**Simple I/O. Data Types**

1. **Output**

std::cout << "Hello World!\n";

* cout is the “**c**haracter **out**put stream”.
* << is an operator, meaning **insertion operator**
* We can chain an output

std::cout << "Hello, I am " << age << " years old\n";

* To **specify decimal points**

cout.setf(ios::fixed);

cout.setf(ios::showpoint); 🡪 any cout statements after will output 2 decimal points

cout.precision(2); (only need last line if we want to change later)

1. **Input (skips all blank characters)**

std::cout << "Enter your account and password (newline/space each time): ";

std::cin >> account >> password;

Where cin stands for character input. The >> operator is the **get from 🡪** specifies where the input goes.

1. **Namespaces**

We know that in order to use cout or cin, we include std::cout or std::cin. When names such as **cin and cout were defined in iostream, their definitions said they were in std namespace**.

- C++ has namespace because sometimes two or more items have same name 🡪 C++ divides them into collections, so no 2 items in the same collection (same namespace) will have the same name.

- Note that a namespace also divides up all C++ name specifications into collections such that each name in a namespace has **only one specification** (one definition) in that namespace.

🡪 A namespace divides up the name, but it also takes a lot of C++ codes along with it.

\*\* This means that we can use 2 items with the same name from different namespaces 🡪 this is why we include the namespace:: before the names.

However, for simple C++ programs, we can safely declare a namespace at the beginning

using namespace std;

1. **Escape Character**

- Raw string literals, meaning disregarding escape characters: cout << R”(c:\files\)”

- To insert a newline, we can also use << endl <<

1. **Data Types \*\***

You can specify exponents using the e notation: 2.34e2 ~ 234

**Some Number Types**

Graphical user interface

Description automatically generated with low confidence

*Precision means the number of meaningful digits (significant figures)*

Note that for these number types, the size (memory used) **varies from one machine to another** (on 32-bit machines, int are 4 bytes, but on 64-bit machines, int are 8 bytes).

**C++11 Types**

🡪 New integer types were added to C++11 that specify exactly the size and whether or not the data type is signed or unsigned.

These types are accessible by including <cstdint>.

Table

Description automatically generated with medium confidence

OR we can do auto x = expression;

OR we can do decltype *<variable\_with\_known\_type> <new\_variable>*;

**The type char**

- 'A' and "A" mean different things, one can be stored in a variable of type *char*, one is a string.

- char will only store 1 character, regardless of initialization (avoid, else overflow) or input.

**The *string* class**

To use the string class we must first include the string library

#include <string>

If we are using namespace std, we don’t need std::string but can use string.

When you use cin to read input into a string variable, the computer only reads until it encounters a **whitespace** character 🡪 Cannot input strings with spaces

Arithmetic operations

Regardless of +-\*/, the rule follows:

int int 🡪 int 7/2 = 3

int float 🡪 float

float float 🡪

**Random Number Generator**

#include <cstdlib>

#include <ctime>

srand(time(NULL)); (called only 1 time, because the code will run faster than time passes

int dice = (rand() % 6) + 1; 🡪 same seed generated for the code)

**Type Cast**

static\_cast <double> (*variable*)

OR double (*variable*) (but may be discontinued in later versions)