Lesson 1 C++ Activity Summary

Contents

Vector and String Output with Range-based For Loop

```
/*
Comment
Block
*/
#include <iostream> // Enables cout for output, cin for input...etc
#include <vector> // Enables use of vector container(a dynamic array)
#include <string> // Enables use of string type

using namespace std; // Allows using cout, vector, string without std:: prefix

int main() { //Entry point of the program
    vector<string> msg {"Hello", "C++", "World", "from", "VS Code", "and the C++ exte
    for (const string& word : msg) { // Iterates through each string in the vector, (
        cout << word << " "; // Prints each word with a space
    }
    cout << endl; // Moves to the next line after printing all words
    return 0; // Signals successful program completion
}
```

Basic Counter-Based For Loop

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

int main()
{
    for (int i{1};i<=10;++i)
    {
        cout << i << " ";
    }
    cout << "\nFor loop using counter";
    cout << endl;
    return 0;
}
```

Range-Based For Loop over an Array

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

int main()
{
    int numbers[] = {1,2,3,4,5,6,7,8,9,10,11};

    for(auto val: numbers)
```

```
    {
        cout << val << endl;
    }
    cout << "For loop using array";
    cout << endl;
    return 0;
}
```

While Loop Example with Character Input

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

int main()
{
    char c{'y'};
    while(c == 'y')
    {
        cout << "Loop again (y/n)";
        cin >> c;
    }
    cout << "last insput was not 'y'"<< endl;
    return 0;
}
```

Function to Multiply Two Numbers

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

// to multiply two numbers
double multiply(double a,double b){
    return a * b;
}

int main(){
    int x{},y{};
    cout << "Please enter two numbers: ";
    cin >> x >> y;
    int result = multiply(x,y);
    cout << "Product of " << x << " and " << y << " = " << result << endl;
    return 0;
}
```

Function Templates for Addition

```
#include <iostream>
```

```
using namespace std;

// Overloaded functions for addition

template <typename X>
X add(X a, X b) {
    return a + b;
}

template <typename X,typename Z>
auto add(X a, Z b) {
    return a + b;
}

// int add(int a, int b) {
//     return a + b;
// }

// float add(float a, float b) {
//     return a + b;
// }

// double add(double a, double b) {
//     return a + b;
// }

int main() {
    // Testing overloaded functions
    cout << "Int addition: " << add(3, 5) << endl;       // Calls int version
    cout << "Float addition: " << add(3.5f, 2.2f) << endl;  // Calls float version
    cout << "Float addition: " << add(3.5f, 9) << endl;  // Calls float version
    cout << "Float addition: " << add(7, 3.5f) << endl;  // Calls float version
    cout << "Double addition: " << add(4.7, 1.3) << endl;  // Calls double version

    return 0;
}
```

Using std::initializer$_l istinTemplates$

```
#include <iostream>
#include <initializer_list> // Include necessary header

using namespace std;

// Function definition
template <typename X>
void processItems(std::initializer_list<X> items) {
    for(auto i: items){
```

```
        cout << i << " ";
    }
    cout << endl;
}

int main() {
    processItems({10, 20, 30, 40, 50});
    processItems({"The", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "do
    return 0;
}
```

Input and Output with Auto Type Deduction

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

int main()
{
    int x{},y{99};
    cout << "Please enter two numbers: ";
    cin >> x >> y;
    auto sum = x + y;
    cout<< x <<" + "<< y << " = " << sum << endl;
    cout<< " Of data type: "<< typeid(sum).name() << endl;
    return 0;
}

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

// int main()99 1

// {
//     int x{},y{99};
//     int sum{};
//     cout << "Please enter two numbers: ";
//     cin >> x >> y;
//     //int sum = x + y;
//     sum = x + y;
//     cout<< x <<" + "<< y << " = " << sum << std::endl;
//     return 0;
// }
```

```
// #include <iostream>

// int main()
// {
//     int x{},y{99};
//     int sum{};
//     std::cout << "Please enter two numbers: ";
//     std::cin >> x >> y;
//     //int sum = x + y;
//     sum = x + y;
//     std::cout<< x <<" + "<< y << " = " << sum << std::endl;
//     return 0;
// }
```

Function with Default Parameters

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

// Function prototype with default arguments
void displayStars(int cols = 10, int rows = 1);

int main()
{
    displayStars();
    cout << endl;
    displayStars(5);
    cout << endl;
    displayStars(7,3);

    return 0;
}

// Function definition
void displayStars(int cols, int rows)
{
    // Nested loop. The outer loop controls the rows
    // and the inner loop controls the columns.
    for (int down = 0; down < rows; down++)
    {
        for (int across = 0; across < cols; across++)
        {
            cout << "*";
        }
        cout << endl;
    }
}
```

Introduction to References

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

int main(){

    char letter{'a'};
    int num_1{0};
    int num_2{0};
    double num_3{0};

    cout << "Enter an integer: ";
    cin >> num_1;
    cout << "Enter another integer: ";
    cin >> num_2;
    cout << "Enter an double: ";
    cin >> num_3;
    cout << "Enter an Character: ";
    cin >> letter;

    int &w = num_1;
    int &x = num_2;
    double &y = num_3;
    char &z = letter;

    cout << "Value of num_1: " << num_1 << endl;
    cout << "Value of num_1 reference: " << w << endl;

    cout << "Value of num_2: " << num_2 << endl;
    cout << "Value of num_2 reference: " << x << endl;

    cout << "Value of num_3: " << num_3 << endl;
    cout << "Value of num_3 reference: " << y << endl;

    cout << "Value of letter: " << letter << endl;
    cout << "Value of letter reference: " << z << endl;
}
```

Modifying Values via References in Functions

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

template <typename X>
```

```
void square(X &input) {
    input = input * input;
}
void flipCase(char &c) {
    if (isupper(c)) {
        c = tolower(c);  // Convert to lowercase
    } else if (islower(c)) {
        c = toupper(c);  // Convert to uppercase
    }
}

int main(){

    char letter{'a'};
    int num_1{0};
    int num_2{0};
    double num_3{0};

    cout << "Enter an integer: ";
    cin >> num_1;
    cout << "Enter another integer: ";
    cin >> num_2;
    cout << "Enter an double: ";
    cin >> num_3;
    cout << "Enter an Character: ";
    cin >> letter;

    int &w = num_1;
    int &x = num_2;
    double &y = num_3;
    char &z = letter;

    cout << "Value of num_1: " << num_1 << endl;
    cout << "Value of num_1 reference: " << w << endl;
    square(num_1);
    cout << "Value of num_1 after function call: " << num_1 << endl;
    cout << "Value of num_1 reference after function call: " << w << endl;
    cout << "################################################" << endl;

    cout << "Value of num_2: " << num_2 << endl;
    cout << "Value of num_2 reference: " << x << endl;
    square(num_2);
    cout << "Value of num_2 after function call: " << num_2 << endl;
    cout << "Value of num_2 reference after function call: " << x << endl;
    cout << "################################################" << endl;

    cout << "Value of num_3: " << num_3 << endl;
    cout << "Value of num_3 reference: " << y << endl;
```

```
    square(num_3);
    cout << "Value of num_3 after function call: " << num_3 << endl;
    cout << "Value of num_3 reference after function call: " << y << endl;
    cout << "################################################" << endl;

    cout << "Value of letter: " << letter << endl;
    cout << "Value of letter reference: " << z << endl;
    flipCase(letter);
    cout << "Value of letter after function call: " << letter << endl;
    cout << "Value of letter reference after function call: " << z << endl;
    cout << "################################################" << endl;
}
```

Vector Operations and Range-based Iteration

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

// Method to print numbers
void printNumbers(const vector<int>& numbers) {
    for (int num : numbers) {
        cout << num << " ";
    }
    cout << endl;
}

int main(){
    vector<int> numbers = {10,20,30};

    // Add elements
    numbers.push_back(40);printNumbers(numbers);
    numbers.pop_back();printNumbers(numbers);

    // Access elemnts
    cout << numbers[1] << " ";
    cout << numbers.at(2) << " ";

    // Iterate using range-based loop
    for (int num: numbers){
        cout << num << " ";
    }
    cout << endl;

    numbers.reserve(100);  // Preallocate memory
    numbers.resize(5);printNumbers(numbers);

    return 0;
```

```
}
```

User Input into Array with Range-based Loop (Note: VLAs)

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

int main(){

    int sz{};
    cout << "Enter the array size: ";
    cin >> sz;

    int numbers[sz];

    //Get values for the array
    for(int &val:numbers){
        cout << "Enter an integer value: ";
        cin >> val;
    }

    // Diplay the values in the array
    cout << "Here are the values you entered:\n";
    for(int val:numbers){
        cout << val << endl;
    }
    return 0;
}
```

Reference Variable Output Example

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

int main(){

    char letter{'a'};
    int num_1{0};
    int num_2{0};
    double num_3{0};

    cout << "Enter an integer: ";
    cin >> num_1;
    cout << "Enter another integer: ";
    cin >> num_2;
    cout << "Enter an double: ";
    cin >> num_3;
```

```
    cout << "Enter an Character: ";
    cin >> letter;

    int &w = num_1;
    int &x = num_2;
    double &y = num_3;
    char &z = letter;

    cout << "Value of num_1: " << num_1 << endl;
    cout << "Value of num_1 reference: " << w << endl;

    cout << "Value of num_2: " << num_2 << endl;
    cout << "Value of num_2 reference: " << x << endl;

    cout << "Value of num_3: " << num_3 << endl;
    cout << "Value of num_3 reference: " << y << endl;

    cout << "Value of letter: " << letter << endl;
    cout << "Value of letter reference: " << z << endl;
}
```

Reference After Function Call Example

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

template <typename X>
void square(X &input) {
    input = input * input;
}
void flipCase(char &c) {
    if (isupper(c)) {
        c = tolower(c);  // Convert to lowercase
    } else if (islower(c)) {
        c = toupper(c);  // Convert to uppercase
    }
}

int main(){

    char letter{'a'};
    int num_1{0};
    int num_2{0};
    double num_3{0};

    cout << "Enter an integer: ";
    cin >> num_1;
    cout << "Enter another integer: ";
```

```
    cin >> num_2;
    cout << "Enter an double: ";
    cin >> num_3;
    cout << "Enter an Character: ";
    cin >> letter;

    int &w = num_1;
    int &x = num_2;
    double &y = num_3;
    char &z = letter;

    cout << "Value of num_1: " << num_1 << endl;
    cout << "Value of num_1 reference: " << w << endl;
    square(num_1);
    cout << "Value of num_1 after function call: " << num_1 << endl;
    cout << "Value of num_1 reference after function call: " << w << endl;
    cout << "################################################" << endl;

    cout << "Value of num_2: " << num_2 << endl;
    cout << "Value of num_2 reference: " << x << endl;
    square(num_2);
    cout << "Value of num_2 after function call: " << num_2 << endl;
    cout << "Value of num_2 reference after function call: " << x << endl;
    cout << "################################################" << endl;

    cout << "Value of num_3: " << num_3 << endl;
    cout << "Value of num_3 reference: " << y << endl;
    square(num_3);
    cout << "Value of num_3 after function call: " << num_3 << endl;
    cout << "Value of num_3 reference after function call: " << y << endl;
    cout << "################################################" << endl;

    cout << "Value of letter: " << letter << endl;
    cout << "Value of letter reference: " << z << endl;
    flipCase(letter);
    cout << "Value of letter after function call: " << letter << endl;
    cout << "Value of letter reference after function call: " << z << endl;
    cout << "################################################" << endl;
}
```

If-Else Statement Example

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

int main() {
    int number;
    cout << "Enter a number: ";
```

```
    cin >> number;

    if (number > 0) {
        cout << "The number is positive." << endl;
    } else if (number < 0) {
        cout << "The number is negative." << endl;
    } else {
        cout << "The number is zero." << endl;
    }

    return 0;
}
```