C++ Variables

Setenay Tutucu

December 2024

0.1 C++ Variables

In the C++ programming language, variables are fundamental components that are used to store data that can be manipulated during the execution of a program. Each variable in C++ has a specific data type, which determines the kind of data it can store, such as integers, floating-point numbers, characters, etc.

0.1.1 Characters

In C++, characters refer to individual symbols or letters that are represented using the **char** data type. These characters can include letters (both uppercase and lowercase), digits, punctuation marks, and special symbols.

Properties

- The size of a **char** is typically 1 byte (8 bits) on most systems. However, the exact size depends on the platform, though it is usually 1 byte.
- A **char** variable can store a single character (such as 'A', 'b', '1', or '!'). It is in single quotations. For example;

```
char letter = 'A';  // Stores the character 'A'
```

- Internally, **char** is represented as an integer value based on its position in the ASCII table (or other character encoding like UTF-8). For example, the character 'A' has an integer value of 65 according to the ASCII table.

```
char letter = 'A';
std::cout << (int)letter;  // Outputs: 65
```

- **Char** can be signed or unsigned depending on the system and compiler. A **signed char** can represent values from -128 to 127 and an **unsigned char** can represent values from 0 to 255.

wchar_t

In computer programming, particularly in C and C++, the **wchar_t** data type is used to represent wide characters, which are essential for handling text in internationalized applications.

In C++,`w_char` plays a crucial role in enabling systems to handle non-ASCII characters, such as those found in languages like Chinese, Japanese, Arabic, and Cyrillic.

Unlike char, which is usually 1 byte (8 bits) in size, `wchar_t` is typically 2 bytes (16 bits) or 4 bytes (32 bits) depending on the system and the platform, and it is designed to handle larger character sets, such as Unicode, which includes thousands of characters for many languages.

0.1.2 Integers

In C++, an integer is a data type that is used to represent whole numbers, both positive and negative, without decimal points. The **int** data type is one of the most commonly used types for integer numbers in C++. Example:

```
#include <iostream>
int main() {
    int a = 42;  // An integer
    std::cout << a << std::endl;  // Output: 42
    return 0;
}
```

Properties

- The size of an int can vary depending on the system, but it is typically 4 bytes (32 bits) on most modern systems.
- In most systems, int can store values ranging from -2,147,483,648 to 2,147,483,647.
- The range of integer types can depend on whether they are signed or unsigned and the system architecture (32-bit, 64-bit).

  Signed int, can represent both positive and negative values. **Unsigned int**, can only represent positive integers and zero, effectively doubling the upper range of the positive numbers.
- In C++, an integer variable should always be initialized before it is used. If you don't initialize an integer variable, its value is undefined. Example:

```
int a;  // Uninitialized integer (value is undefined)
std::cout << a << std::endl;  // This will print an
    undefined value.
```

- C++ allows various operations on integers, including arithmetic, relational, and bitwise operations.

Integer Types in C++

- **Short**: A smaller integer type (usually 2 bytes) with a smaller range. Short typically ranges from -32,768 to 32,767.
- **Long**: A larger integer type (usually 4 bytes or more) with a larger range than int. Long typically ranges from -2,147,483,648 to 2,147,483,647 (or larger on 64-bit systems).
- **Long long**: An even larger integer type (usually 8 bytes) for extremely large numbers. Long long typically ranges from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.
- **Fixed-Width Integer Types** (from cstdint library)**:** The fixed-width integer types guarantee the exact number of bits used for the integer representation, which is important for applications requiring precise control over data sizes, like systems programming, embedded systems, or protocols. These types include:

Type	Size (in bits)	Range (signed)
`int8_t`	8	-128 to 127
`uint8_t`	8	0 to 255
`int16_t`	16	-32,768 to 32,767
`uint16_t`	16	0 to 65,535
`int32_t`	32	-2,147,483,648 to 2,147,483,647
`uint32_t`	32	0 to 4,294,967,295
`int64_t`	64	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
`uint64_t`	64	0 to 18,446,744,073,709,551,615

0.1.3 Floats

The float data type in C++ is used to represent floating-point numbers, which are numbers that have decimal points. Floating-point types are an essential feature of most programming languages, including C++, as they allow for the representation of real numbers.

Properties

- A float in C++ typically takes up 4 bytes (32 bits) of memory. This size allows for the storage of floating-point numbers with significant precision but limits the range of values that can be represented.
- A float typically offers around 6-7 decimal digits of precision. This means that any number with more than 7 digits after the decimal point may lose precision when stored in a float variable. For example, the value 3.141592653589793238 can only be approximated to 3.141593 when stored as a float.

  Since a float can store a maximum of 6-7 significant digits, any value with more precision than that will lose information when stored. This can result in rounding errors when performing arithmetic operations.

Syntax and Usage of float

The syntax for declaring a float variable is similar to other types in C++:

```
1    float variable_name;
```

Alternatively, a float variable can be initialized directly:

```
1    float pi = 3.14159;
```

Comparison with Other Floating-Point Types

In C++, besides float, there are two other primary floating-point types: double and long double. The main differences between these types are in their precision and storage requirements.

- **Precision:** double provides double-precision (64-bit) floating-point numbers, which offer more precision and a larger range than float. A double can store around 15-16 decimal digits of precision, compared to 6-7 digits for float.
- **Size:** A double typically occupies 8 bytes (64 bits) of memory, double the size of a float.

When to Use Float

While float is not as precise as double or long double, it offers advantages in terms of memory usage and performance. It is commonly used in situations where the precision provided by float is sufficient and memory constraints or performance are important.

Float is frequently used for storing values such as coordinates, colors, and transformation matrices in 3D graphics due to its smaller size and sufficient precision.

It is also used in embedded systems with limited memory; using float helps save space and ensure faster calculations.

0.1.4 Double

The double data type in C++ is a fundamental floating-point type used for representing real numbers that require greater precision and range compared to the float type.

Properties

- Double typically occupies 8 bytes (64 bits) in memory.
- A double provides about 15-16 decimal digits of precision. This higher precision is important in applications where small differences in values are significant, such as scientific calculations and financial modeling.
- While double provides higher precision, it comes with a tradeoff in terms of memory usage and computational performance. double requires more memory (8 bytes) and can be slower to compute compared to float (4 bytes), especially in resource-constrained environments. In applications where memory or performance is critical, the float type may be preferred.

Syntax and Usage of double

In C++, the double data type can be declared similarly to other types. It can be used to store real numbers, and it is often employed when higher precision is needed.

```cpp
double variable_name;      // Declaration without initialization
double pi = 3.141592653589793;  // Declaration with initialization
```

Fixed-Width Floating-Point Types: `float(N)_t`

In addition to the standard float type, C++ provides fixed-width floating-point types such as `float32_t` and `float64_t,` which ensure that floating-point numbers have a specific size and behavior, regardless of the platform.

When to Use double

The double data type is commonly used when higher precision than float is needed, but the extra precision provided by long double is not required. It is suitable for applications where slight rounding errors could affect the results.

In fields such as physics, chemistry, and biology, precise measurements are essential, and double is often used for storing constants, results of calculations, and intermediate values.

Although double is not always ideal for exact decimal representation (due to rounding errors), it is often used in financial calculations that involve floating-point operations.

In engineering, double is used for simulations, calculations, and modeling where precision is important, but the number of significant digits does not exceed the capabilities of double.

0.2 Conclusion

Example usage of c++ variables is given below:

```cpp
#include <iostream>
#include <limits>
#include <iomanip>

int main() {
    std :: cout << std::setw(15) << std::left<< "
       ---------------------";
    std :: cout << std::setw(15) << std::right<< "
       ---------------------" << std::endl;
    std :: cout << std::setw(15) << std::left<< "Type";
```

```
9     std :: cout << std::setw(3) << std::left<< " ";
10    std :: cout << std::setw(15) << std::right<< "Size (byte)
          " << std::endl;
11    std :: cout << std::setw(15) << std::left<< "
          ---------------------";
12    std :: cout << std::setw(15) << std::right<< "
          ---------------------" << std::endl;
13
14    std :: cout << std::setw(15) << std:: left<< "int";
15    std :: cout << std::setw(3) << std::left<< " ";
16    std :: cout << std::setw(10) << std::right<< sizeof(int)
          << std::endl;
17    std :: cout << std::setw(15) << std:: left<< "long";
18    std :: cout << std::setw(3) << std::left<< " ";
19    std :: cout << std::setw(10) << std::right<< sizeof(long)
           << std::endl;
20    std :: cout << std::setw(15) << std:: left<< "long long";
21    std :: cout << std::setw(3) << std::left<< " ";
22    std :: cout << std::setw(10) << std::right<< sizeof(long
          long) << std::endl;
23    std :: cout << std::setw(15) << std:: left<< "long int";
24    std :: cout << std::setw(3) << std::left<< " ";
25    std :: cout << std::setw(10) << std::right<< sizeof(long
          int) << std::endl;
26    std :: cout << std::setw(15) << std:: left<< "long long
          int";
27    std :: cout << std::setw(3) << std::left<< " ";
28    std :: cout << std::setw(10) << std::right<< sizeof(long
          long int) << std::endl;
29    std :: cout << std::setw(15) << std:: left<< "short";
30    std :: cout << std::setw(3) << std::left<< " ";
31    std :: cout << std::setw(10) << std::right<< sizeof(short
          ) << std::endl;
32    std :: cout << std::setw(15) << std:: left<< "short int";
33    std :: cout << std::setw(3) << std::left<< " ";
34    std :: cout << std::setw(10) << std::right<< sizeof(short
           int) << std::endl;
35    std :: cout << std::setw(15) << std:: left<< "unsigned
          int";
36    std :: cout << std::setw(3) << std::left<< " ";
37    std :: cout << std::setw(10) << std::right<< sizeof(
          unsigned int) << std::endl;
38    std :: cout << std::setw(15) << std:: left<< "float";
39    std :: cout << std::setw(3) << std::left<< " ";
```

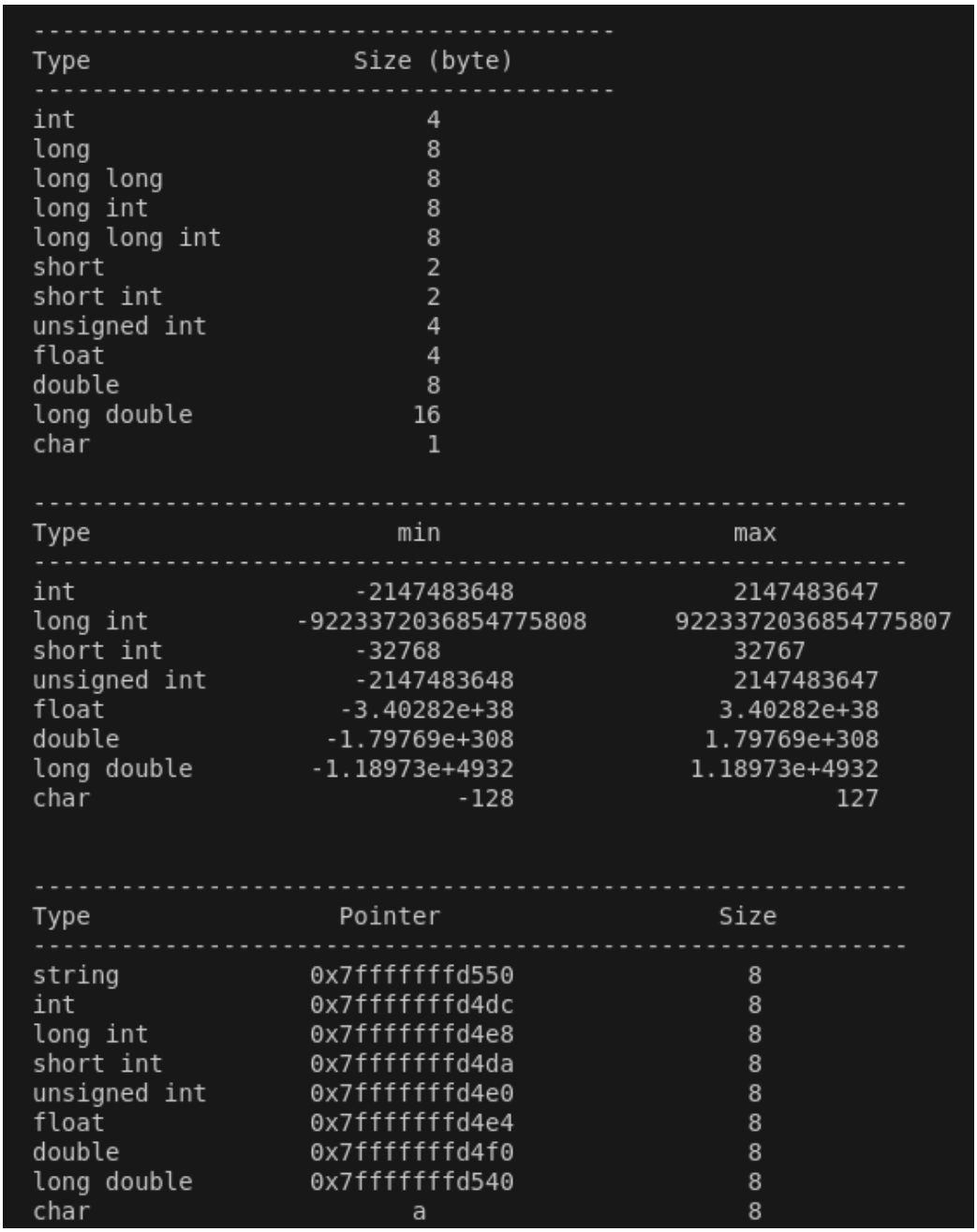
```
40      std :: cout << std::setw(10) << std::right<< sizeof(float
            ) << std::endl;
41      std :: cout << std::setw(15) << std:: left<< "double";
42      std :: cout << std::setw(3) << std::left<< " ";
43      std :: cout << std::setw(10) << std::right<< sizeof(
            double) << std::endl;
44      std :: cout << std::setw(15) << std:: left<< "long double
            ";
45      std :: cout << std::setw(3) << std::left<< " ";
46      std :: cout << std::setw(10) << std::right<< sizeof(long
            double) << std::endl;
47      std :: cout << std::setw(15) << std:: left<< "char";
48      std :: cout << std::setw(3) << std::left<< " ";
49      std :: cout << std::setw(10) << std::right<< int(sizeof(
            char)) << std::endl << std::endl;
50
51
52
53      std :: cout << std::setw(15) << std::left<< "
            ------------------------------";
54      std :: cout << std::setw(15) << std::right<< "
            ------------------------------" << std::endl;
55      std :: cout << std::setw(15) << std::left<< "Type";
56      std :: cout << std::setw(3) << std::left<< " ";
57      std :: cout << std::setw(10) << std::right<< "min";
58      std :: cout << std::setw(3) << std::left<< " ";
59      std :: cout << std::setw(20) << std:: right << "max" <<
            std::endl;
60      std :: cout << std::setw(15) << std::left<< "
            ------------------------------";
61      std :: cout << std::setw(15) << std::right<< "
            ------------------------------" << std::endl;
62      std :: cout << std::setw(15) << std::left<< "int";
63      std :: cout << std::setw(3) << std::left << " ";
64      std :: cout << std::setw(15) << std::right << std::
            numeric_limits<int>::min();
65      std :: cout << std::setw(25) << std::right << std::
            numeric_limits<int>::max() << std::endl;
66      std :: cout << std::setw(15) << std::left<< "long int";
67      std :: cout << std::setw(3) << std::left << " ";
68      std :: cout << std::setw(15) << std::right << std::
            numeric_limits<long int>::min();
69      std :: cout << std::setw(25) << std::right << std::
            numeric_limits<long int>::max() << std::endl;
70      std :: cout << std::setw(15) << std::left<< "short int";
```

```
71      std :: cout << std::setw(3) << std::left << " ";
72      std :: cout << std::setw(10) << std::right << std::
           numeric_limits<short int>::min();
73      std :: cout << std::setw(25) << std::right << std::
           numeric_limits<short int>::max() << std::endl;
74      std :: cout << std::setw(15) << std::left<< "unsigned int
           ";
75      std :: cout << std::setw(3) << std::left << " ";
76      std :: cout << std::setw(15) << std::right << std::
           numeric_limits<int>::min();
77      std :: cout << std::setw(25) << std::right << std::
           numeric_limits<int>::max() << std::endl;
78      std :: cout << std::setw(15) << std::left<< "float";
79      std :: cout << std::setw(3) << std::left << " ";
80      std :: cout << std::setw(15) << std::right << std::
           numeric_limits<float>::lowest();
81      std :: cout << std::setw(25) << std::right << std::
           numeric_limits<float>::max() << std::endl;
82      std :: cout << std::setw(15) << std::left<< "double";
83      std :: cout << std::setw(3) << std::left << " ";
84      std :: cout << std::setw(15) << std::right << std::
           numeric_limits<double>::lowest();
85      std :: cout << std::setw(25) << std::right << std::
           numeric_limits<double>::max() << std::endl;
86      std :: cout << std::setw(15) << std::left<< "long double"
           ;
87      std :: cout << std::setw(3) << std::left << " ";
88      std :: cout << std::setw(15) << std::right << std::
           numeric_limits<long double>::lowest();
89      std :: cout << std::setw(25) << std::right << std::
           numeric_limits<long double>::max() << std::endl;
90      std :: cout << std::setw(15) << std::left<< "char";
91      std :: cout << std::setw(3) << std::left << " ";
92      std :: cout << std::setw(15) << std::right << int(std::
           numeric_limits<char>::min());
93      std :: cout << std::setw(25) << std::right << int(std::
           numeric_limits<char>::max()) << std::endl<< std::endl
           << std::endl;
94
95
96
97      std :: cout << std::setw(15) << std::left<< "
           ------------------------------";
98      std :: cout << std::setw(15) << std::right<< "
           ------------------------------" << std::endl;
```

```
99      std :: cout << std::setw(15) << std::left<< "Type";
100     std :: cout << std::setw(3) << std::left<< " ";
101     std :: cout << std::setw(10) << std::right<< "Pointer";
102     std :: cout << std::setw(3) << std::left<< " ";
103     std :: cout << std::setw(20) << std:: right << "Size" <<
           std::endl;
104     std :: cout << std::setw(15) << std::left<< "
           ------------------------------";
105     std :: cout << std::setw(15) << std::right<< "
           ------------------------------" << std::endl;
106
107
108     std::string string = "Hello World!";
109     std::string *pStr = &string;
110     std::cout << std::setw(15) << std::left << "string";
111     std::cout << std::setw(3) << std::left << " ";
112     std::cout << std::setw(15) << std::right << pStr;
113     std::cout << std::setw(17) << std::right << sizeof(pStr)
           << std::endl;
114
115     int integer = 5;
116     int *pInt = &integer;
117     std::cout << std::setw(15) << std::left << "int";
118     std::cout << std::setw(3) << std::left << " ";
119     std::cout << std::setw(15) << std::right << pInt;
120     std::cout << std::setw(17) << std::right << sizeof(pInt)
           << std::endl;
121     long int longInt = 2;
122     long int *pLongInt = &longInt;
123     std::cout << std::setw(15) << std::left << "long int";
124     std::cout << std::setw(3) << std::left << " ";
125     std::cout << std::setw(15) << std::right << pLongInt;
126     std::cout << std::setw(17) << std::right << sizeof(
           pLongInt) << std::endl;
127     short int shortInt = 3;
128     short int *pShortInt = &shortInt;
129     std::cout << std::setw(15) << std::left << "short int";
130     std::cout << std::setw(3) << std::left << " ";
131     std::cout << std::setw(15) << std::right << pShortInt;
132     std::cout << std::setw(17) << std::right << sizeof(
           pShortInt) << std::endl;
133     unsigned int unsignedInt = 4;
134     unsigned int *pUnsignedInt = &unsignedInt;
135     std::cout << std::setw(15) << std::left << "unsigned int"
           ;
```

```
136     std::cout << std::setw(3) << std::left << " ";
137     std::cout << std::setw(15) << std::right << pUnsignedInt;
138     std::cout << std::setw(17) << std::right << sizeof(
            pUnsignedInt) << std::endl;
139     float floatVar = 5.5;
140     float *pFloat = &floatVar;
141     std::cout << std::setw(15) << std::left << "float";
142     std::cout << std::setw(3) << std::left << " ";
143     std::cout << std::setw(15) << std::right << pFloat;
144     std::cout << std::setw(17) << std::right << sizeof(pFloat
            ) << std::endl;
145     double doubleVar = 6.6;
146     double *pDouble = &doubleVar;
147     std::cout << std::setw(15) << std::left << "double";
148     std::cout << std::setw(3) << std::left << " ";
149     std::cout << std::setw(15) << std::right << pDouble;
150     std::cout << std::setw(17) << std::right << sizeof(
            pDouble) << std::endl;
151     long double longDoubleVar = 7.7;
152     long double *pLongDouble = &longDoubleVar;
153     std::cout << std::setw(15) << std::left << "long double";
154     std::cout << std::setw(3) << std::left << " ";
155     std::cout << std::setw(15) << std::right << pLongDouble;
156     std::cout << std::setw(17) << std::right << sizeof(
            pLongDouble) << std::endl;
157     char charVar = 'a';
158     char *pChar = &charVar;
159     std::cout << std::setw(15) << std::left << "char";
160     std::cout << std::setw(3) << std::left << " ";
161     std::cout << std::setw(10) << std::right << pChar;
162     std::cout << std::setw(23) << std::right << sizeof(pChar)
            << std::endl;
163     return 0;
164 }
```

Output:



```
---------------------------------------
Type                  Size (byte)
---------------------------------------
int                           4
long                          8
long long                     8
long int                      8
long long int                 8
short                         2
short int                     2
unsigned int                  4
float                         4
double                        8
long double                  16
char                          1

------------------------------------------------------------
Type                        min                        max
------------------------------------------------------------
int                     -2147483648                2147483647
long int          -9223372036854775808       9223372036854775807
short int             -32768                     32767
unsigned int          -2147483648                2147483647
float                -3.40282e+38               3.40282e+38
double              -1.79769e+308              1.79769e+308
long double         -1.18973e+4932             1.18973e+4932
char                          -128                       127


------------------------------------------------------------
Type                   Pointer                    Size
------------------------------------------------------------
string              0x7fffffffd550                8
int                 0x7fffffffd4dc                8
long int            0x7fffffffd4e8                8
short int           0x7fffffffd4da                8
unsigned int        0x7fffffffd4e0                8
float               0x7fffffffd4e4                8
double              0x7fffffffd4f0                8
long double         0x7fffffffd540                8
char                         a                    8
```

Figure 1: Sizes of Variables and Pointers