Indian Institute of Science
Bangalore, India
भारतीय विज्ञान संस्थान
बंगलौर, भारत

DS286 | 2016-08-12

L2: Introduction to Programming

Yogesh Simmhan

simmhan@cds.iisc.ac.in

CDS
The Department of Computational and Data Science

ALGORITHM + DATA STRUCTURES = PROGRAM*

- Programming: From craft to a discipline
  - Djikstra: Programming a scientific subject, intellectual challenge
  - Hoare: Exacting program analysis based on mathematical reasoning
- Programs are concrete formulations of abstract algorithms
  - Large, complex programs involve complicated data sets
  - Hence, need data structures
- Algorithms & Data Structures are inter-twined
  - Cannot choose/design one without the other

**Algorithms and Data Structures, Niklaus Wirth, 1985…Turing Award winner*

Digital Data

Movies

Music

Photos

Protein Shapes

Maps

DNA

```
gatcttttta tttaaacgat ctctttatta gatctcttat taggatcatg atcctctgtg
gataagtgat tattcacatg gcagatcata taattaagga ggatcgtttg ttgtgagtga
ccggtgatcg tattgcgtat aagctgggat ctaaatggca tgttatgcac agtcactcgg
cagaatcaag gttgttatgt ggatatctac tggttttacc ctgcttttaa gcatagttat
acacattcgt tcgcgcgatc tttgagctaa ttagagtaaa ttaatccaat ctttgaccca
```

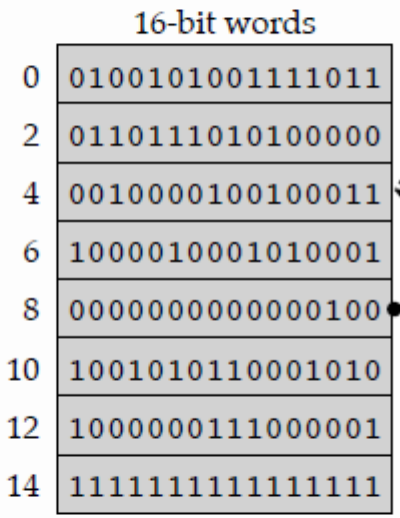
0010101001010101010100100100101010000010010010100....

RAM = Symbols + Pointers

(for our purposes)

Binary	Oct	Dec	Hex	Glyph
010 0000	040	32	20	SP
010 0001	041	33	21	!
010 0010	042	34	22	"
010 0011	043	35	23	#
010 0100	044	36	24	$
010 0101	045	37	25	%
010 0110	046	38	26	&
010 0111	047	39	27	'
010 1000	050	40	28	(
010 1001	051	41	29	)

ASCII table: agreement for the meaning of bits



We may agree to interpret bits as an address (pointer)

Physically, RAM is a random accessible array of bits.

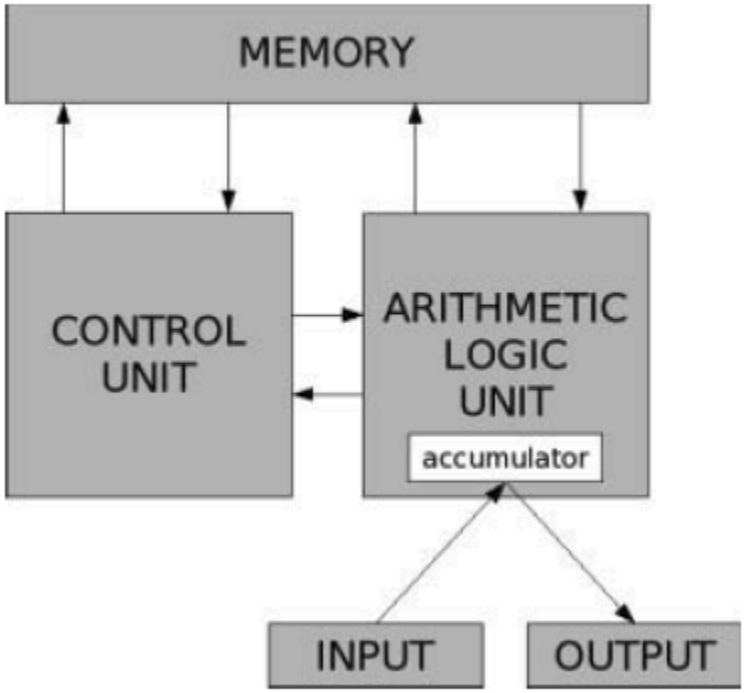
=> We can store and manipulate arbitrary symbols (like letters) and associations between them.

Digital Data must be…

- Encoded
  - Analog ←→ Digital
- Arranged
  - On disk, in memory
- Accessed
  - Insert, delete, update, compared
- Processed
  - Algorithms: Searching, Sorting, Distance measures

Stored Program: *Program & Data are both digital…*

Treat data and instructions as the same thing.



Syntax, Static Semantics, and Semantics

- Syntax: which sequences of characters and symbols constitute a well-formed string
  - Grammar, format, API
- Semantics: what that meaning of the syntax is
- Static semantics: which *well-formed strings* have a meaning
  - Can be checked at "compile" time

Compiled vs. Interpreted Programs

- Interpreted
  - **source code** → checker → interpreter → **output**
- Compiled
  - **source code** → checker/compiler → **object code** → interpreter → **output**

Declarative Knowledge

- Declarative knowledge is composed of statements of facts
  - "The sum of the arithmetic progression of a natural number 'n' is the sum of all natural numbers from '1' to the number 'n', inclusive."
  - "y is the square root of x if and only if y*y = x"
  - "Return the largest number in a given list of numbers"
- It states the "what" not the "how"
- E.g. in programming, **SQL** queries are declarative
  - `SELECT MAX(column_name) FROM table_name;`

http://www.w3schools.com/sql/sql_func_max.asp

Imperative Knowledge

- Imperative knowledge is about how to accomplish something. It gives the recipe…
- The sum of the arithmetic progression of a natural number ‘n’ is given as follows:
  - 1+2+3+...+n
  - n*(n+1)/2
- …

Iterative Algorithms

- Summation of Series: Arithmetic Progression

```
1+2+3+...+n=?
```


https://en.wikipedia.org/wiki/Arithmetic_progression

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


int main()
{
        int num=0;
        int sum=0;
        int i;

        cout<<"Sum the series to what number? : ";
        cin>>num;
        sum = 0;
        for(i=1; i<=num;i++){
                sum = sum + i;
                cout<<i<<": Current sum is "<<sum<<endl;
        }

        cout<<"The sum of the series till "<<num<<" is "<< sum <<endl;
        cout<<"The sum of the series till "<<num<<" is also "<< (num * (num+1))/2 <<endl;


        return(0);
}
```

```
simmhan@serc-lenovo:~/ds286.aug16$ ./a.out
Sum the series to what number? : 5
1: Current sum is 1
2: Current sum is 3
3: Current sum is 6
4: Current sum is 10
5: Current sum is 15
The sum of the series till 5 is 15
The sum of the series till 5 is also 15
simmhan@serc-lenovo:~/ds286.aug16$
```

Newtons' Series for Square Root

Newton's iteration is an algorithm for computing the square root $\sqrt{n}$ of a number n via the recurrence equation

$$x_{k+1} = \frac{1}{2}\left(x_k + \frac{n}{x_k}\right),$$

where $x_0 = 1$. This recurrence converges quadratically as $\lim_{k\to\infty} x_k$.

http://mathworld.wolfram.com/NewtonsIteration.html

```
#include <iostream>
#include <math.h>       /* sqrt */
using namespace std;


double approxSQRT(double,double);

int main()
{
        double num=0;      // num represents the number under the square root.
        double appox=0;     // appox represents the current appox value of the square root.
        int count=0;

        cout<<"Take the square root of what number? : ";
        cin>>num;
        appox = num;
        do{
                appox = (0.5)*(appox + (num/appox));
                count++;
                cout<<count<<": Updated square root to "<< appox <<endl;
        } while((appox*appox-num)>0.0001);
        cout<<"The approx square root of "<<num<<" is "<< appox <<endl;
        cout<<"The exact  square root of "<<num<<" is "<< sqrt(num) <<endl;


        return(0);
}
```


http://forums.codeguru.com/showthread.php?473394-Newtons-Method-for-Approximating-Roots

```
simmhan@serc-lenovo:~/ds286.aug16$ ./a.out
Take the square root of what number? : 100
1: Updated square root to 50.5
2: Updated square root to 26.2401
3: Updated square root to 15.0255
4: Updated square root to 10.8404
5: Updated square root to 10.0326
6: Updated square root to 10.0001
7: Updated square root to 10
The approx square root of 100 is 10
The exact  square root of 100 is 10
simmhan@serc-lenovo:~/ds286.aug16$
```

For vs. While vs. Do While

- FOR: Fixed number of iterations
- WHILE: Test if condition met
  - Test before/after iteration

Recursive Algorithms

- Fibonacci Series

$$0,\ 1,\ 1,\ 2,\ 3,\ 5,\ 8,\ 13,\ 21,\ 34,\ 55,\ 89,\ 144,\ \ldots$$

$$F_n = F_{n-1} + F_{n-2},$$

with seed values[1][2]

$$F_1 = 1,\ F_2 = 1$$

or[5]

$$F_0 = 0,\ F_1 = 1.$$

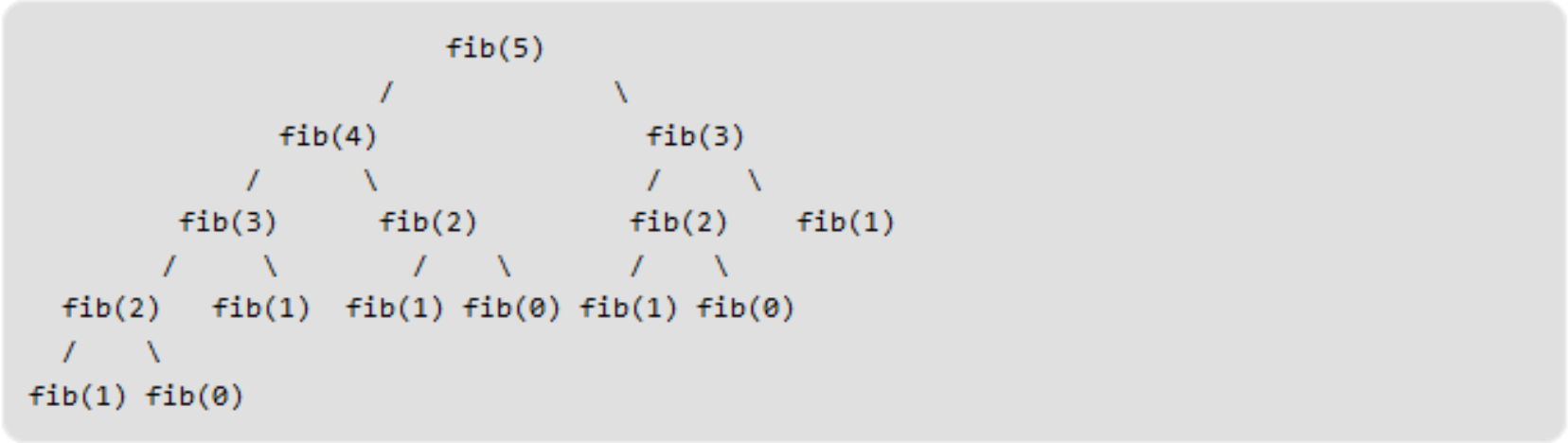
https://en.wikipedia.org/wiki/Fibonacci_number

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

int fib(int n)
{
        int f1, f2;
        if (n <= 1)
                return n;

        f1 = fib(n-1);
        f2 = fib(n-2);
        cout<<"("<<f1<<" + "<<f2<<")="<<(f1+f2)<<endl;
        return f1 + f2;
}

int main()
{
        int num=0;      // the index of the fibonacci number in the series
        int f=-1;

        cout<<"Which index in the fibonacci series do you want? ";
        cin>>num;
        f = fib(num);
        cout<<"The "<<num<<"th fibonacci number is "<< f <<endl;

        return(0);

}
```


http://www.geeksforgeeks.org/program-for-nth-fibonacci-number/

```
simmhan@serc-lenovo:~/ds286.aug16$ ./a.out
Which index in the fibonacci series do you want? 5
(1 + 0)=1
(1 + 1)=2
(1 + 0)=1
(2 + 1)=3
(1 + 0)=1
(1 + 1)=2
(3 + 2)=5
The 5th fibonacci number is 5
simmhan@serc-lenovo:~/ds286.aug16$
```

Recursive Fibonacci



- **Time Complexity:** T(n) = T(n-1) + T(n-2) … *exponential!*
- **Space Complexity**: O(n) if for the function call stack

http://www.geeksforgeeks.org/program-for-nth-fibonacci-number/

```
#include <iostream>
#include <math.h>       /* sqrt */
using namespace std;

int depth = 0;
double appoxSqrt(double num, double appox)
{
        appox = (0.5)*(appox + (num/appox));
        depth++;
        cout<<depth<<": Updated square root to "<< appox <<endl;
        if((appox*appox-num)>0.0001)
                return appoxSqrt(num, appox);
        else
                return appox;
}

int main()
{
        double num=0;      // num represents the number under the square root.
        double appox=0;     // appox represents the current appox value of the square root.
        int count=0;

        cout<<"Take the square root of what number? : ";
        cin>>num;
        appox = num;
        appox = appoxSqrt(num, appox);
        cout<<"The approx square root of "<<num<<" is "<< appox <<endl;
        cout<<"The exact  square root of "<<num<<" is "<< sqrt(num) <<endl;


        return(0);
}
```

Tasks

- Think about a few iterative algorithms for mathematical operations, and implement them…e.g.
  - Pi
  - Geometric progression
- Think about a few recursive algorithms

Announcements & Tasks

- No Lab on Aug 15: ***Happy Independence Day!***
  - 5k run@630am, Flag hoisting@830am … Main Building
  - Blood donation 930am-3pm@CCE
- Next lecture on Aug 17
- Course website is up: `cds.iisc.ac.in/courses/ds286`
- Sign up for mailing list, if not done so already
  - `mailman.serc.iisc.in/mailman/listinfo/ds286.aug16`
- Fill in Google Sheets with details
  - Link will be posted on mailing list
- '**`turing`**' Cluster access will be provided by Mon Aug 15
  - All submissions MUST work (compile, run) on cluster node!
- Finish Assignment 0 by Fri Aug 19 *(0 points)*
- Assignment 1 will be posted by Aug 19, due Aug 28 *(50 points)*

Questions?