CS101 Short Notes

Aditya Byju

Course Professor: Prof. Bhaskaran Raman
& Prof. Kameswari Chebrolu
Ref: Prof's slides
There is an easy way and a hard way
The hard part is finding the easy way

Computer Programming and Utilization

September 2021

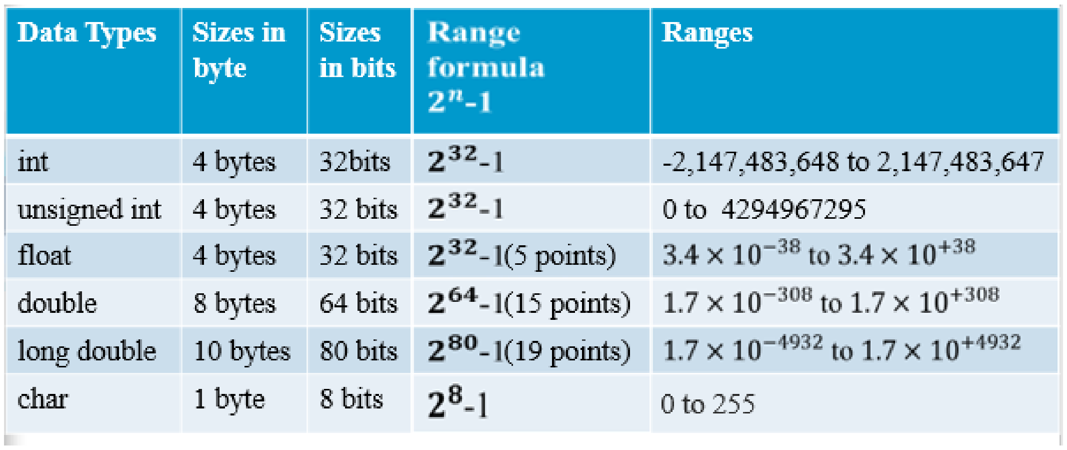
Contents

Basic C++

- Linux is open source and is developed by Linux community of developers. Unix was developed by AT&T Bell labs and is not open source. Linux is free to use. Unix is licensed OS.
- Why use Unix command-line?
  - Windows GUI: uses pre-programmed interface - set of possible actions are pre-decided
  - Command-line shell: a prog. (scripting) language - we can use pre-written programs and compose new scripts
- Some commands:
  - ls: list files or directories
  - ls -a: list all files or directories (hidden too)
  - pwd: prints current working directory
  - cd: change directory
  - mkdir: make directory
  - rmdir: remove directory
  - cat: show content of given filename
  - less: show content of given filename in a scrollable format
  - cp: copy file from source to destination
  - mv: move file from source to destination
  - rm: remove files, directories and links
  - emacs: text editor
  - ps: to list the processes
  - kill: to kill a process
  - tar: to zip files and folders
  - grep: used for finding patterns within files
  - man: manual of a particular command
- The Unix File System:
  - Three kinds of permissions: read, write, execute
  - Three levels of access control: user, group, all
- program - a precise description of the calculations we want the computer to perform
- The C++ programming language was designed by Bjarne Stroustrup, in the 1980s. It evolved out of C programming language.
- syntax - grammatical rules indicating how commands must be written
- control flow - the order in which statements get executed
- ASCII - American Standard Code for Information Interchange. ‘a’= 97, ‘b’= 98, . . .
- algorithm - a precise description of the operations needed to solve a problem
- Single precision float (32 bits):
  - 1 bit for sign of significand/mantissa

- 23 bits for mantissa itself
- 8 bits for exponent (signed)

- Double precision float (64 bits):
  - 1 bit for sign of significand/mantissa
  - 52 bits for mantissa itself
  - 11 bits for exponent (signed)
- Signed numbers are represented by two's complement
- Real numbers are represented by IEEE 754 standard (float, double)



Data Types	Sizes in byte	Sizes in bits	Range formula 2^n-1	Ranges
int	4 bytes	32bits	2^{32}-1	-2,147,483,648 to 2,147,483,647
unsigned int	4 bytes	32 bits	2^{32}-1	0 to 4294967295
float	4 bytes	32 bits	2^{32}-1(5 points)	3.4×10^{-38} to $3.4 \times 10^{+38}$
double	8 bytes	64 bits	2^{64}-1(15 points)	1.7×10^{-308} to $1.7 \times 10^{+308}$
long double	10 bytes	80 bits	2^{80}-1(19 points)	1.7×10^{-4932} to $1.7 \times 10^{+4932}$
char	1 byte	8 bits	2^{8}-1	0 to 255

Figure : Variable Types and Ranges

- Compiler translates the C++ program to a Machine language program
- To print newline, use endl
- Multiplication, division have same precedence. Addition, subtraction have same precedence. Operators of same precedence will be evaluated left to right.
- Initialization happens left to right
- x % y evaluates to the remainder when x is divided by y. x, y must be integer expressions.
- Code inside {} is called a block
- All variables defined in a block are destroyed every time control reaches the end of the block
- Variables defined outside a block can be used inside the block, if no variable of the same name is defined inside the block. If a variable of the same name is defined, then from the point of definition to the end of the block, the newly defined variable gets used. The new variable is said to "shadow" the old variable. The region of the program where a variable defined in a particular definition can be used is said to be the scope of the definition.
- Variables are regions of memory which can store values
- 'consequent' is a C++ statement
- Flowchart is a pictorial representation of a program. Diamond shaped boxes are used for condition checks.

- AND: &&
- OR: ||
- NOT: !
- ! has higher precedence than && which has higher precedence than ||
- C++ has data type bool into which values of conditions can be stored
- If break appears inside a while statement which is itself nested inside another while, then then the inner statement is terminated. The continue statement enables us to skip the rest of the iteration.
- Form of for statement: for(initialization; condition; update) body
- In the case of for statement new variables can be defined in initialization. These variables are accessible inside the loop body, including condition and update, but not outside. The variable which is initialized is called the control variable of the for statement. Leaving the initialization part of the for statement empty is allowed.
- m, n, have the same common divisors as m-n, n. The largest divisor of m,n is also the largest divisor of m-n,n. To find GCD(m,n), we might as well find GCD(n, m-n).
- Euclid's theorem: Let $m, n > 0$ be positive integers. If n divides m then GCD(m,n) = n. Otherwise GCD(m,n) = GCD(m%n, n).
- In Taylor series error using first k terms $\leq |(k+1)^{th}$ term$|$
- In mathematics, the bisection method is a root-finding method that applies to any continuous function for which one knows two values with opposite signs.
- In order to perform numerical integration we just need to be able to evaluate the given function at an arbitrary point.
- The Newton-Raphson method begins with an initial estimate of the root, denoted $x_0 \neq x_r$, and uses the tangent of f(x) at x0 to improve on the estimate of the root. In particular, the improvement, denoted x1, is obtained from determining where the line tangent to f(x) at x0 crosses the x-axis. This represents a single iteration of the Newton-Raphson method
- A number is said to be perfect if it is the sum of all its divisors smaller than itself
- Area allocated in memory where a function will have its variables is called the activation frame of that function
- Calling program will refer to the variables in its activation frame. Called program will refer to the variables in its activation frame.
- Function definition must appear before any call to it in the program file
- Main program is also a function
- If a function does not return a value its return type is void
- Function is a piece of code which takes the responsibility of getting something done
- Preconditions are certain properties that the arguments of a function satisfy. What is promised about the state of the computer after the function finishes execution is called postcondition.
- Variable names can be the same in the main program and in the function

- A function is like an independent program except that it gets some values (arguments) from the calling program, and returns results to the calling program
- & before the name of the parameter does not allocate space for the parameter, but instead just uses the variable from the calling program. Such parameters are called reference parameters. If a certain parameter is a reference parameter, then the corresponding argument is said to be "passed by reference".
- If we want to return more than one result we can do so by using a reference parameter.
- A pointer is a variable that can store addresses
- The operator & can be used to get the address of a variable. The same & is used to mark reference parameters; but the meaning will be clear from the context.
- Customarily, addresses get printed in hexadecimal radix, i.e. they will consist of a sequence of hexadecimal digits prefixed by "0x"
- Variable for storing addresses of variables of type int : int *v;
- The * is not to be read as multiplication. Think of it as (int*) v; where int* means the type: "address of int".
- In general, to create a variable w to store addresses of variables of type T, write: T* w;
- Recursion is the phenomenon of a function calling itself. There is no circularity if the arguments to the new call are different from the arguments in the original call. Each call executes in its own activation frame.
- Recursion often produces compact, elegant programs. Recursive programs might be slightly slower because they need to create activation frames etc.
- Designing a recursive algorithm:
  - attempt to solve the given problem instance by constructing and solving smaller instances of the same type
  - solve the simplest instances directly
- Top level recursive call: the call made from the main program, or the first call made to the recursive function
- Level 1 recursive calls: calls made directly while executing the top level call
- Base cases: input values for which the top level call returns without recursing
- Preconditions: valid values for inputs
- Problem size: something indicative of the amount of e=work needed to find the solution. Needs to be chosen creatively.
- To check if a recursive function is correct we should check:
  - there are base cases and correct results are obtained for the base cases
  - the level 1 recursive calls satisfy the preconditions
  - the problem size reduces but cannot reduce indefinitely
  - if the level 1 calls work correctly, whether the top level call works correctly
- The execution of any recursive program can be visualized by drawing its "Execution tree" or its "Recursion tree"

- Fibonacci sequence are the Virahanka numbers
- The natural recursive algorithm might be very slow. By examining what the algorithm does, we might be able to discover a better algorithm.
- The function main is allowed not to have a return statement
- If code in a file F calls a function f, it must be declared inside F, textually before any call to it. A function definition is a declaration. A function declaration is essentially just the definition without the body. Other names for declaration are ”Signature” and ”Prototype”.
- If a file contains a call to a function but does not contain its definition, then it can only be partially compiled into an object module. To get an executable program, all the object modules containing all called functions must be linked together.
- Tedious to remember what declarations to include in each file. Instead, we can put all declarations into a header file, and ”include” the header file into every file. Header files customarily have the suffix .h or .hpp, or no suffix.
- It is acceptable if we have both a declaration and then the definition of a function in the same file
- If header file is mentioned in “ ”, it is picked up from the current directory. If it is mentioned in <>, it is picked up from some standard place.
- Namespace = catalog of names. The “full name” of a function defined in a namespace N is N::f.
- Defining a namespace:

  ```
  namespace N{
  declarations/definition of names
  }
  ```

- You can add more names to a namespace N simply by writing namespace n{} again. A name g defined without putting it inside a namespace is said to belong to the global namespace. Its fullname is ::g.
- You will be allowed to use any name from N without having to write N:: before it if you put the following line at the top of your program: using namespace n;
- Some initial lines of code in C++:

  ```
  #include <iostream>
  #include <cmath>
  using namespace std;
  ```

  The names cin, cout, endl are defined in the namespace std, in the standard header file iostream. The header file cmath includes math functions such as sqrt, sin, abs.
- To pass a function h to another function B the type of h is: std::function<return-type(parameter1-type, . . .)>. Must include header file ¡functional¿ in order to use this feature.
- Lambda expression = an expression which represents a (nameless) function. General form: [](parameter list)body.
- You can pass arguments to the function and it is evaluated like an ordinary function call: [](double x)return x*x − 2;(3.5)
- Return type is not stated explicitly. It is inferred by the C++ compiler. In case it is not possible to infer the type, you can specify it explicitly too: [](parameter-list)-¿return type body

- In lambda expressions you can capture outside variables by putting a comma seperated list in [], i.e., lambda expressions can also capture the values of variables defined in the function in which the lambda expression appears.
- One or more parameters occurring at the end of the parameter list can be given default values. Note that if you want to specify a default value for the r^{th} parameter, you must specify a default value for all subsequent parameters as well.
- C++ allows you to define multiple functions with the same name, provided they have different argument lists
- General form of an array: data-type array-name[size];
- Indexing into an array happens very fast, independent of how many elements are present in the array
- Array name = address of allocated region = address of 0^{th} array element. Type of array name: int *. Array name is a pointer, but its value cannot be changed.
- [] is a binary operator. aname[index] means that the variable is stored at aname + S*index, where S = size of a single element of the type aname points to. The computer does a multiplicatin and addition to find the position of the element in memory. Note that only a single multiplication and addition is done, however large the array is.
- Some programming languages prevent index out of range by explicitly checking. Index checking is not done in C++, because it takes extra work and C++ designers believe that it is the programmer's job to ensure that the index is in range.
- In a function call, array elements are not copied. If the function modifies the elements, then the modification will be seen in the calling program.
- The interesting part is the [] operator: given the address of an array and an index it can get us to the corresponding element, even if the address belongs to a different activation frame
- An alternate syntax is also allowed: int avg(double M[], int n).... In this double M[] is synonymous with double *M, but slightly more indicative that we expect M to be an array name, and not just any old pointer.
- The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. The algorithm maintains two subarrays in a given array: The subarray which is already sorted, and the remaining subarray which is unsorted. In every iteration of selection sort, the minimum element (considering ascending order) from the unsorted subarray is picked and moved to the sorted subarray.
- Time complexity of selection sort: O(n^2), where n is the number of elements being sorted
- char type meant to store single letter. Array of char can be used to store sequences of letters.
- Standard protocol inherited from the C language for char:
  - store characters in the string in the array starting from element 0
  - after all the characters are stored, store the character '\0'
  - '\0' = "sentinel" = character whose ASCII value is 0. "null" character.
  - key idea: "Everything until '\0' is a part of the string, not what comes later"
  - no need to explicitly store the length of the string
- Safe way to read into a char array: cin.getline(charArrayName, n);

- For 2-dimensional arrays values picked from the initialization list in row major order
- When passing 2-dimensional arrays to functions, C++ requires to specify the number of columns as a fixed number at the time of writing the function. This is a C++ language limitation.
- We can call main function with a more elaborate signature as shown:

  ```
  namespace N{
  declarations/definition of names
  }
  ```

  argc: gives the number of “words” typed on the command line. Each element of argv is a pointer to each word typed on the command line.
- Linear search is a sequential searching algorithm where we start from one end and check every element of the list until the desired element is found. It is the simplest searching algorithm. Time complexity: O(n).
- Binary Search is a searching algorithm used in a sorted array by repeatedly dividing the search interval in half. The idea of binary search is to use the information that the array is sorted and reduce the time complexity to O(Log(n)).
- If you are likely to search an array frequently, first sort it, then binary search. The time to sort the array will be compensated by the time saved in subsequent searches.
- Merge sort divides the input array into two halves, calls itself for the two halves, and then it merges the two sorted halves. Time complexity: O(n Log(n))
- Structure is a group/supervariable which is a collection of all information about an entity
- Members of a structure are the variables in the collection. Each structure has a type which defines what variables there will be in the collection. You can define a structure type foe each type of entity that you want to represent on the computer - “Programmer defined type”
- General form of struct:

  ```
  struct structure-type{
  member1-type member1-name;
  member2-type member2-name;
  ...
  }; // Don't forget the semicolon!
  ```

- int, char, double etc. are primitive data types
- One structure can contain another
- Main program is at high level
- (*x).y is same as x −> y
- NULL is a keyword in C++. It can be assigned to any pointer, to indicate that the pointer does not point to anything meaningful.
- Functions inside the structure which is used to access the attributes are called member functions
- The member function call can modify the members in the receiver, since the receiver is implicitly passed by reference
- The only difference between a structure and a class is that structure members have public access by default and class members have private access by default
- An Object is an instance of a Class

- In C++, the programmer may define a special member function called a constructor. The constructor is called whenever an instance of the struct is created. A constructor has the same name as the struct, and no return type. The code inside the constructor can perform initializations of members. Constructor can take arguments. You can have many constructors, provided they have different signatures.
- C++ defines a constructor which takes no arguments, and does nothing if and only if you do not define a constructor. A constructor that does not take arguments (defined by you or C++) is called a default constructor.
- If a member is itself a struct, its constructor is called first
- Operator overloading: consider x@y where @ is any “infix” operator, C++ treats this as: x.operator@(y). operator@ must be a member function.
- It is possible to restrict access to members or member functions of a struct. This is called access control. Members declared private can only be accessed inside the definition of the struct.
- Typical strategy: declare all data members to be private, and some subset of function members to be public
- public:, private: are access specifiers. An access specifier applies to all members defined following it, until another specifier is given.
- cin, cout are objects of class istream, ostream respectively. <<, >>: operators defined for the objects of these classes.
- You must include header file <fstream> to use ifstream and ofstream
- Example of file i/o:

```
#include <fstream>
#include <iostream>
int main(){
 ifstream infile("f1.txt");
 ofstream outfile("f2.txt");
 int v;
 infile >> v;
 outfile << v<<endl;
 }
```

- object, structure variable, instance of a class: synonyms
- Object oriented programming: metholodogy for programming design -
  - suggests that real life entities be modelled using structs/classes
  - recommends member functions be used for performing operations
  - recommends that data members be hidden
- If you do not define any member functions, C++ defines folloeing member functions:
  - constructor
  - assignment operator
  - copy constructor
  - destructor

  If you define any of these, your definitions override.
- For nested structures, first constructors of members are called, then constructor of outer object

- Assignment operator: if you do not define an assignment operator, operator=, C++ defines one. This predefined assignment operator simply does member by member copy.
- Copy constructor: when you pass a structure to a function (by value), a copy needs to be made. This is done by a function called the copy constructor. C++ provides a predefined copy constructor. It copies member by member.
- Destructor: when an object goes out of scope, or is deleted, a destructor member function gets called. Predefined destructor does nothing.
- In automatic variables, memory allocation/deallocation happens automatically
- "Heap" memory, also known as "dynamic" memory, is an alternative to local stack memory. Local memory is quite automatic. Local variables are allocated automatically when a function is called, and they are deallocated automatically when the function exits. Heap memory is different in every way.
- It is possible to explicitly request that, memory for a certain variable be allocated in the heap. When there is no more use for the allocated memory, the program must explicitly return the memory to the heap. After the memory is returned, it can be used to satisfy other memory allocation requests in the future. Neither allocation, nor deallocation of this memory is automatic. No deallocation happens just because a block is exited by the control.
- The standard library classes use heap memory
- Syntax for allocating heap memory: new T (T = typename). Memory for storing one variable of type T is allocated on the heap. new T returns address of allocated memory. After the memory is no longer needed, it must be returned by executing delete. An example:

  ```
  class Book{
   char title[100];
   double price;
  };
  Book *bptr;
  bptr = new Book;
  bptr-\hspace{-0.2cm}>price = 399;
  ...
  delete bptr;
  ```

- Allocating arrays on the heap:

  ```
  char* cptr;
  cptr = new char[10];
  // allocates array of length 10.
  // array can be accessed as usual
  // cptr[0],...,cptr[9]
  delete[] cptr;
  // When not needed.
  // Note: delete[] not delete
  ```

- Storing many names:

  ```
  char *names[100]; // array of pointers to char
  for(int i=0; i<100; i++){
   char buffer[80]; cin >> buffer;
   int L = length(buffer)+1; // length: returns no of chars till '\0'
   // +1 so that we can append '\0'.
   names[i] = new char[L];
   for(int j=0;j<L;j++)
   names[i][j] = buffer[j];
  }
  ```

- Managing heap memory is tricky and prone to errors:

  - forgetting to deallocate (delete) memory
  - referring to memory that has been deallocated ("dangling reference")
  - destroying the only pointer to memory allocated on the heap before it is deallocated ("memory leak")

- Simple strategy for preventing memory leaks:
  - suppose a certain pointer variable, ptr, is the only variable that contains the address of a variable allocated on the heap
  - we must not store anything into ptr and destroy its contents
  - when ptr is about to go out of scope, (control exits the block in which ptr is defined) we must execute delete ptr;

- For preventing dangling references, ensure that at all times, each variable on the heap will be pointed to only by one pointer

- this is a C++ keyword. Inside a member function, this is a pointer to the receiver. *this would mean the receiver itself.

- #include <string> is needed to use the string class

- #include <vector> is needed to use the vector class

- Use vectors, not arrays!

- General form of maps: map<indextype, valuetype> mapname;