CS4002
Applied Programming

National University of Computer & Emerging Sciences
Islamabad, Pakistan

Ms. Umarah Qaseem

1

Content

- Repetition Control Structures
- While Loop
- Counter-Controlled `While`Loops
- Sentinel-Controlled `While`Loops
- Flag-Controlled `While`Loops
- For Loop
- Do while
- Break continue
- Nested Loops
- Functions (Built-in, User Defined)
- Arrays 1D

2

Repetition Control Structures

3

3

Sum of 5 Numbers

```
#include <iostream>
namespace std

int main() {

        int num1, num2, num3, num4, num5;
        // Input 5 numbers individually
        cin >> num1;
        cin >> num2;
        cin >> num3;
        cin >> num4;
        cin >> num5;
        // Sum the numbers
        int sum = num1 + num2 + num3 + num4 + num5;

        cout << "The sum of the 5 numbers is: " << sum << endl;

return 0;
}
```

4

4

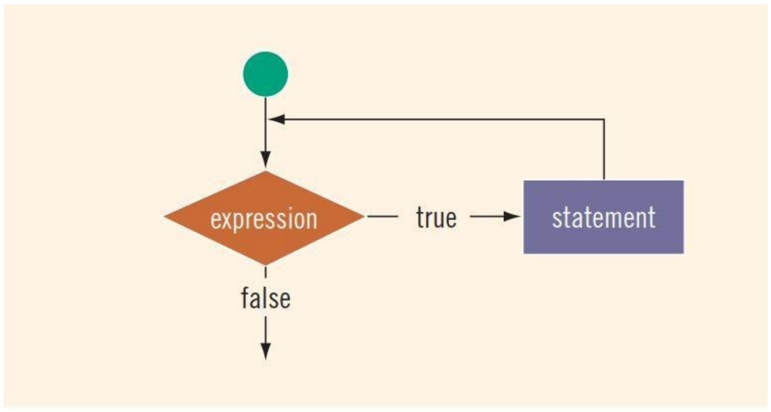
Introduction to Loops

- **loops are used to repeat a block of code.**
  - Types of Loops
    - **While**
    - **Do…While**
    - **For**

5

5

While Loop Flow Chart



6

6

While Loop Format

```
while(condition)
{
   // loop body
}
```

7

7

Sum of 5 Numbers using While Loop

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

int main()
{
	int i = 1;
	int num;
	int sum = 0;

	while (i <= 5)
	{
		cin >> num;
		sum = sum + num;
		i++;
	}

	cout << "The sum of the 5 numbers is: " << sum << endl;

return 0;
}
```

7

8

Loop Control Variable

```
//initialize the loop control variable(s)

while (expression)  //expression tests the LCV
{
    .
    .
    .
    //update the loop control variable(s)
    .
    .
    .

}
```

9

9

While Loop Controls

- Counter-Controlled `While` Loops
- Sentinel-Controlled `While` Loops
- Flag-Controlled `While` Loops

10

10

Counter-Controlled While Loop

```
counter = 0;          //initialize the loop control variable

while (counter < N)   //test the loop control variable
{
    .
    .
    .
    counter++;        //update the loop control variable
    .
    .
    .
}
```

11

11

Example of Counter-Controlled While Loop

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

int main()
{
        int i = 1;
        int num;
        int sum = 0;

        while (i <= 5)
        {
                cin >> num;
                sum = sum + num;
                i++;
        }

        cout << "The sum of the 5 numbers is: " << sum << endl;

return 0;
}
```

11

12

Sentinel-Controlled While Loop

```
cin >> variable;             //initialize the loop control variable

while (variable != sentinel)  //test the loop control variable
{
     .
     .
     .
     cin >> variable;      //update the loop control variable
     .
     .
     .
}
```

13

13

Sentinel-Controlled While Loop

14

14

Sentinel controlled

A sentinel-controlled loop in C++ is a type of indefinite repetition loop that continues to execute until a specific condition is encountered.

Sentinel-controlled loops are used when the number of iterations is unknown and depends on user input or data from an external source (like a file).

15

Example of Sentinel-Controlled `While` Loop

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

const int SENTINEL = -1;

int main()
{
        int number;     //variable to store the number
        int sum = 0;    //variable to store the sum
        int count = 0; //variable to store the total numbers read

        cout << "Enter integers ending with " << SENTINEL << endl;

        cin >> number;
        while (number != SENTINEL)
        {
                sum = sum + number;
                count++;
                cin >> number;
        }
        cout << "The sum of the " << count << " numbers is " << sum << endl;

return 0;
}
```

16

16

Flag-Controlled While Loop

```
found = false;       //initialize the loop control variable

while (!found)       //test the loop control variable
{
    .
    .
    .
    if (expression)
        found = true; //update the loop control variable
    .
    .
    .
}
```

17

17

Fibonacci Sequence

1	2	3	4	5	6	7	8	9	10	11	12	13
0	1	1	2	3	5	8	13	21	34	55	89	144

18

Find nth Fibonacci Number

1 - Populate the first two Fibonacci numbers.

2 - Get the desired Fibonacci number position, n.

3 - Calculate the next Fibonacci number by adding the previous two elements of the Fibonacci sequence.

4 - Repeat Step 3 until the nth Fibonacci number is found.

5 - Output the nth Fibonacci number.

19

19

Program of Finding nth Fibonacci Number

```
int n, Previous2 = 0, Previous1 = 1, next = 0, count = 2;
cout << "Enter the required position of the Fibonacci number: ";
cin >> n;
// Use a while loop to find the nth Fibonacci number
while (count < n) {
      next = Previous2 + Previous1;  // Calculate the next Fibonacci number
      Previous2 = Previous1;         // Update Previous2
      Previous1 = next;             // Update Previous1
      count++;                     // Increment count
}
cout << "Fibonacci number at position " << n << " is: " << next << endl;
```

20

20

Recall Reverse a Number

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

int main()
{
        int num = 1234;

        cout << num % 10 << " ";    //(1234 % 10 = 4)
        num = num / 10;             //(1234 / 10 = 123)

        cout << num % 10 << " ";    //(123 % 10 = 3)
        num = num / 10;             //(123 / 10 = 12)

        cout << num % 10 << " ";    //(12 % 10 = 2)
        num = num / 10;             //(12 / 10 = 1)

        cout << num % 10 << " ";    //(1 % 10 = 1)
        num = num / 10;             //(1 / 10 = 0)
}
```

21

21

Assignment # 01

Write a program to separate the digits of a number.

Example:

Input: 12345

output: 1 2 3 4 5

22

22

Reverse a Number using Loop

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

int main()
{

	int num = 1234;

	while (num > 0)
	{
		cout << num % 10 << " ";
		num = num / 10;
	}

	return 0;
}
```

23

23

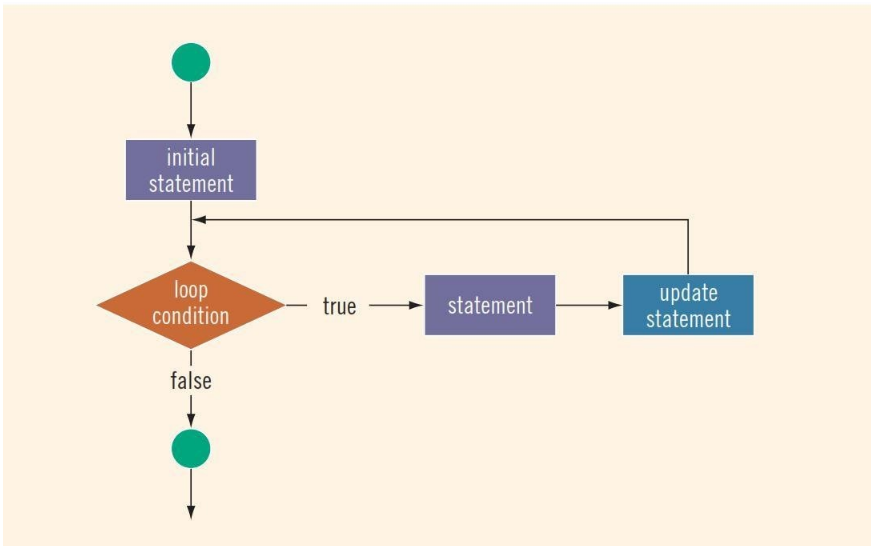
For Loop

- A Specialized Form of While Loop
- The purpose is to simplify the writing of counter-controlled loops

24

24

For Loop Flow Chart



25

25

For Loop Program

```
for (initialization; condition; update)
{
   // body of-loop
}
```

26

26

Recall the Sum of 5 Numbers Program

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

int main()
{
        int i = 1;
        int num;
        int sum = 0;

        while (i <= 5)
        {
                cin >> num;
                sum = sum + num;
                i++;
        }

        cout << "The sum of the 5 numbers is: " << sum << endl;

return 0;
}
```

24

27

Sum of 5 Numbers Program using For Loop

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

int main()
{
        int num;
        int sum = 0;

        for (int i = 1; i <= 5; i++)

        {
                cin >> num;
                sum = sum + num;
        }

        cout << "The sum of the 5 numbers is: " << sum << endl;

return 0;
}
```

25

28

Program to Calculate x^y

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

int main() {

        int x, y;
        int result = 1;
        cout << "Enter value of x:";
        cin >> x;
        cout << endl << "Enter value of y:";
        cin >> y;

        for (int i = 1; i <= y; i++)
                result *= x;

        cout << x << " power " << y << " = " << result;

return 0;
}
```

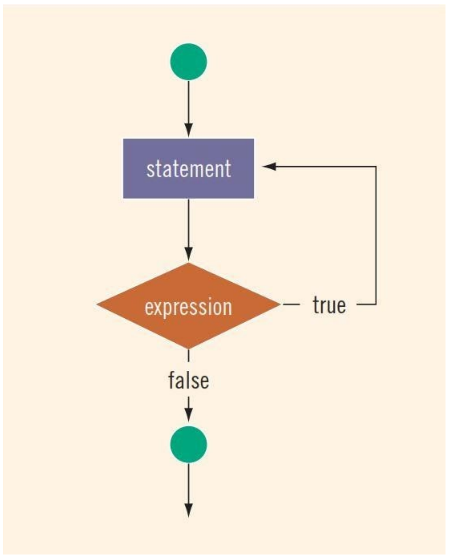
26

29

Do-While Loop

30

Do-While Loop Flow Chart



28

31

Do … While loop

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

int main()
{
        int number;

        do
        {

                cout << "Enter a number between 1 and 10: ";
                cin >> number;

        } while (number < 1 || number > 10);

        cout << "You entered: " << number << endl;

        return 0;
}
```

32

Choosing the Right Looping Structure

- If the number of repetitions can be determined in advance, use the **for** loop
- If you do not know the number of repetitions needed **while** loop is the right choice
- If you do not know the number of repetitions needed and it is at least one, the **do…while** loop is the right choice.

34

34

Break and Continue Statements

- break provides an immediate exit from the loop
- continue skips the remaining statements in the loop and proceeds with the next iteration of the loop

35

35

Continue Statement

```
#include <iostream>
int main() {
// Loop to print numbers from 1 to 10
     for (int i = 1; i <= 10; i++) {
          if (i == 5)
              continue;  // Skip the rest of the loop when i is 5
          cout << i << " ";  // Print the number if it's not 5
     }
return 0;
}
```

33

36

Nested Loops

37

37

Nested Loops

- Suppose we have to print the following pattern:

```
*
**
***
****
*****
```

38

```cpp
#include <iostream>
using namespace std;
int main() {
      for (int i = 1; i <= 10; i++)
      {
             for (int j = 1; j <= i; j++)
             {
                    cout << "*";
             }
             cout << endl;
      }
return 0;
}
```

39

Assignment #1

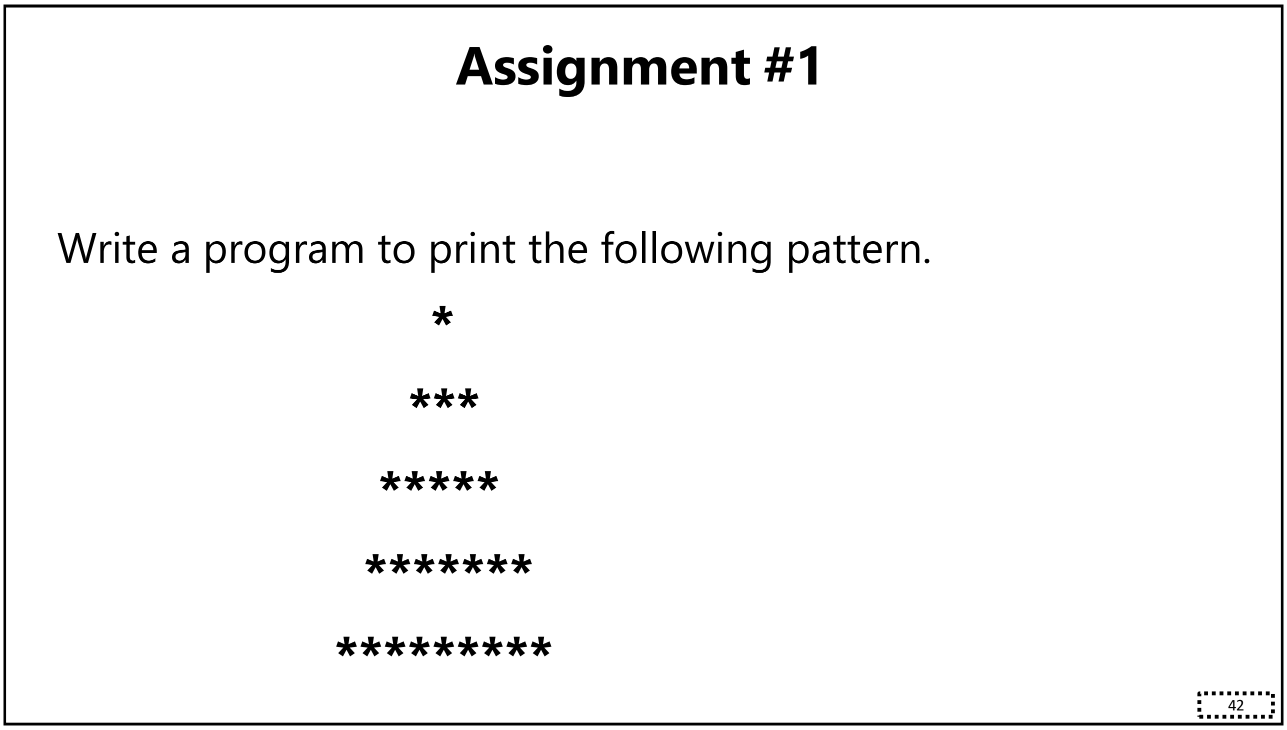
Write a program to print the following pattern.

```
    *
   **
  ***
 ****
*****
```

Assignment #1

Write a program to print the following pattern.

```
*****
****
***
**
*
```



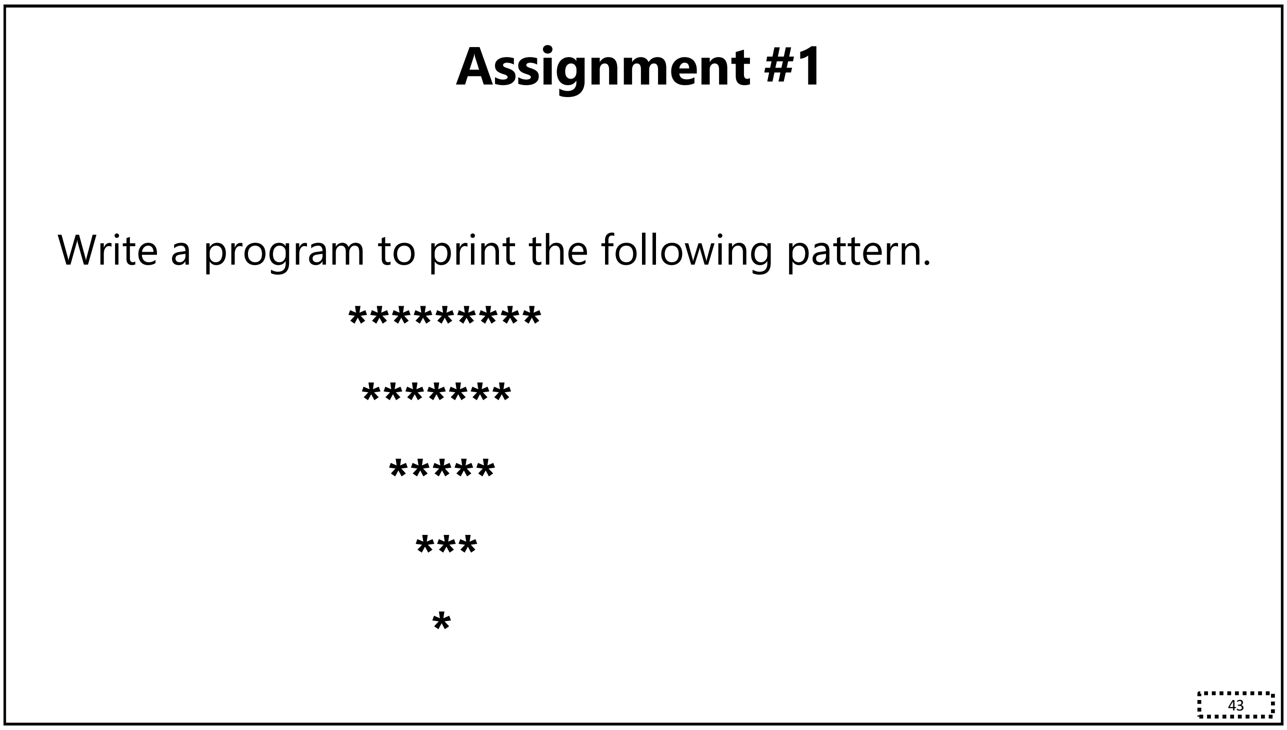
Assignment #1

Write a program to print the following pattern.

```
    *
   ***
  *****
 *******
*********
```

42

42



Assignment #1

Write a program to print the following pattern.

```
*********
 *******
  *****
   ***
    *
```

43

43

Functions

Factorial!

n! = *n* x (*n*-1) x (*n* - 2) x … x 1

For Example:

- 5! = 5 × 4 × 3 × 2 × 1 = 120
- 3! = 3 × 2 × 1 = 6
- 0! = 1 (by definition, the factorial of 0 is 1)

Factorial! Program

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

int main()
{
          int num;
          int fact = 1;
          cout << "Enter a number: ";
          cin >> num;

          for (int i = num; i > 0; i--)
                    fact *= i;

          cout << "Factorial of " << num << " is " << fact << endl;
return 0;
}
```

46

Functions

- Algebra Functions
  - f (x) = 2x + 5

Example:

- f(*1*) = 2(*1*) + 5 = 7
- f(*3*) = 2(*3*) + 5 = 11
- f(*5*) = 2(*5*) + 5 = 15

47

Function Types

- ❑ Predefined Functions
  - ○ **pow(x,y)** Calculates x^y for example $4^2 = 16$
  - ○ **abs(x)** Returns the absolute value
  - ○ **floor(x)** calculates the largest whole number that is less than or equal to x

  ➢ predefined functions are organized into separate libraries e.g. iostream, cmath

- ❑ User-defined Functions

48

Program to Calculate x^y

- ❑ Write a program to calculate x^y through a built-in function.

49

```
#include <iostream>
#include <cmath>  // Needed for the pow() function

using namespace std;

int main()
{
	double base, exponent, result;

	// Input the base and exponent
	cout << "Enter the base: ";
	cin >> base;

	cout << "Enter the exponent: ";
	cin >> exponent;

	// Calculate base^exponent using pow() function
	result = pow(base, exponent);

	// Display the result
	cout << base << " raised to the power of " << exponent << " is: " << result << endl;

return 0;
}
```

50

User-defined Function

- ❑ Value-returning Functions
- ❑ Void Functions

51

Value-returning Function

```
int larger(int x, int y ) //function definition
{
    int max;
    if (x >= y)
         max = x;
    else
        max = y;
    return max;
}
```

52

```
int larger(int x, int y ) //function definition
{
       int max;
       if (x >= y)
               max = x;
       else
              max = y;
       return max;
}
int compareThree(int x, double y, int z)
{
       return larger(x, larger(y, z));
}
int main()
{
       cout<< compareThree(5,2,9);
return 0
}
```

53

User-defined Function Prototype

```
int larger(int x, int y );
int compareThree(int x, double y, int z);

int main()
{
    …
    …
return 0
}
```

54

Assignment

Write a program to return the largest of ***n*** numbers

Input: n = 5 (9,2,13,15,1)

output: 15

Input: n = 7 (12,3,11,6,19,5,8)

output: 19

14

55

Assignment

Write a program to return the letter grade against marks

Example:

Input: 80.56

output: B+

15

56

Assignment

a) Write a value-returning function, isVowel, that returns the value true if a given character is a vowel and otherwise returns false.

Input: A output: 1 Input: B output: 0

b) Write a program that prompts the user to input a sequence of characters and outputs the number of vowels. (Use the function isVowel written in question 04 -a)

Input: qwertyabc output: 2

16

57

void Functions

- **Does not return a value:**
  - Unlike functions that return data types like int, float, or string, void functions are used when no data needs to be returned to the caller.
- **Can have parameters:**
  - Can accept parameters to operate on
- **Use return statement optionally:**
  - return; statement can be used to exit the function prematurely if needed.

58

Usage of void functions

- **Performing Actions**:
  - Use void functions when the purpose is to perform an action, like printing messages or updating values without needing feedback.
- **Modularity**:
  - Break down complex operations into smaller, manageable tasks
- **Event Handling**:
  - Often used in event-driven programming for tasks triggered by user actions, like button clicks in graphical user interfaces.

59

Best Practices for Using void functions

➢ **Clear Naming**

- Use descriptive names that convey the action performed by the function, such as printReport or updateRecord.

➢ **Parameter Usage**

- Ensure parameters are used effectively to allow the function to be flexible and applicable in various contexts.

➢ **Limit Side Effects**

- Minimize changes to global variables or data outside the function scope

60

void Function code sample

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

// Function declaration
void greet()
{
	cout << "Hello, welcome to the Applied Programming class!" << endl;
}

int main()
{
	greet(); // Function call
	return 0;
}
```

61

Exercise

- ❑ Write different patterns printing programs using functions and input parameters

62

void Function code sample

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

Void printPatern(int num)
{
      for (int i = 1; i <= num; i++)
      {
            for (int j = 1; j <= i; j++)
            {
                  cout << "*";
            }
            cout << endl;
      }

}
int main()
{
      printPatern(10);
      return 0;
}
```

63

Function Overloading

- several functions can have the same name
  - Different number of parameters
  - Different types of parameters
  - Order of parameters
  - The return type of a function is not considered

```
int calculateSum(int, int);

int calculateSum(int, int, int);

double calculateSum(float, float, float);
```

64

```
int larger(int x, int y) //function definition
{
        int max;
        if (x >= y)
                max = x;
        else
                max = y;
return max;
}

int larger(int x, int y, int z) //function definition
{
        int max;
        if (x >= y && x >= z)
                max = x;
        else if (y >= x && y >= z)
                max = y;
        else max = z;

return max;
}

int main()
{
        cout << larger(37, 9) << endl;
        cout << larger(5, 2, 9);
return 0;
}
```

65

Default Parameters for a Function

- If you do not specify the value of a default parameter, the default value is used for that parameter.
- All of the default parameters must be the far-right parameters of the function.
- Suppose a function has more than one default parameter. In a function call, if a value to a default parameter is not specified, then you must omit all of the arguments to its right.

66

default Parameter Function

```
void markAttendance(char a = 'P')
{
        cout << a;
}

int main()
{

        markAttendance('A’);
        markAttendance();
        markAttendance('P');

        return 0;
}
```

```
Output: A P P
```

67

Assignment

Write a function bool isPalindrome(int) to find whether a number is Palindrome or not?

Example:

Input: 1221

output: 1

27

68

Program for Calculating Class Average

28

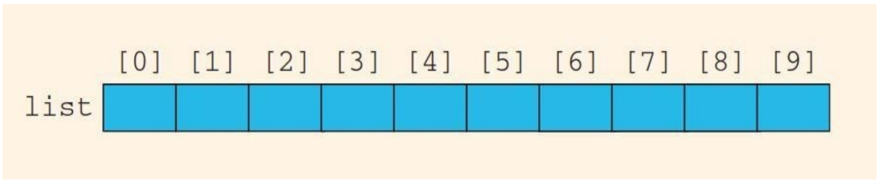
69

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

const int SENTINEL = -1;

int main()
{
        int noOfStudents = 0;
        int sumOfMarks = 0;
        int marks = 0;
        cout << "Enter " << SENTINEL << " to end…" < endl;
        cout << "Enter marks: ";

        cin >> marks;
        while (marks != SENTINEL && noOfStudents < 20)
        {
                sum = sum + marks;
                noOfStudents++;
                cin >> marks;

        }
        cout << "The Average of " << noOfStudents << " students is:" << sumOfMarks/ noOfStudents << endl;

return 0;
}
```

70

Arrays

- collection of elements of the same type
- Allow to store multiple values in a single variable



71

Arrays

Characteristics

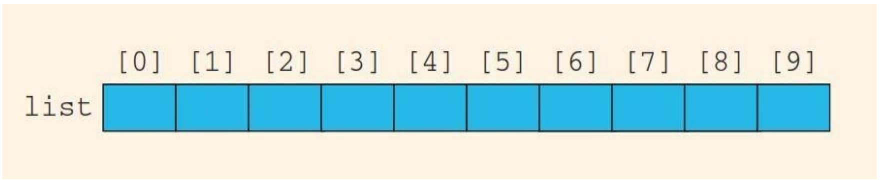
- **Fixed Size:**
  The size of an array is fixed at the time of declaration and cannot be changed.

  Zero-based Indexing:
  Array indices start at 0, meaning the first element is accessed with index 0.

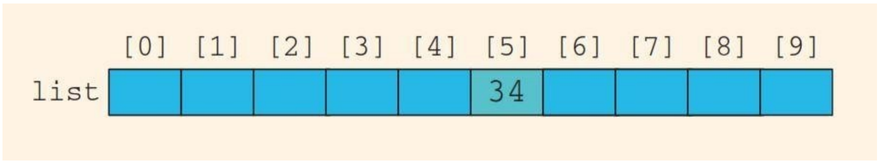
- **Contiguous Memory:**
  Elements are stored in contiguous memory locations, making access efficient.

72

Assigning Values in an Arrays



list[5] = 34;



73

Array Operations

- **Declaring an Array**

```
    float marks[10];
    int index;
    float highestMarks, sum, average;
```

- **Initializing an array**

```
    for (index = 0; index < 10; index++)
        marks[index] = 0.0;
```

- **Input Data in an Array**

```
    for (index = 0; index < 10; index++)
        cin>>marks[index];
```

- **Printing an Array**

```
    for (index = 0; index < 10; index++)
        cout<<marks[index];
```

- **Finding Sum and Average an Array**

```
   for (index = 0; index < 10; index++)
        sum = sum + marks[index];
   average = sum / 10;
```

74

Array Operations

- **Finding highest marks in marks[10] ??**

75

Program to calculate average

```
int main()
{

	float sumOfMarks = 0;
	int count = 0;
	float marks[10];

	cout << "Enter marks: ";

	for (int i = 0; i < 10; i++)
		cin >> marks[i];

	for (int i = 0; i < 10; i++)
		sumOfMarks += marks[count];

	cout << "The Class Average is:" << sumOfMarks / 10 << endl;

return 0;
}
```

76

Assignment

Write a program that

1. Declares an array of 50 components of type double.
2. Initialize the array so that the first 25 components are equal to the square of the index variable, and the last 25 components are equal to three times the index variable.
3. Output the array so that 10 elements per line are printed.

36

77

Removing duplicate Values in an Arrays

Input: [1,1,2,3,4,4,4,5,5,6,9]

Output: [1,2,3,4,5,6,9,0,0,0,0]

78

Removing duplicate Values in an Arrays

```
int main()
{
	int arr[11] = { 1,1,2,3,4,4,4,5,5,6,9 };
	int resultArr[11] = { 0 };
	int j = 0;

	resultArr[j] = arr[0];

	for (int i = 1; i < 11; i++)
	{
		if (resultArr[j] != arr[i])
			resultArr[++j] = arr[i];
	}

return 0;
}
```

79

Prime Number Function

```
bool isPrime(int num) //function definition
{

        for(int i = 2; i<num; i++)
            if (num % i == 0)
                return false;

    return true;
}
```

Num	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

80