**Program 1**: Factors of a Positive Integer

#include <iostream>

int main() {

int N;

// Input validation using do-while loop

do {

std::cout << "Enter a positive integer: ";

std::cin >> N;

} while (N <= 0);

std::cout << "Factors = ";

for (int i = 1; i <= N; ++i) {

if (N % i == 0) {

std::cout << i << " ";

}

}

return 0;

}

**Output Snip:**

**Program 2:** Reverse a Positive Integer

#include <iostream>

int main() {

int N;

// Input validation using do-while loop

do {

std::cout << "Enter a positive integer: ";

std::cin >> N;

} while (N <= 0);

int reverse = 0;

while (N > 0) {

reverse = reverse \* 10 + N % 10;

N /= 10;

}

std::cout << "Reverse = " << reverse;

return 0;

}

**Output Snip:**

**Program 3:** Sum of Numbers between Two Positive Integers N and M

#include <iostream>

int main() {

int N, M;

// Input validation using do-while loop

do {

std::cout << "Enter two positive integers (N and M): ";

std::cin >> N >> M;

} while (N <= 0 || M <= 0);

int sum = 0;

for (int i = N; i <= M; ++i) {

sum += i;

}

std::cout << "Sum = " << sum;

return 0;

}

**Output Snip:**

**Program 4:** Prime Numbers between Two Positive Integers

#include <iostream>

int main() {

int N, M;

// Input validation using do-while loop

do {

std::cout << "Enter two positive integers (N and M): ";

std::cin >> N >> M;

} while (N <= 0 || M <= 0);

std::cout << "Prime Numbers = ";

for (int num = N; num <= M; ++num) {

bool isPrime = true;

if (num <= 1) {

isPrime = false;

} else {

for (int i = 2; i \* i <= num; ++i) {

if (num % i == 0) {

isPrime = false;

break;

}

}

}

if (isPrime) {

std::cout << num << " ";

}

}

return 0;

}

**Output Snip:**

**Program 5:** Find the First Digit in a Positive Integer

#include <iostream>

int main() {

int Number;

// Input validation using do-while loop

do {

std::cout << "Enter a positive integer: ";

std::cin >> Number;

} while (Number <= 0);

int firstDigit;

while (Number >= 10) {

firstDigit = Number / 10;

Number /= 10;

}

std::cout << "First Digit = " << Number;

return 0;

}

**Output Snip:**

**Program 6:** Sum of Digits of a Positive Integer

#include <iostream>

int main() {

int Number;

// Input validation using do-while loop

do {

std::cout << "Enter a positive integer: ";

std::cin >> Number;

} while (Number <= 0);

int sum = 0;

while (Number > 0) {

sum += Number % 10;

Number /= 10;

}

std::cout << "Sum of digits = " << sum;

return 0;

}

**Output Snip:**

**Program 7:** Sum of Digits of an Integer (Positive or Negative)

#include <iostream>

int main() {

int Number;

// Input validation using do-while loop

do {

std::cout << "Enter an integer: ";

std::cin >> Number;

} while (std::cin.fail()); //cin.fail() returns true when input is a non-integer value like string

Number = -1\*(Number);

int sum = 0;

while (Number > 0) {

sum += Number % 10;

Number /= 10;

}

std::cout << "Sum of digits = " << sum;

return 0;

}

**Output Snip:**

**Program 8:** Product of Digits of a Positive Integer

#include <iostream>

int main() {

int Number;

// Input validation using do-while loop

do {

std::cout << "Enter a positive integer: ";

std::cin >> Number;

} while (Number <= 0);

int product = 1;

while (Number > 0) {

product \*= Number % 10;

Number /= 10;

}

std::cout << "Product of digits = " << product;

return 0;

}

**Output Snip:**

**Program 9:** Check if a Positive Integer is a Palindrome

#include <iostream>

int main() {

int Number, temp;

// Input validation using do-while loop

do {

std::cout << "Enter a positive integer: ";

std::cin >> Number;

} while (Number <= 0);

temp = Number;

int reverse = 0;

while (temp > 0) {

reverse = reverse \* 10 + temp % 10;

temp /= 10;

}

if (Number == reverse) {

std::cout << "A palindrome.";

} else {

std::cout << "Not a palindrome.";

}

return 0;

}

**Output Snip:**

**Program 10:** Check if a Positive Integer is a Perfect Number

#include <iostream>

int main() {

int N;

// Input validation using do-while loop

do {

std::cout << "Enter a positive integer: ";

std::cin >> N;

} while (N <= 0);

int sum = 0;

for (int i = 1; i <= N / 2; ++i) {

if (N % i == 0) {

sum += i;

}

}

if (sum == N) {

std::cout << "A perfect number.";

} else {

std::cout << "Not a perfect number.";

}

return 0;

}

**Output Snip:**

**Program 11:** Check if a Positive Integer is an Armstrong Number

#include <iostream>

int main() {

int num;

do {

std::cout << "Enter a positive integer: ";

std::cin >> num;

} while (num <= 0);

int originalNum = num;

int digitCount = 0;

int sum = 0;

// Count the number of digits in the input number

int temp = num;

while (temp != 0) {

digitCount++;

temp /= 10;

}

// Calculate the sum of digits raised to the power of digitCount

temp = num;

while (temp != 0) {

int digit = temp % 10;

int digitPowerN = 1;

for (int i = 0; i < digitCount; i++)

digitPowerN \*= digit;

sum += digitPowerN;

temp /= 10;

}

// Check if it's an Armstrong number

if (sum == originalNum) {

std::cout << originalNum << " is an Armstrong number." << std::endl;

} else {

std::cout << originalNum << " is not an Armstrong number." << std::endl;

}

return 0;

}

**Output Snip:**

**Program 12:** Count of a Digit in a Given Positive Integer

#include <iostream>

int main() {

int Number;

// Input validation using do-while loop

do {

std::cout << "Enter a positive integer: ";

std::cin >> Number;

} while (Number <= 0);

int digit;

// Input validation using do-while loop

do {

std::cout << "Enter a digit (0-9): ";

std::cin >> digit;

} while (digit < 0 || digit > 9);

int count = 0;

while (Number > 0) {

if (Number % 10 == digit) {

count++;

}

Number /= 10;

}

std::cout << "Count of digit " << digit << " in the number = " << count;

return 0;

}

**Output Snip:**

**Program 13:** Print Digits in a Number as Words

#include <iostream>

int main() {

int N;

// Input validation using do-while loop

do {

std::cout << "Enter a positive integer: ";

std::cin >> N;

} while (N <= 0);

int digit;

int temp = N;

int numDigits = 0;

// Count the number of digits in the integer

while (temp > 0) {

temp /= 10;

numDigits++;

}

// Arrays to store the words for digits 1-9 and 10-19

const char\* words1to9[] = {"One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine"};

const char\* words10to19[] = {"Ten", "Eleven", "Twelve", "Thirteen", "Fourteen", "Fifteen", "Sixteen", "Seventeen", "Eighteen", "Nineteen"};

// Arrays to store the words for tens multiples

const char\* wordsTens[] = {"", "", "Twenty", "Thirty", "Forty", "Fifty", "Sixty", "Seventy", "Eighty", "Ninety"};

// Program 13:

int divisor = 1;

for (int i = 1; i < numDigits; ++i) {

divisor \*= 10;

}

std::cout << "Result = ";

while (divisor > 0) {

int currentDigit = N / divisor;

N %= divisor;

divisor /= 10;

if (currentDigit > 0) {

if (numDigits == 2 && currentDigit == 1) {

int nextDigit = N % 10;

std::cout << words10to19[nextDigit] << " ";

break;

} else if (numDigits == 2 && currentDigit > 1) {

std::cout << wordsTens[currentDigit] << " ";

} else {

std::cout << words1to9[currentDigit - 1] << " ";

}

}

numDigits--;

}

return 0;

}

**Output Snip:**

**Program 14:** Print a Number in Words

#include <iostream>

int main() {

int N;

// Input validation using do-while loop

do {

std::cout << "Enter a positive integer: ";

std::cin >> N;

} while (N <= 0);

// Arrays to store the words for digits 1-9 and 10-19

const char\* words1to9[] = {"One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine"};

const char\* words10to19[] = {"Ten", "Eleven", "Twelve", "Thirteen", "Fourteen", "Fifteen", "Sixteen", "Seventeen", "Eighteen", "Nineteen"};

// Arrays to store the words for tens multiples

const char\* wordsTens[] = {"", "", "Twenty", "Thirty", "Forty", "Fifty", "Sixty", "Seventy", "Eighty", "Ninety"};

int numDigits = 0;

int temp = N;

while (temp > 0) {

temp /= 10;

numDigits++;

}

int divisor = 1;

for (int i = 1; i < numDigits; ++i) {

divisor \*= 10;

}

while (divisor > 0) {

int currentDigit = N / divisor;

N %= divisor;

divisor /= 10;

if (currentDigit > 0) {

if (numDigits == 2 && currentDigit == 1) {

int nextDigit = N % 10;

std::cout << words10to19[nextDigit] << " ";

break;

} else if (numDigits == 2 && currentDigit > 1) {

std::cout << wordsTens[currentDigit] << " ";

} else {

std::cout << words1to9[currentDigit - 1] << " ";

}

}

numDigits--;

}

return 0;

}

**Output Snip:**