

Programming in C++
Prof. Partha Pratim Das
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 05
Arrays and Strings

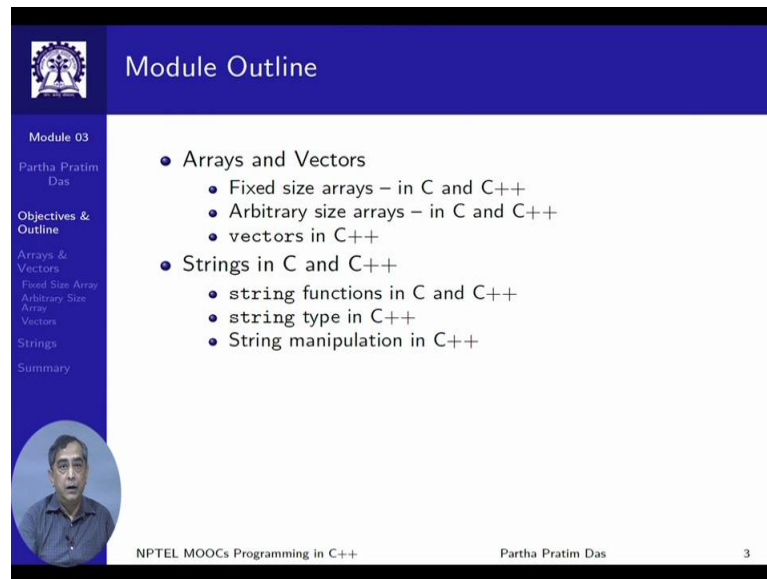
Welcome to module 3 of Programming in C++. This module we will discuss about arrays and strings. We have in module 2 seen what are the basic differences between a C program and a C++ program. We will continue on the same note.

(Refer Slide Time: 00:41)

The slide is titled "Module Objectives" and features a dark blue header with the IIT Kharagpur logo on the left. A sidebar on the left contains the following text: "Module 03", "Partha Pratim Das", "Objectives & Outline", "Arrays & Vectors", "Fixed Size Array", "Arbitrary Size Array", "Vectors", "Strings", and "Summary". Below the sidebar is a circular portrait of Prof. Partha Pratim Das. The main content area lists three bullet points: "• Understand array usage in C and C++", "• Understand vector usage in C++", and "• Understand string functions in C and string type in C++". The footer of the slide includes "NPTEL MOOCs Programming in C++", "Partha Pratim Das", and the number "2".

In this module we will try to particularly understand the use of arrays in C and C++. We will introduce a basic notion of what is called vector in C++, which is pretty much like arrays, but lot more powerful and we will try to see how strings are used in C and in contrast how the string type operates in C++.

(Refer Slide Time: 01:06)



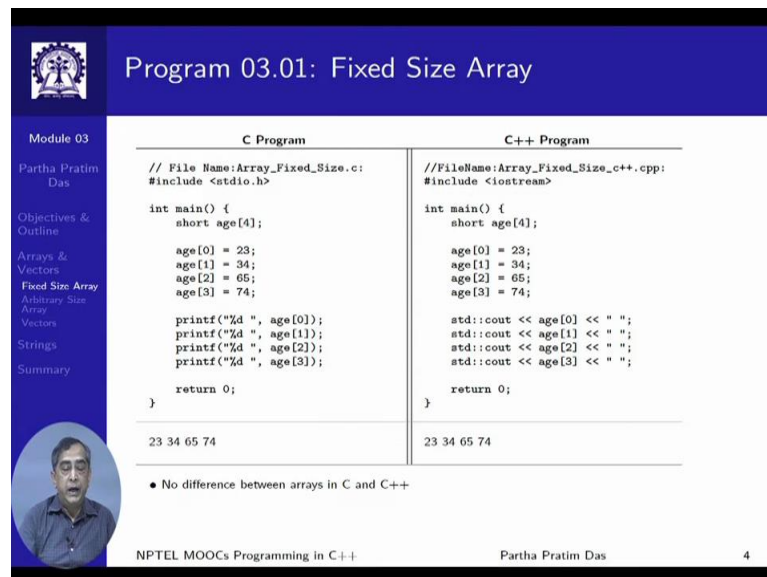
The slide is titled "Module Outline" and is part of "Module 03" by Partha Pratim Das. It lists the following topics:

- Arrays and Vectors
 - Fixed size arrays – in C and C++
 - Arbitrary size arrays – in C and C++
 - vectors in C++
- Strings in C and C++
 - string functions in C and C++
 - string type in C++
 - String manipulation in C++

The slide footer includes "NPTEL MOOCs Programming in C++", "Partha Pratim Das", and the number "3".

So, these are the points that we will cover.

(Refer Slide Time: 01:11)



The slide is titled "Program 03.01: Fixed Size Array" and compares C and C++ code for a fixed-size array. The C program uses `printf` and the C++ program uses `std::cout`. Both programs declare a `short age[4]` array with values 23, 34, 65, and 74. The output for both is "23 34 65 74".

C Program	C++ Program
<pre>// File Name:Array_Fixed_Size.c: #include <stdio.h> int main() { short age[4]; age[0] = 23; age[1] = 34; age[2] = 65; age[3] = 74; printf("%d ", age[0]); printf("%d ", age[1]); printf("%d ", age[2]); printf("%d ", age[3]); return 0; }</pre>	<pre>//FileName:Array_Fixed_Size_c++.cpp: #include <iostream> int main() { short age[4]; age[0] = 23; age[1] = 34; age[2] = 65; age[3] = 74; std::cout << age[0] << " "; std::cout << age[1] << " "; std::cout << age[2] << " "; std::cout << age[3] << " "; return 0; }</pre>
23 34 65 74	23 34 65 74

• No difference between arrays in C and C++

The slide footer includes "NPTEL MOOCs Programming in C++", "Partha Pratim Das", and the number "4".

To get started we have side by side shown two programs, both of them. They actually are identical except for the difference in the IO header. So, a basic use of array as in C can be done in C++ exactly in the same notation and with the same meaning. So, the first

message is arrays can be used in C++ exactly as you know as in C. So, here we are just assigning some values to the different array elements and printing them, only difference is in terms the using the printf or using the cout.

(Refer Slide Time: 01:56)

Arbitrary Size Array

Module 03
Partha Pratim Das
Objectives & Outline
Arrays & Vectors
Fixed Size Array
Arbitrary Size Array
Vectors
Strings
Summary

This can be implemented in C (C++) in the following ways:

- **Case 1:** Declaring a large array with size greater than the size given by users in all (most) of the cases
 - Hard-code the maximum size
 - Declare a manifest constant for the maximum size
- **Case 2:** Using `malloc` (`new[]`) to dynamically allocate space at run-time for the array

NPTEL MOOCs Programming in C++ Partha Pratim Das 5

Now, the one of the main issues in C, in terms of using arrays that you all must have faced is when I want to use an array in C, I need to know; I need to specify the size of the array which means at the maximum number of elements that the array can contain beforehand, that is at the time of writing the program or to be specific at the time of compiling the program. So, if I do not know that size, then I need to provide a size which is greater than what can happen at in any one of the cases that I execute, run in the program.

Certainly there are two ways of handling this situation. One is I define, declare a large array and this can be done; this can be either hard coded, the size can be hard coded within the program or the size can be somewhat soft coded by using a manifest constant and the other case that you have seen earlier in C programming is you would use malloc to dynamically allocate space and allocate the array at the run time and then you use it and if you are using malloc, you will also have to remember to free it up, when you are use of that array is done.

(Refer Slide Time: 03:21)

The slide is titled "Program 03.02: Fixed size large array in C". It is divided into two columns: "Hard-coded" and "Using manifest constant".

Hard-coded:

```
// FileName:Array_Large_Size.c:
#include <stdio.h>
#include <stdlib.h>

int main() {
    int arr[100], sum = 0, i;
    int count;
    printf("Enter no. of elements: ");
    scanf("%d", &count);
    for(i = 0; i < count; i++) {
        arr[i] = i;
        sum += arr[i];
    }
    printf("Array Sum: %d", sum);
    return 0;
}
```

Enter no. of elements: 10
Array Sum: 45

• Hard-coded size

Using manifest constant:

```
// FileName:Array_Macro.c:
#include <stdio.h>
#include <stdlib.h>
#define MAX 100

int main() {
    int arr[MAX], sum = 0, i;
    int count;
    printf("Enter no. of elements: ");
    scanf("%d", &count);
    for(i = 0; i < count; i++) {
        arr[i] = i;
        sum += arr[i];
    }
    printf("Array Sum: %d", sum);
    return 0;
}
```

Enter no. of elements: 10
Array Sum: 45

• Size by manifest constant

NPTEL MOOCs Programming in C++ Partha Pratim Das 6

So, let us see, how does this look in terms of as we migrate from C to C++. So, this is just showing the C example on the left hand side. We are hard coding the size of the array arr to 100 on the right column we are doing the same thing, except that now you have a manifest constant math, which is defined to have a value 100 and we are using that math. The advantage of using the manifest constant is there could be 10 different arrays, whose size is to be specified to 100.

Now, if I hard code all of them and sometime later I need to change all those sizes from 100 to 1000, then at 10 places I will need to go and edit and I might just forget to do all of them, but if I hash define or use a manifest constant then I can make the change only at one place, change the definition of max 100 to 1000 and all of those will change. So, it is a better programming practice in C to use manifest constant and not hard code value, you already know this.

(Refer Slide Time: 04:35)

Program 03.03: Fixed large array / vector

C (array & constant)	C++ (vector & constant)
<pre>// FileName:Array_Macro.c: #include <stdio.h> #include <stdlib.h> #define MAX 100 int main() { int arr[MAX], sum = 0, i; int count; printf("Enter no. of elements: "); scanf("%d", &count); for(i = 0; i < count; i++) { arr[i] = i; sum += arr[i]; } printf("Array Sum: %d", sum); return 0; }</pre>	<pre>// FileName:Array_Macro_c++.cpp: #include <iostream> #include <vector> using namespace std; #define MAX 100 int main() { vector<int> arr(MAX); // Define-time size cout <<"Enter the no. of elements: "; int count, j, sum = 0; cin >>count; for(int i = 0; i < count; i++) { arr[i] = i; sum += arr[i]; } cout << "Array Sum: " << sum << endl; return 0; }</pre>
<pre>Enter no. of elements: 10 Array Sum: 45</pre>	<pre>Enter no. of elements: 10 Array Sum: 45</pre>
<ul style="list-style-type: none">• MAX is the declared size of array• No header needed• arr declared as int []	<ul style="list-style-type: none">• MAX is the declared size of vector• Header vector included• arr declared as vector<int>

NPTEL MOOCs Programming in C++ Partha Pratim Das 7

Now, you show the similar situation in between C and C++ on the right column now, you have a C++ program. Certainly, the array size, the maximum array size as max can be hash define to 100 as has been done, but what we show here is just focus on the line right after the header of main. We are writing vector within corner bracket int and then the array name and within parentheses, we show the size max.


Vector is a new introduction in C++. This is not a built-in time; please do not consider this to be a built-in time. This is something which is provided by the C++ standard library. So, if you move your attention to the top in terms of the hash include list, you will find hash, there is hash include vector. So, there is standard library header vector which has all the necessary declaration definitions for a vector type and you can use that in this way; what it means is vector for in all respect what likes arrays can be. So, you just focus within the 'for' loop, you see how the array elements are being accessed.

On the left hand side, it is a well known array int arr. We write it as arr i on the right, it is a vector of int we use the same indexing notation to access the array elements. So, vector is same in terms of access notation and the result of doing the read or write access with the traditional array, but it has the advantage that its size is not necessarily fixed at the compile time.

Now, in this example we have just shown that how to use vector with a fixed initial size max. So, when we say vector and within corner bracket int, what we mean that within corner bracket, we are providing the type of the element that the array is composed of, which is what we write in C as int arr we write it as vector within corner bracket int and whatever we provide as the maximum size within the square brackets here, we pass it as a parameter after the arr name.

So, this is just a notational difference, right now just accept this as a different notation of writing, declaring arrays and once you have done that rest of the program you can forget about, that you are specifically using a vector you can just continue to use them as arrays.

(Refer Slide Time: 07:42)



Program 03.04: Dynamically managed array size

Module 03

Partha Pratim Das

Objectives & Outline

Arrays & Vectors

Fixed Size Array


Arbitrary Size Array

Vectors

Strings

Summary

C Program	C++ Program
<pre style="font-family: monospace; font-size: 0.8em; margin: 0;">// FileName:Array_Malloc.c: #include <stdio.h> #include <stdlib.h> int main() { printf("Enter no. of elements "); int count, sum = 0, i; scanf("%d", &count); int *arr = (int*) malloc (sizeof(int)*count); for(i = 0; i < count; i++) { arr[i] = i; sum += arr[i]; } printf("Array Sum:%d ", sum); return 0; }</pre>	<pre style="font-family: monospace; font-size: 0.8em; margin: 0;">// FileName:Array_Resize_c++.cpp: #include <iostream> #include <vector> using namespace std; int main() { cout << "Enter the no. of elements: "; int count, j, sum=0; cin >> count; vector<int> arr; // Default size arr.resize(count); // Set resize for(int i = 0; i < arr.size(); i++) { arr[i] = i; sum += arr[i]; } cout << "Array Sum: " << sum << endl; return 0; }</pre>
<pre style="font-family: monospace; font-size: 0.8em; margin: 0;">Enter no. of elements: 10 Array Sum: 45</pre>	<pre style="font-family: monospace; font-size: 0.8em; margin: 0;">Enter no. of elements: 10 Array Sum: 45</pre>
<ul style="list-style-type: none"> • malloc allocates space using sizeof 	<ul style="list-style-type: none"> • resize fixes vector size at run-time



NPTEL MOOCs Programming in C++

Partha Pratim Das

8

Now with this, let me show where you actually get the advantage. Now, let us focus on the second mechanism of using arbitrary sized arrays that is you do not know at all as to how large an array can be you will get to know only when the program is executed by the user. So, the user will probably provide the size of the array, the number of elements that the user wants.

So, on the C program, on the left see that the first we are asking the user; how many

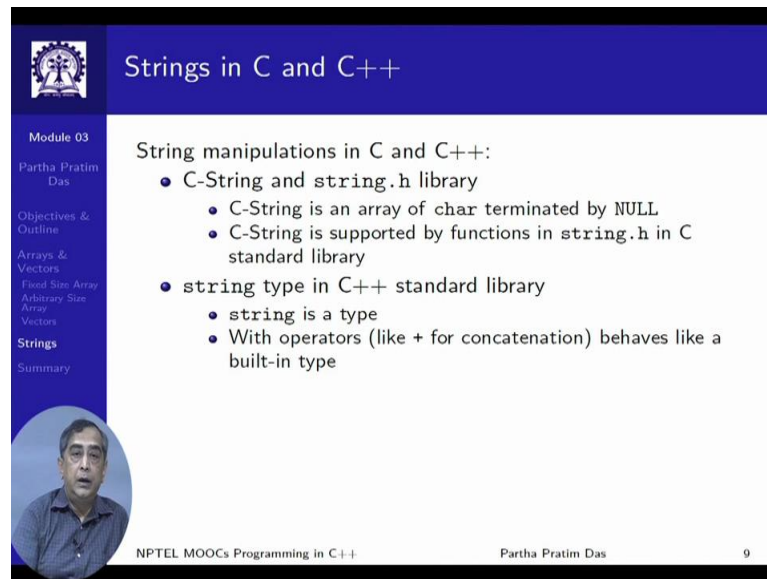
elements are there, and the user provides a count value and if you have to do this in C, then will have to dynamically allocate the array using malloc as is shown and proceed with that. Certainly you will need to write a very complex font in malloc because you really need to say how much memory you want, malloc returns you a void star pointer, you need to remember and cast that to the int star.

All those nuances of C programming exist; now just shift your focus to the right on the same lines. Now, we are declaring the variable as a vector of int the variable is arr and please note in contrast to what we had shown earlier, we are not passing any max size. So, if we do not pass a max size then we get a vector of a default sizes, C++ standard library will have some default size which is not fixed, but some default size array will be there, but the following line we write something which you are not familiar in the notation. We are writing arr dot resize, we call this as resize being a member function of the vector type. What it does is in this resize function if we pass a value as we are passing through the variable count, then the vector will resize to the count number of elements.

So, let us assume that the default size with which the vector was created is 10 and now the user at the run time as given an input 100 to count, when arr dot resize is done the value will be passed as 100 and the vector will change to having hundred elements from the original ten elements.

So, resize can be used very conveniently to increase or decrease the number of elements that a vector can have or for that matter the vector form of array can have. So, with that you get rid of using all this malloc and is complicated notation and remembering to free that location and so on. We can just use vector and resize them as needed which makes the use of arrays in C++ programs as vector container is far more convenient and compact than the similar mechanism in C.

(Refer Slide Time: 10:59)



The slide is titled "Strings in C and C++" and is part of "Module 03" by Partha Pratim Das. It lists topics such as "Objectives & Outline", "Arrays & Vectors", "Fixed Size Array", "Arbitrary Size Array", "Vectors", "Strings", and "Summary". A circular portrait of the instructor is visible in the bottom left corner. The main content is a bulleted list under the heading "String manipulations in C and C++:".

- C-String and `string.h` library
 - C-String is an array of `char` terminated by `NULL`
 - C-String is supported by functions in `string.h` in C standard library
- `string` type in C++ standard library
 - `string` is a type
 - With operators (like `+` for concatenation) behaves like a built-in type

NPTEL MOOCs Programming in C++ Partha Pratim Das 9

Next, let us take a look in the handling of strings as you are; would be already familiar that besides 2 numerical types that is whole numbers `int` and the floating point numbers, the next most widely used and most required type or values that we need to deal with our strings where we are talking about a sequence of characters, and what do we have in; if we are working in C, we have what is now called a C string.

C does not have a default type as `string`, but it as a `string.h` standard library header which provides a whole lot of string functions like `strlen`, `strcpy`, `strcat` and so on and with that C string is just an array of characters, which we say is terminated by `NULL`, which means that if you scan the array from left to right, you will continue to consider that you have a string till you come across the first `NULL` characters or the characters with ASCII value 0; please note that in the array after this `NULL` there could be several other characters still remaining, but they are not considered to be part of the string.

Now, with this convention if we use the string functions from `string.h` `string` dot `h` header then you will be able to achieve variety of string operations as you all are familiar with. In contrast, C++ now introduces a `string` type in C++ standard library. This is pretty much like we talked about `vector`. So, `string` also is not a built-in type, but it is a type added through the standard library and you will have to use the `string` header of C++

standard library to get the strings and it has some amazing behavior like being able to write concatenations of string as an addition expression.

(Refer Slide Time: 13:06)

The slide is titled "Program 03.05: Concatenation of Strings". It is divided into two columns: "C Program" and "C++ Program".

C Program:

```
// FileName: Add_strings.c:
#include <stdio.h>
#include <string.h>

int main() {
    char str1[] =
        {'H', 'E', 'L', 'L', 'O', ' ', '\0'};
    char str2[] = "WORLD";

    char str[20];
    strcpy(str, str1);
    strcat(str, str2);

    printf("%s\n", str);

    return 0;
}
```

C++ Program:

```
// FileName: Add_strings_++.cpp:
#include <iostream>
#include <string>
using namespace std;

int main(void) {
    string str1 = "HELLO ";
    string str2 = "WORLD";

    string str = str1 + str2;

    cout << str;

    return 0;
}
```

Both programs output "HELLO WORLD".

Notes for C Program:

- Need header string.h
- C-String is an array of characters
- String concatenation done with strcat function
- Need a copy into str
- str must be large to fit the result

Notes for C++ Program:

- Need header string
- string is a data-type in C++ standard library
- Strings are concatenated like addition of int.

NPTEL MOOCs Programming in C++ Partha Pratim Das 10

So, we will illustrate those. Here is a simple parallel between a C program and a C++ program. This program starts with two strings that are defined within the program the hello world and we want to concatenate the second string after the first string. So, we just want to put them side by side the first string followed by the second string and make one concatenated string.

So, if you have to do that in C, on the left you can see what you will need to do you will need to have an array large enough to contain the concatenated string let us called it str, you will have to copy the first string str 1 into str and then you will have to concatenate str 2 into what is already covered in str. So, it will just come after that, so first hello will hello followed by a blank will get copied to str and then world will get concatenated str copy and str cat does the job and then you can print it.

In contrast, in C++ you have a string type in the string header. So, you include the string header now you do not declare them as characters arrays you declare them as string which is the name of the type given in that header; please note that this name is all in

lower case and then you have the string variable, name str 1 and you initialize it constant string hello blank or world.

The very interesting thing is when you have to concatenate it you do not really need to copy the first string and then do concatenation, you can just say that I am adding str 2 to str 1. So, we say str 1 plus str 2, so this is pretty much like I have a variable x having value 3, have a variable y having value 5. I write x plus y to mean 3 plus 5, which is 8. So that is an integer addition.

This is kind of a string addition in the type of string, this becomes a concatenation operation and we will see the amazing power in C++ to be able to define operators for your own types in whatever way you want to interpret them. For example, you could use this to write algebra for rectangles you can have two rectangles, if you have a rectangle type and you can define then the addition of two rectangles is basically making a union of these rectangles to make a rectangle large enough to contain both these rectangles and so on. So, this feature in terms of string is available in C++ therefore, it becomes really easy to deal with strings.

In C++ particularly note that in C, you will really need to know what is the size of the result? So, that you can define again an array large enough for the variable str because if this str is not enough in size then str copy, str cat later on will fail in C++ you do not need to bother about any of this when you do when you declare the variable str as a type string and you initialize it with the concatenation of str one plus str two the compiler automatically takes care of managing the size and will give you a string which is large enough to contain the concatenation. So, there is a lot of ease in the whole handling of strings.

(Refer Slide Time: 17:00)

More on Strings

Module 03
Partha Pratim Das

Objectives & Outline
Arrays & Vectors
Fixed Size Array
Arbitrary Size Array
Vectors
Strings
Summary

Further,

- `operator=` can be used on strings in place of `strcpy` function in C.
- `operator<=`, `operator<`, `operator>=`, `operator>` operators can be used on strings in place of `strcmp` function in C

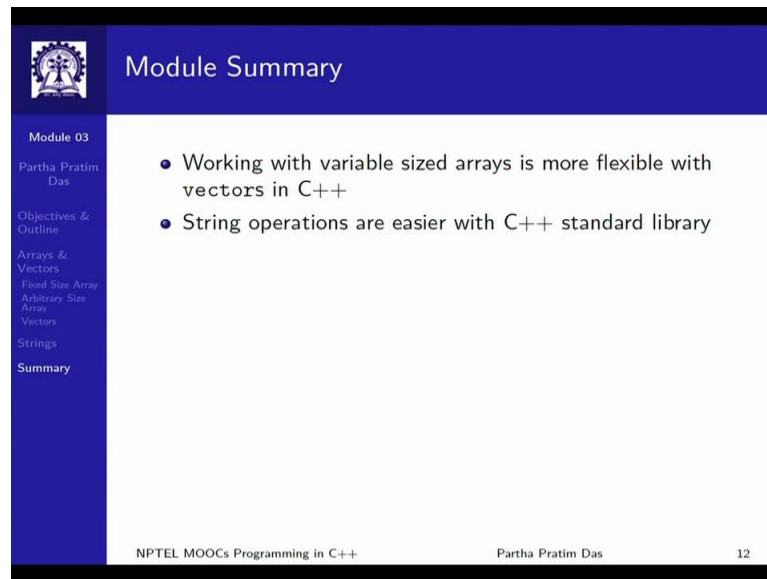
NPTEL MOOCs Programming in C++ Partha Pratim Das 11

Further, actually does not just end with adding strings or use the addition operators for concatenation of strings, you can do several other operations.

In fact, you do not actually need any of the string dot h functions that you have in the C standard library and achieve their task by using the more natural operators like you can use an assignment in place of doing string copy, you can use all the comparison operators less than equal to less than greater than equal to greater than in place of using `strcmp` `cmp`. We know `strcmp` is a relatively complex function to use because it can take; it takes two strings; two C strings that is card stack pointers and returns you a value which could either be minus 1 or be 0 or be plus 1 depending on which string is larger or equal if the strings are equal and so on.

Now, you do not with string type in C++ you do not need to get into any of this. You can just use the comparison operators and compare strings much in the same way you compare integers or floating point numbers. So, this is a very strong feature of C++ and particularly for string, this is an extremely convenient way. So, even without getting into deep understanding of C++ you could just start using string and make all your programs smarter and easier to write.

(Refer Slide Time: 18:34)



The slide is titled "Module Summary" and features a blue header bar. On the left side, there is a vertical navigation menu with a logo at the top. The menu items are: "Module 03", "Partha Pratim Das", "Objectives & Outline", "Arrays & Vectors", "Fixed Size Array", "Arbitrary Size Array", "Vectors", "Strings", and "Summary". The main content area is white and contains two bullet points. At the bottom, there is a footer with the text "NPTEL MOOCs Programming in C++", "Partha Pratim Das", and the page number "12".

Module Summary

- Working with variable sized arrays is more flexible with vectors in C++
- String operations are easier with C++ standard library

NPTEL MOOCs Programming in C++ Partha Pratim Das 12

In this module we have shown - how we can work with arrays, how vector really makes it easier to make arrays variable sized and how strings operations can be done very easily in C++ using the string type from the standard library.