c++ Assignment

Basics Of C++

SUBMITTED BY: Rahul Kumar (85)

**Section 1: Basic Calculations and Input/Output**

**1. Area and Circumference of a Circle**

#**include** <iostream>

#**include** <iomanip> *// Required for setprecision*

**int** main() {

**double** radius, area, circumference;

**const** **double** PI = 3.14159;

std::cout << "Enter the radius of the circle: ";

std::cin >> radius;

area = PI \* radius \* radius;

circumference = 2 \* PI \* radius;

std::cout << std::fixed << std::setprecision(2); *// Set precision to 2 decimal points*

std::cout << "Area: " << area << std::endl;

std::cout << "Circumference: " << circumference << std::endl;

**return** 0;

}

**Example Interactions:**

* **Input:**
  + Radius: 5
* **Output:**

Area: 78.54

Circumference: 31.42

**2. Evaluating an Expression**

#**include** <iostream>

#**include** <cmath> *// Required for pow function*

**int** main() {

**double** a, b, c, d, e, result;

std::cout << "Enter the values of a, b, c, d, and e: ";

std::cin >> a >> b >> c >> d >> e;

result = pow((a + b \* c - d / e), 2);

std::cout << "Result: " << result << std::endl;

**return** 0;

}

**Example Interactions:**

* **Input:**
  + a: 1
  + b: 2
  + c: 3
  + d: 4
  + e: 5
* **Output:**

Result: 43.56

**Section 2: Conditional Statements**

**3. Prime Number Check (Nested if Statements)**

#**include** <iostream>

**int** main() {

**int** number;

std::cout << "Enter a number: ";

std::cin >> number;

**if** (number <= 1) {

std::cout << number << " is not a prime number." << std::endl;

} **else** {

**if** (number == 2) {

std::cout << number << " is a prime number." << std::endl;

} **else** {

**if** (number % 2 == 0) {

std::cout << number << " is not a prime number." << std::endl;

} **else** {

**if** (number == 3) {

std::cout << number << " is a prime number." << std::endl;

}

**else** **if** (number % 3 == 0) {

std::cout << number << " is not a prime number." << std::endl;

}**else**{

std::cout << number << " is a prime number." << std::endl;

}

}

}

}

**return** 0;

}

**Example Interactions:**

* **Input:**
  + Number: 7
* **Output:**

7 is a prime number.

* **Input:**
  + Number: 4
* **Output:**

4 is not a prime number.

**Section 3: Increment Operators**

**4. Post-increment vs. Pre-increment**

#**include** <iostream>

**int** main() {

**int** i = 5;

std::cout << "Initial value of i: " << i << std::endl;

std::cout << "Post-increment (i++): " << i++ << std::endl;

std::cout << "Value of i after post-increment: " << i << std::endl;

i = 5; *// Reset i to 5*

std::cout << "Initial value of i: " << i << std::endl;

std::cout << "Pre-increment (++i): " << ++i << std::endl;

std::cout << "Value of i after pre-increment: " << i << std::endl;

**return** 0;

}

**Example Output:**

Initial value of i: 5

Post-increment (i++): 5

Value of i after post-increment: 6

Initial value of i: 5

Pre-increment (++i): 6

Value of i after pre-increment: 6

**Section 4: Arrays**

**5. Sum of Even Numbers and Product of Odd Numbers in an Array**

#**include** <iostream>

**int** main() {

**int** numbers[10];

**int** sumOfEven = 0;

**int** productOfOdd = 1;

std::cout << "Enter 10 integers:" << std::endl;

**for** (**int** i = 0; i < 10; ++i) {

std::cin >> numbers[i];

}

**for** (**int** i = 0; i < 10; ++i) {

**if** (numbers[i] % 2 == 0) {

sumOfEven += numbers[i];

} **else** {

productOfOdd \*= numbers[i];

}

}

std::cout << "Sum of even numbers: " << sumOfEven << std::endl;

std::cout << "Product of odd numbers: " << productOfOdd << std::endl;

**return** 0;

}

**Example Interactions:**

* **Input:**
  + 1 2 3 4 5 6 7 8 9 10
* **Output:**

Sum of even numbers: 30

Product of odd numbers: 945

**Section 5: Matrices**

**6. Transpose of a 3x3 Matrix**

#**include** <iostream>

**int** main() {

**int** matrix[3][3];

std::cout << "Enter the elements of the 3x3 matrix:" << std::endl;

**for** (**int** i = 0; i < 3; ++i) {

**for** (**int** j = 0; j < 3; ++j) {

std::cin >> matrix[i][j];

}

}

std::cout << "Transpose of the matrix:" << std::endl;

**for** (**int** i = 0; i < 3; ++i) {

**for** (**int** j = 0; j < 3; ++j) {

std::cout << matrix[j][i] << " ";

}

std::cout << std::endl;

}

**return** 0;

}

**Example Interactions:**

* **Input:**

1 2 3

4 5 6

7 8 9

* **Output:**

Transpose of the matrix:

1 4 7

2 5 8

3 6 9

**Section 6: Strings**

**7. Counting Vowels, Consonants, Digits, and Special Characters**

#**include** <iostream>

#**include** <string>

**int** main() {

std::string str;

**int** vowels = 0, consonants = 0, digits = 0, specialChars = 0;

std::cout << "Enter a string: ";

std::getline(std::cin, str); *// Use getline to read the entire line*

**for** (**char** c : str) {

**if** (isalpha(c)) {

c = tolower(c); *// Convert to lowercase for easy comparison*

**if** (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u') {

vowels++;

} **else** {

consonants++;

}

} **else** **if** (isdigit(c)) {

digits++;

} **else** {

specialChars++;

}

}

std::cout << "Vowels: " << vowels << std::endl;

std::cout << "Consonants: " << consonants << std::endl;

std::cout << "Digits: " << digits << std::endl;

std::cout << "Special characters: " << specialChars << std::endl;

**return** 0;

}

**Example Interactions:**

* **Input:**
  + Hello, World! 123
* **Output:**

Vowels: 3

Consonants: 7

Digits: 3

Special characters: 3

**Section 7: Patterns**

**8. Printing a Number Pattern**

#**include** <iostream>

**int** main() {

**int** n;

std::cout << "Enter the number of rows: ";

std::cin >> n;

**for** (**int** i = 1; i <= n; ++i) {

**for** (**int** j = 1; j <= i; ++j) {

std::cout << j << " ";

}

std::cout << std::endl;

}

**return** 0;

}

**Example Interactions:**

* **Input:**
  + Number of rows: 4
* **Output:**

1

1 2

1 2 3

1 2 3 4

**Section 8: Function Pointers and Dynamic Binding**

**9. Dynamic Binding with Function Pointers**

#**include** <iostream>

**int** add(**int** a, **int** b) {

**return** a + b;

}

**int** subtract(**int** a, **int** b) {

**return** a - b;

}

**int** main() {

**int** a, b, choice;

std::cout << "Enter two integers: ";

std::cin >> a >> b;

std::cout << "Enter 1 for addition, 2 for subtraction: ";

std::cin >> choice;

**int** (\*operation)(**int**, **int**); *// Function pointer declaration*

**if** (choice == 1) {

operation = add;

} **else** **if** (choice == 2) {

operation = subtract;

} **else** {

std::cout << "Invalid choice." << std::endl;

**return** 1;

}

**int** result = operation(a, b); *// Dynamic function call*

std::cout << "Result: " << result << std::endl;

**return** 0;

}

**Example Interactions:**

* **Input:**
  + Two integers: 10 5
  + Choice: 1
* **Output:**

Result: 15

* **Input:**
  + Two integers: 10 5
  + Choice: 2
* **Output:**

Result: 5

**Section 9: Function Calls and Operator Input**

**10. Performing Operations Based on User Input**

#**include** <iostream>

**int** performOperation(**int** a, **int** b, **char** op) {

**switch** (op) {

**case** '+':

**return** a + b;

**case** '-':

**return** a - b;

**case** '\*':

**return** a \* b;

**case** '/':

**if** (b == 0) {

std::cout << "Error: Division by zero!" << std::endl;

**return** 0;

}

**return** a / b;

**default**:

std::cout << "Error: Invalid operator!" << std::endl;

**return** 0;

}

}

**int** main() {

**int** num1, num2, result;

**char** op;

std::cout << "Enter two integers: ";

std::cin >> num1 >> num2;

std::cout << "Enter an operator (+, -, \*, /): ";

std::cin >> op;

result = performOperation(num1, num2, op);

std::cout << "Result: " << result << std::endl;

**return** 0;

}

**Example Interactions:**

* **Input:**
  + Two integers: 10 5
  + Operator: +
* **Output:**

Result: 15

* **Input:**
  + Two integers: 10 5
  + Operator: /
* **Output:**

Result: 2

* **Input:**
  + Two integers: 10 0
  + Operator: /
* **Output:**

Error: Division by zero!

Result: 0