Data types and Operators

C++ has a rich set of data types and operators that you can use to create powerful and flexible applications. In this section, we'll take a look at some of the most important data types and operators in C++.

Integer Data Types

Integers are whole numbers that can be positive, negative, or zero. In C++, there are four integer data types:

signed char
unsigned char
signed int
unsigned int

The **signed char** data type can store values from **-128 to 127**. The **unsigned char** data type can store values from **0 to 255**. The **signed int** data type can store values from **-32,768 to 32,767**. The **unsigned int** data type can store values from **0 to 65,535**.

You can declare an integer variable like this:

```
int my_int;
```

You can assign a value to an integer variable like this:

```
my_int = 42;
```

Arithmetic Operators

C++ has a full set of arithmetic operators that can be used on integer data types. The operators are:

+ addition
- subtraction

*** multiplication**
/ division
% modulo (remainder after division)

For example, the following code calculates the result of adding two integers:

```
int x = 5;
int y = 3;
int result = x + y; // result is 8
```

The following code calculates the result of subtracting two integers:

```
int x = 5;
int y = 3;
int result = x - y; // result is 2
```

The following code calculates the result of multiplying two integers:

```
int x = 5;
int y = 3;
int result = x * y; // result is 15
```

The following code calculates the result of dividing two integers:

```
int x = 5;
int y = 3;
int result = x / y; // result is 1
```

The following code calculates the remainder after division of two integers:

```
int x = 5;
int y = 3;
int result = x % y; // result is 2
```

Comparison Operators

C++ has a full set of comparison operators that can be used to compare two integer values. The operators are:

== equal to
!= not equal to
> greater than
< less than
>= greater than or equal to
<= less than or equal to

For example, the following code compares two integers:

```
int x = 5;
int y = 3;

if (x == y)
{
   std::cout << "x is equal to y" << std::endl;
}
else if (x > y)
{
   std::cout << "x is greater than y" << std::endl;
}
else if (x < y)
{
   std::cout << "x is less than y" << std::endl;
}
```

Logical Operators

C++ has a full set of logical operators that can be used to combine multiple comparisons. The operators are:

&& logical AND
|| logical OR
! logical NOT

For example, the following code uses the logical AND operator to compare two integers:

```
int x = 5;
int y = 3;

if (x == 5 && y == 3)
{
   std::cout << "x is 5 and y is 3" << std::endl;
}
```

The following code uses the logical OR operator to compare two integers:

```
int x = 5;
int y = 3;

if (x == 5 || y == 3)
{
   std::cout << "x is 5 or y is 3" << std::endl;
}
```

The following code uses the logical NOT operator to compare two integers:

```
int x = 5;
int y = 3;

if (!(x == y))
{
   std::cout << "x is not equal to y" << std::endl;
}
```