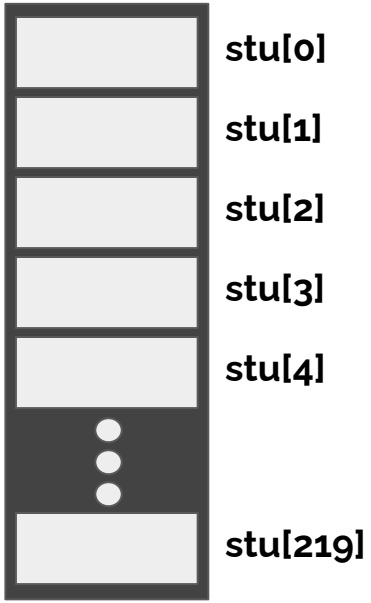
Now, the question is:

What is the benefit for us if we have to repeat the code for 200 times?

```
#include <iostream>
using namespace std;
main()
{
    float stu[220];
    cout << "Enter 1st student Marks: ";
    cin >> stu[0];
    cout << "Enter 2nd student Marks: ";
    cin >> stu[1];
    cout << "Enter 3rd student Marks: ";
    cin >> stu[2];
    .
    .
    .
    cout << "Enter 220th student Marks: ";
    cin >> stu[219];
}
```

Arrays: Make the input more dynamic

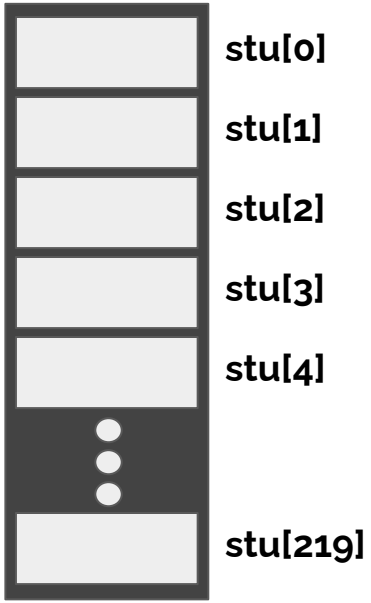
We can use loops to take input from the user more dynamically.



Arrays: Make the input more dynamic

We can use loops to take input from the user more dynamically.

```
#include <iostream>
using namespace std;
main(){
    int arr_size = 220;
    int stu[arr_size];
    for (int x = 0; x < arr_size; x = x + 1)
    {
        cout << "Enter " << x+1 << " element: ";
        cin >> stu[x];
    }
}
```



Scalar Variables VS Arrays

Scalar Variables

```
#include <iostream>
using namespace std;
main()
{
    float stu1, stu2, stu3, ... stu220;
    cin >> stu1;
    cin >> stu2;
    .
    .
    .
    cin >> stu220;
}
```

Arrays

```
#include <iostream>
using namespace std;
main(){
    int arr_size = 220;
    int stu[arr_size];
    for (int x = 0; x < arr_size; x = x + 1)
    {
        cout << "Enter " << x+1 << " element: ";
        cin >> stu[x];
    }
}
```

Arrays: Benefit

With 1 variable we can access many memory locations.

Scalar Variables

```
#include <iostream>
using namespace std;
main()
{
    float stu1, stu2, stu3, … stu220;
    cin >> stu1;
    cin >> stu2;
    .
    .
    .
    cin >> stu220;
}
```

Arrays

```
#include <iostream>
using namespace std;
main(){
    int arr_size = 220;
    int stu[arr_size];
    for (int x = 0; x < arr_size; x = x + 1)
    {
        cout << "Enter " << x+1 << " element: ";
        cin >> stu[x];
    }
}
```

Arrays: Benefit

We just have to vary the indexes of the array.

Scalar Variables

```
#include <iostream>
using namespace std;
main()
{
    float stu1, stu2, stu3, … stu220;
    cin >> stu1;
    cin >> stu2;
    .
    .
    .
    cin >> stu220;
}
```

Arrays

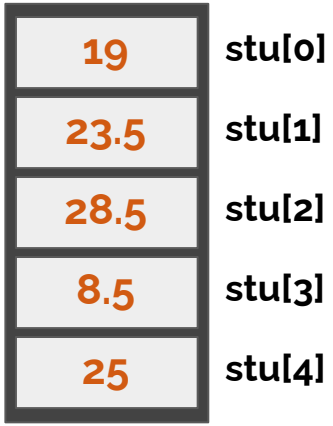
```
#include <iostream>
using namespace std;
main(){
    int arr_size = 220;
    int stu[arr_size];
    for (int x = 0; x < arr_size; x = x + 1)
    {
        cout << "Enter " << x+1 << " element: ";
        cin >> stu[x];
    }
}
```

Store values in Arrays: 1st Method

```
float stu[5];
```

Take input from user:

```
cin >> stu[0];
cin >> stu[1];
cin >> stu[2];
cin >> stu[3];
cin >> stu[4];
```



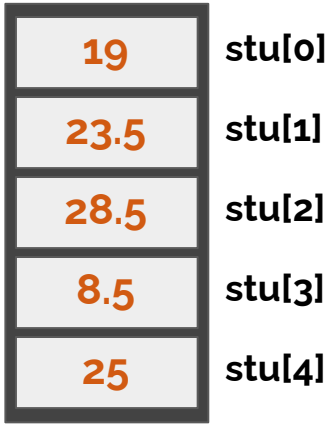
Here, we have to store each value separately.

Store values in Arrays: 2nd Method

```
float stu[5];
```

The Assignment Statement:

```
stu[0] = 19;
stu[1] = 23.5;
stu[2] = 28.5;
stu[3] = 8.5;
stu[4] = 25;
```

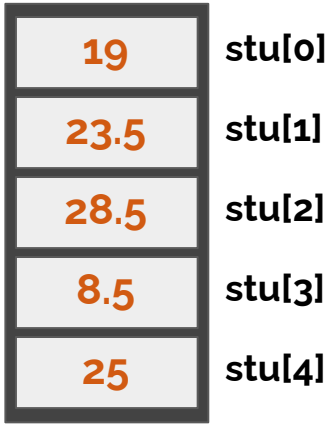


Here, we have to store each value separately.

Store values in Arrays: 3rd Method

An alternate way to initialize the complete array is:

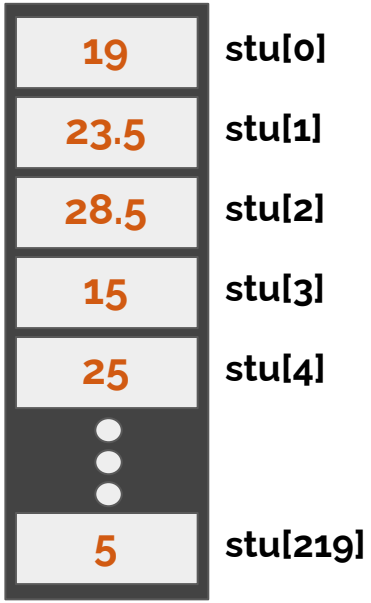
```
float stu[5] = {19, 23.5, 28.5, 8.5, 25};
```



Arrays: Processing with loop

Now we have data in the stu[220] float array, we want to print Passed or Failed based on student performance.
Condition is: If the marks are greater than 15 print marks and write Passed Otherwise Failed.

How to do it ?



Arrays:

```
#include <iostream>
using namespace std;
string result(float number)
{
    string grade;
    if (number > 15){
        grade = "Passed";
    }
    else{
        grade = "Failed";
    }
    return grade;
}
main(){
    float stu[220];
    for(int x = 0; x < 220; x = x+1){
        cin >> stu[x];
    }
    for(int idx = 0; idx < 220; idx = idx + 1){
        string  grade = result(stu[idx]);
        cout << stu[idx] << ": " << grade << endl;
    }
}
```

Scalar Variables: Working Example

Let's suppose three variables are given and the requirement is to find whether the number entered by the user is present in any of the variables or not.

```
n1 = 34
n2 = 98
n3 = 45
```

Scalar Variables: Solution

```
#include <iostream>
using namespace std;
main(){
    int n1 = 34, n2 = 98, n3 = 45, findNum;
    cout << "Please Enter the Number: ";
    cin >> findNum;
    if (findNum == n1 || findNum == n2 || findNum == n3){
        cout << "The number is present in the variable" << endl;
    }
    else{
        cout << "The number is not present in the variable" << endl;
    }
}
```

Arrays: Working Example

Let's suppose three numbers are given in array and the requirement is to find whether the number entered by the user is present in array or not.

```
int n[3] = {34, 98, 45};
```

Arrays: Solution

```
#include <iostream>
using namespace std;
main()
{
    int n[3] = {34, 98, 45}, findNum;
    cout << "Please Enter the Number: ";
    cin >> findNum;
    for (int idx = 0; idx < 3; idx = idx + 1){
        if (n[idx] == findNum){
            cout << "The number is present in the array" << endl;
        }
        else{
            cout << "The number is not present in the array" << endl;
        }
    }
}
```

Arrays: Solution

```
int n[3] = {34, 98, 45};
```

```
C:\C++>c++ program.cpp -o program.exe

C:\C++>program.exe
Please Enter the Number: 34
The number is present in the array
The number is not present in the array
The number is not present in the array

C:\C++>
```

Arrays

```
#include <iostream>
using namespace std;
main(){
    int n[3] = {34, 98, 45};
    int findNum;
    bool isFound = false;
    cout << "Please Enter the Number: ";
    cin >> findNum;
    for (int idx = 0; idx < 3; idx++)    // For accessing values
    {
        if (n[idx] == findNum){
            isFound = true;
        }
    }
    if(isFound == true){
        cout << "The number is present in the array" << endl;
    }
    else{
        cout << "The number is not present in the array" << endl;
    }
}
```

Arrays: Output

```
C:\C++>c++ example.cpp -o example.exe

C:\C++>example.exe
Please Enter the Number: 34
The value is present in the array

C:\C++>
```

```
C:\C++>c++ example.cpp -o example.exe

C:\C++>example.exe
Please Enter the Number: 66
The value is not present in the array

C:\C++>
```

Arrays: Working Example

Let's suppose numbers are given in array and the requirement is to find whether the number entered by the user is present in array or not.

Now extent this program for 100 numbers.

Learning Objective

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

Conclusion

- The general form of declaring an array is:

dataType arrayName[number]

- The general form (syntax) used for accessing an array component is:

arrayName[number]

the number is called the index which is any expression whose value is a nonnegative integer.

- in C++, the array index starts at 0.

Self Assessment

1. Write a program that takes 5 numbers in the array from user and find the largest of these numbers.

2. Suppose a Cinema displays 5 movies. Price of Each movie ticket is 500. Write a program in which you have 5 movie names stored in the array.
   For Example

```
string movies[5] = {Gladiator, StarWars, Terminator, TakingLives, TombRider};
```

   Take 1 movie name as input from the user and if the movie is stored on an odd index of the array then give 5% discount on the movie ticket otherwise give 10% discount.