DATATYPES

Prerequisites:

1. Basic structure of a C++ program

PYTHON: STORING DATA IN VARIABLES

In Python what were the two types of data we dealt with? One was numbers and the other was strings(which is basically nothing but a sequence of characters enclosed by ‘’ or “”).
In other words there were two types of data, numeric and alphanumeric.

How did we input a string and stored it in a variable?
We wrote something like this:
a = input()
How did we input a number and stored it in a variable?
A = int(input())
The int() just converted the string inputted to a number(integer).

And later on when we wrote:
a=A
a would store an integer instead of a string(the integer would overwrite the string stored in ‘a’ initially).

You can see that when storing something in a variable in Python we don’t need to decide before hand what type of data will that variable(container) store.
We can just put anything, numeric or alphanumeric.

C++ : STORING DATA IN VARIABLES

But in C++ it turns out that we need to to tell C++ beforehand what we want our variable to store(whether it is numeric or alphanumeric).

And once we say that our variable ‘a’ stores a string it can never be used to store an integer in any part of the program after that. Only we can overwrite it with integers. This is unlike Python.[We saw above how ‘a’ containing a string was overwritten with an integer later. But C++ won’t let us do this.]

DECLARING VARIABLES IN C++:

Let us see how to instruct C++ that we want our variable ‘a’ to store a string.
We simply write:
string a;

Note:
***Remember that semicolon

How do we instruct C++ that we want variable A to store integers?
Simply write:
int A;

This process of instructing C++ that I want to store an integer in A or I want to store a string in variable a is called declaration.

Note:
****In C++ strings are only enclosed by “”(double quotes) and not ‘’(single quotes), unlike Python.

INITIALIZING VARIABLES IN C++

If I want to assign the variables with some values we say that we are initializing that variable with a value.
When we write:
A=5;
we say that int A is initialized to 5. Similarly when we write:
a = “hello”;
we say string a is initialized with “hello”.

DECLARING AND INITIALIZING VARIABLES IN C++ SIMULTANEOUSLY:
We can also write something like: int A=5;
When we write the above statement we say that variable A is declared as an int as well as initialized to 5.

READ BEFORE WRITES CALLS FOR UNDEFINED BEHAVIOUR IN C++ FOR INTEGER DATA TYPES :
Now, what if we don’t initialize int A? Then if we try to read(print) A, what happens? We get undefined behaviour. In other words no writes before read is undefined.

Note: Undefined behaviour(U.B) is the result of executing computer code whose behavior is not prescribed by the language specification to which the code adheres, for the current state of the program. This happens when the translator of the source code makes certain assumptions, but these assumptions are not satisfied during execution.
READ BEFORE WRITES IS DEFINED IN C++ FOR STRING DATA TYPES :

String a on the other hand (even if we don't initialize it to anything) automatically gets initialized to an empty string.

SO MANY DATA TYPES IN C++!:

Data can be of two types: numeric(int in Python) and alphanumeric(string in Python).
In C++ numeric data types are classified into many types. Apart from int there are many other data types as well which store numbers.

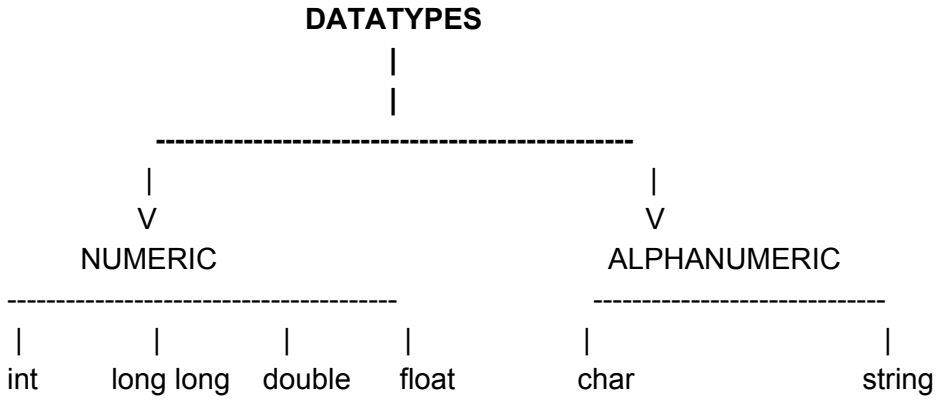
Similarly alphanumeric data are classified into two types. Apart from string as in Python there is one more data type which can store characters. But this datatype can store no more than one character.

NEED FOR HAVING SO MANY DATA TYPES IN C++:

Why do we need all this classification and what to use when? We could do very well with only numeric(int) and alphanumeric(string) data types.
It turns out that these different data types require different amount of memory during storage. So appropriate usage of data types will help us to save memory which can be used for something else in our program. Unnecessary usage of memory would also slow our program to a certain extent. After all in competitive programming we fight against time as well as memory. Each program allows you to use a limited amount of memory. So it is important for a programmer to understand the available data types well and use these efficiently.

WHAT ARE THE DATA TYPES IN C++? :

Note: We won't be covering all the available data types but will cover only the ones that are mostly used.
Let's draw a chart where you will get to know the names of different data types available and what they store, numbers or characters?



NUMERIC DATA TYPES:

1. int- When you declare a variable to be int, it reserves 4 bytes of memory in the computer's memory to store a number. int data type can be of two types:

A. signed int- stores values in the range: -2147483648 to 2147483647
B. unsigned int stores values in the range: 0 to 4294967295.
The ranges indicate that in your program if you want to store values in that range in a variable, it is most appropriate to declare that variable to be of this data type. Most of the time we won't be using unsigned int. Let us first see how to declare a variable of type unsigned int:
unsigned int A;
Let us now see how to declare a variable of type signed int:
int A;
OR
signed int A;
When you write only int A, C++ takes it as signed int. In most of the situations you will find that you are using this data type: signed int. unsigned int is almost used never however it is good to know about it.

2. long long - When you declare a variable of type long long the computer reserves 8 bytes of memory to store a number. Again they are of two types:
unsigned long long : 0 to 18,446,744,073,709,551,615
signed long long: -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
Like unsigned int , unsigned long long is also not used very often. We declare unsigned long long as follows:
unsigned long long A;
We declare a variable of type signed long long as follows:
signed long long A;
OR simply,
long long A;
Most of the time we will be using signed long long in our programs. long long is also sometimes know as long int. You can declare variables as unsigned long long also as follows:
unsigned long int A;
You can declare variables as signed long long also as follows:
signed long int B;
OR
long int C;
long int and long long are essentially the same thing. We will most of the time call this data type long long.

Note: The above two data types can only store whole numbers and not numbers with decimal points. This is unlike Python where int could store decimals as well.

To store numbers having decimal points, we have two other data types called double and float:

1. float :
Range: -3.4*10^-38 to 3.4*10^38(7 significant digits)
Size: 4 bytes

2. Double:
   Range: -1.7*10^-308 to 1.7*10^308(15 significant digits)
   Size: 8 bytes

Note: In float, by 7 significant digits it means that you can store something like: 1.2234567e+38 in a variable declared as float, but you cannot store something like: 1.223456782e+38. If you try storing 1.223456782e+38, the variable will end up storing something where the 8th and 9th digits after the decimal point are random, like 1.223456741e+38(this leads to overflow of values, covered next). It means that after the decimal point, float can keep track of maximum 7 digits correctly. This is the meaning of 7 significant digits. You can treat e+38 as 10^38(10 to the power 38). Similarly, in double you can store correct up to 15 significant digits i.e after the decimal point, double can keep track of maximum 15 digits correctly.
****double A = 1.456789e+305; is a valid syntax!
**Actually float can store correct upto 7-9 significant digits. The number of significant digits it can store depends on certain other things which is out of the scope of this tutorial. Similarly double can store correct upto 15-17 significant digits. But on an average we take it that float stores correct up to 7 significant digits whereas double stores correct up to 15 significant digits.
****So the golden rule is whenever you have to store numbers that have decimal points use double or float! Most of the time we use double.

OVERFLOW OF VALUES:

Great! Now have you wondered that what if we store some number in a particular data type variable which is not in the range permitted by that data type. Try it out yourself. Declare a variable as int and store in it a 10 digit number. Print the variable what do you find? You will find some random number(in the range permitted by int data type)! What had actually happened? Since you stored a number out of range, the variable could not hold that value and as a result it lead to an ‘overflow’.

ALPHA NUMERIC DATA TYPES:

Let us come to the alpha numeric part. Strings in C++ are as they were in Python. They are a sequence of characters enclosed only by “”(double quotes) and not by ‘’(single quotes).
We declare and initialize a variable to be of type string in the following way:
string A=”hello”;

What is char? char is short for Character. It is another data type. It is capable of storing only a single character enclosed by ‘’.
For example:
char a = ‘X’;
When you declare a variable as char the computer reserves 2 bytes of memory for it in the memory. But, remember char can store exactly 1 character enclosed by ‘’.

INPUTTING DATA IN C++:

You must be wondering now, how to input values of a particular data type. Well this is something where C++ comes handy. You just need to tell C++ that my variable is of this data type and then you can use the standard way of inputting using cin. We will cover in detail how to input values of various data types in the tutorial on I/O.
For now remember, all types of data are inputted in the same way(using same syntax :))!

THAT DATA TYPE WHICH IS NOT NUMERIC OR ALPHANUMERIC:

Another data type called bool is available in C++. This data type can take only two values, 'true' or 'false'.(the quotes are used only for clarity) This data type is very useful to keep track of situations involving binary values(yes/no,on/off). Many inbuilt C++ functions return this data type. In many cases you will find we use int to keep track of such situations as well.(int taking values 0 or 1. 0 indicating false while 1 indicating true).But bool data type reserves only 1 byte of memory where as int reserves 4 bytes of memory! So it is always preferred to use bool when dealing with binary valued quantities.

ARRAYS

MOTIVATION:
Let's suppose you are taking a survey of 100 people and you are required to input and store the age of each of the persons. Going by our present knowledge we would create 100 variables(to store the ages) and then inputted the 100 ages.
Our program would be something like:

```
int age1;
int age2;
int age3;
.
.
.
.
.
.
.
.
int age100;
cin>>age1;
cin>>age2;
cin>>age3;
.

.
.
```

.
.
cin>>age100;

***Don’t worry if you don’t understand cin. We will cover this in the tutorial on I/O. For the time being think of it as a way of inputting age1,age2,... and so on.

Okay I did not finish the whole program! But you can understand easily that it would have been long.
Let me know change my decision and tell you to do a survey of 101 people and input their age. What would you do? You would have to rewrite the whole program again. You can clearly see that this does not generalise well.
Here comes the utility of arrays.
ARRAYS IN C++ :
Please refer to the following document to learn about arrays.
**** Please don’t look at **Example: C++ Array** and **How to insert and print array elements?** in the article as it has I/O and loops which we are about to cover in future tutorials. I would strongly urge you to refer back to these subparts once you have read the article on I/O and Control Flow(which will cover loops).

https://www.programiz.com/cpp-programming/arrays

DIFFERENCE BETWEEN PYTHON LISTS AND C++ ARRAYS:

In Python, remember we learnt about lists? What is the difference between lists and these arrays.
Well you may be thinking that these arrays are of fixed size. But hang on! We will learn about resizeable arrays a bit later. Any other notable difference you can spot?
If you observe carefully, in an array we can have only values of the same data type but in lists(in Python) we could append anything(number, string, other lists etc).