CS1100 – Introduction to Programming

CS1100 – Introduction to Programming

Instructor:

Shweta Agrawal (shweta.a@cse.iitm.ac.in)

CS1100 – Introduction to Programming

Instructor:

Shweta Agrawal (shweta.a@cse.iitm.ac.in)

Lectures : Three per week (F1 slot)

- Wednesday: 11:00 - 11:50 pm
- Thursday: 9:00 - 9:50 pm
- Friday: 8:00 - 8:50 pm

CS1100 – Introduction to Programming

Instructor:

Shweta Agrawal (shweta.a@cse.iitm.ac.in)

Lectures : Three per week (F1 slot)

- Wednesday: 11:00 - 11:50 pm
- Thursday: 9:00 - 9:50 pm
- Friday: 8:00 - 8:50 pm

Lab : One session per week

- Wednesday (R1) /Thursday (S1): 2-5 pm

Course Outline

- Introduction to Computing and Computers.
- Programming (in C).
- Exercises and examples from various domains.
- Problem solving using computers.

Course Requirements

- Labs: 12 assignments, total weight 30%.
- Quiz 1 (Feb 23): 20 %.
- Quiz 2 (March 22): 20%
- Final (May 13): 30%

Course Requirements

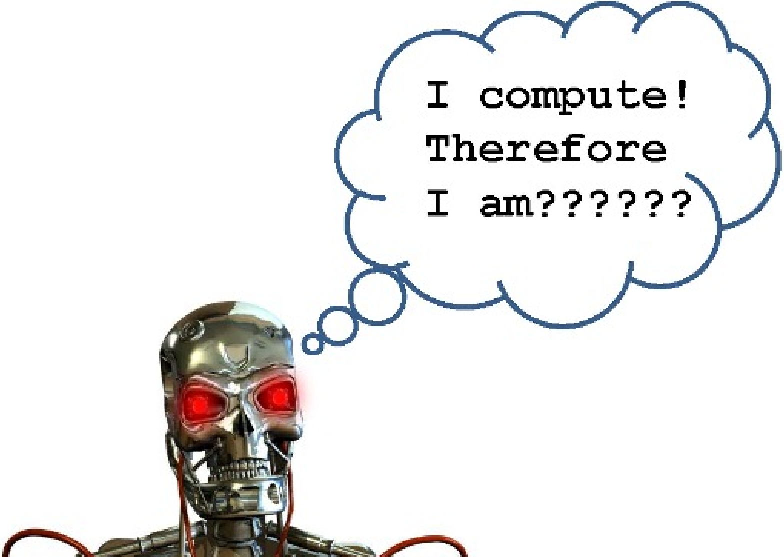
- Labs: 12 assignments, total weight 30%.
- Quiz 1 (Feb 23): 20 %.
- Quiz 2 (March 22): 20%
- Final (May 13): 30%
- Attendance: As per institute policy, no exceptions.

Course Requirements

- Labs: 12 assignments, total weight 30%.
- Quiz 1 (Feb 23): 20 %.
- Quiz 2 (March 22): 20%
- Final (May 13): 30%
- Attendance: As per institute policy, no exceptions.
- Ethical violations reported to disciplinary committee.

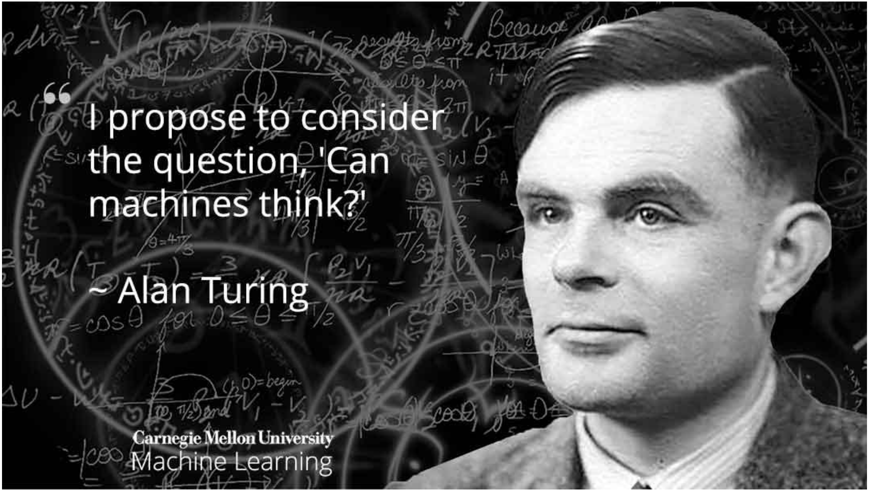
What is a computer?

What is a computer?



What is the difference between a human and a computer?

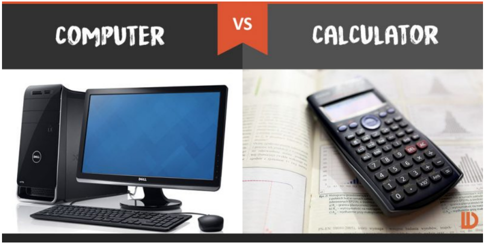
What is the difference between a human and a computer?



“I propose to consider the question, 'Can machines think?'
~ Alan Turing
Carnegie Mellon University
Machine Learning

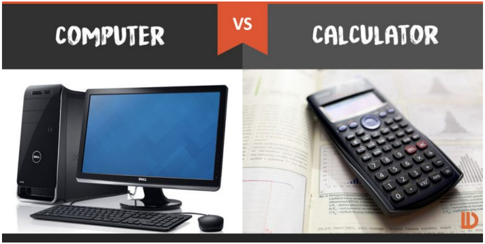
Simpler: Difference between Calculator and Computer?

Simpler: Difference between Calculator and Computer?



- Calculators are single-purpose devices that perform mathematical operations input by the user.

Simpler: Difference between Calculator and Computer?



- Calculators are single-purpose devices that perform mathematical operations input by the user.
- Computers are calculators that have vastly expanded capabilities, and are often called "general purpose computing devices".

What is a computer?

We started with machines that can do one job.

What is a computer?

We started with machines that can do one job.

What is a computer?

We started with machines that can do one job.

What is a computer?

- A huge electrical circuit.
- Can accept data from external world, remember, process it, return results to the external world.
- Data : Text typed in your mobile, electrical signals from a sensor which senses the temperature in farms, speech, handwriting, touch.
- Program : A precise description of steps that we want to perform on the data.

Goal for today – have fun!

Observe the following patterns:

```
                                ********
********        **              *      *
********        ****            *      *
********        ******          *      *
********        ********        ********
```

Goal for today – have fun!

Observe the following patterns:

```
                                    ********
********          **                *      *
********          ****              *      *
********          ******            *      *
********          ********          ********
```

- It is very easy to draw these patterns on paper.

Goal for today – have fun!

Observe the following patterns:

```
                                    ********
********          **                *      *
********          ****              *      *
********          ******            *      *
********          ********          ********
```

- It is very easy to draw these patterns on paper.
- How would you describe the same to a friend on the phone?

Describing a pattern

- How do you communicate?
- Use commonly understood commands.

Describing a pattern

- How do you communicate?
- Use commonly understood commands.
    - draw a star.
    - go to new line.
    - repeat a set of commands k times.

Describing a pattern

- How do you communicate?
- Use commonly understood commands.
    - draw a star.
    - go to new line.
    - repeat a set of commands *k* times.

```
********
********
```

- repeat 8 times
    - draw a star.
- go to new line.
- repeat 8 times
    - draw a star.

Can you describe all patterns in that list?

```
                                    ********
********          **                *      *
********          ****              *      *
********          ******            *      *
********          ********          ********
```

- draw a star.
- go to new line.
- repeat a set of commands k times.

Can you describe all patterns in that list?

```
                                        ********
********          **                    *      *
********          ****                  *      *
********          ******                *      *
********          ********              ********
```

- draw a star.
- go to new line.
- repeat a set of commands *k* times.
- move right (without drawing a star).

Your computer is your friend ...

What have we achieved?

- Describe simple patterns using a set of commands.
- When required, introduce new commands.
  (and also inform the friend of its meaning).

Your computer is your friend ...

What have we achieved?

- Describe simple patterns using a set of commands.
- When required, introduce new commands.
  (and also inform the friend of its meaning).

- Recall: A **Program** is a precise description of steps that we want to perform on the data.

Your computer is your friend ...

What have we achieved?

- Describe simple patterns using a set of commands.
- When required, introduce new commands.
  (and also inform the friend of its meaning).
- Recall: A **Program** is a precise description of steps that we want to perform on the data.
- So, is the above a "program"?

Your computer is your friend ...

What have we achieved?

- Describe simple patterns using a set of commands.
- When required, introduce new commands.
  (and also inform the friend of its meaning).

- Recall: A **Program** is a precise description of steps that we want to perform on the data.

- So, is the above a "program"? Yes. But the computer does not know the above language.

Your computer is your friend ...

What have we achieved?

- Describe simple patterns using a set of commands.
- When required, introduce new commands.
  (and also inform the friend of its meaning).

- Recall: A **Program** is a precise description of steps that we want to perform on the data.

- So, is the above a “program”? Yes. But the computer does not know the above language.
- Is the above a “computer program”?

Your computer is your friend ...

What have we achieved?

- Describe simple patterns using a set of commands.
- When required, introduce new commands.
  (and also inform the friend of its meaning).

- Recall: A **Program** is a precise description of steps that we want to perform on the data.

- So, is the above a "program"? Yes. But the computer does not know the above language.
- Is the above a "computer program"? No

Your computer is your friend ...

What have we achieved?

- Describe simple patterns using a set of commands.
- When required, introduce new commands.
  (and also inform the friend of its meaning).

- Recall: A **Program** is a precise description of steps that we want to perform on the data.

- So, is the above a “program”? Yes. But the computer does not know the above language.
- Is the above a “computer program”? No

- Goal of the course: learn to program the computer to perform different tasks.

Illustrative Example : Turtle Drawing

Imagine that we have taught the computer to display a turtle and move it according to the following commands.

- forward(n) : "Move the turtle n pixels in the direction it is currently headed."
- left(d) : "Make the turtle, turn d degrees to the left."
- wait(t) : "Do nothing for t seconds."

Illustrative Example : Turtle Drawing

Imagine that we have taught the computer to display a turtle and move it according to the following commands.

- forward(n) : “Move the turtle n pixels in the direction it is currently headed.”
- left(d) : “Make the turtle, turn d degrees to the left.”
- wait(t) : “Do nothing for t seconds.”
- Ignore first four lines; they just make sure computer knows what to do in the above commands.

```
#include simplecpp
main_program
{
  turtleSim();
  forward(100); left(90);
  forward(100); left(90);
  forward(100); left(90);
  forward(100);
  wait(5);
}
```

Turtle Computer - More exercises

- How will you make the turtle draw a triangle?
- how about a hexagon?
- how about a decagon?
- how about a picture which “looks like” a circle?

Summarizing . . .

The pattern drawing, turtle drawings ... what have we achieved?

Summarizing . . .

The pattern drawing, turtle drawings ... what have we achieved?

- We made our “trained friend” to draw patterns using simple instructions. This was more English instructions.

Summarizing . . .

The pattern drawing, turtle drawings ... what have we achieved?

- We made our “trained friend” to draw patterns using simple instructions. This was more English instructions.
- We made our “turtle-trained computer” to draw patterns using simple instructions. This was more “short instructions”.

Summarizing . . .

The pattern drawing, turtle drawings ... what have we achieved?

- We made our "trained friend" to draw patterns using simple instructions. This was more English instructions.
- We made our "turtle-trained computer" to draw patterns using simple instructions. This was more "short instructions".
- Bottomline : the computer should know the meaning of the commands that we give.

Summarizing . . .

The pattern drawing, turtle drawings ... what have we achieved?

- We made our "trained friend" to draw patterns using simple instructions. This was more English instructions.
- We made our "turtle-trained computer" to draw patterns using simple instructions. This was more "short instructions".
- Bottomline : the computer should know the meaning of the commands that we give.
- Computers are "trained" in some languages.

What Languages are Computers Taught with . . .

Programming languages : C, C++, Java, Python . . .

- the languages that the computers are apriori trained on. (how? - for later !).
- means of communication with a computer.

What Languages are Computers Taught with . . .

Programming languages : C, C++, Java, Python . . .

- the languages that the computers are apriori trained on. (how? - for later !).
- means of communication with a computer.
- In this course : The C Programming Language
  - Developed by Dennis Ritchie (1969 – 1973).
  - One of the most popular programming languages.
  - Used to “write” large softwares, scientific computing etc.

What Languages are Computers Taught with . . .

Programming languages : C, C++, Java, Python . . .

- the languages that the computers are apriori trained on. (how? - for later !).
- means of communication with a computer.
- In this course : The C Programming Language
  - Developed by Dennis Ritchie (1969 – 1973).
  - One of the most popular programming languages.
  - Used to “write” large softwares, scientific computing etc.
- Was the “turtle program”, a program in written in C?

What Languages are Computers Taught with . . .

Programming languages : C, C++, Java, Python . . .

- the languages that the computers are apriori trained on. (how? - for later !).
- means of communication with a computer.
- In this course : The C Programming Language
  - Developed by Dennis Ritchie (1969 – 1973).
  - One of the most popular programming languages.
  - Used to "write" large softwares, scientific computing etc.
- Was the "turtle program", a program in written in C? No

What Languages are Computers Taught with . . .

Programming languages : C, C++, Java, Python . . .

- the languages that the computers are apriori trained on. (how? - for later !).
- means of communication with a computer.
- In this course : The C Programming Language
  - Developed by Dennis Ritchie (1969 – 1973).
  - One of the most popular programming languages.
  - Used to "write" large softwares, scientific computing etc.
- Was the "turtle program", a program in written in C? No
- To be able to write programs in C, we need to learn the language.

What Languages are Computers Taught with . . .

Programming languages : C, C++, Java, Python . . .

- the languages that the computers are apriori trained on. (how? - for later !).
- means of communication with a computer.
- In this course : The C Programming Language
  - Developed by Dennis Ritchie (1969 – 1973).
  - One of the most popular programming languages.
  - Used to “write” large softwares, scientific computing etc.
- Was the “turtle program”, a program in written in C? No
- To be able to write programs in C, we need to learn the language.
- That is the goal of this course.

Summarizing . . .

- Programming is fun.

Summarizing . . .

- Programming is fun.
- Programming is useful - computational techniques to simulate, visualize and conclude without actually making the physical system.

Summarizing . . .

- Programming is fun.
- Programming is useful - computational techniques to simulate, visualize and conclude without actually making the physical system.
- Programming is the Designer and the Programmer of a company. You need to know how to manage both!
- The Designer designs – MUST be accurate. The product must be relevant – so we need a CTO too.
- The programmer converts the design verbatim to a program in a language that the computer understands! S/he is responsible for efficient programming too.

Summarizing . . .

- Programming is fun.
- Programming is useful - computational techniques to simulate, visualize and conclude without actually making the physical system.
- Programming is the Designer and the Programmer of a company. You need to know how to manage both!
- The Designer designs – MUST be accurate. The product must be relevant – so we need a CTO too.
- The programmer converts the design verbatim to a program in a language that the computer understands! S/he is responsible for efficient programming too.
- "Why this course", "What is in the course".

Books for the course

- Paul Deitel and Harvey Deitel. C: How to Program.
- V. Rajaraman: Computer Programming in C.
- R. G. Dromey: How to Solve It By Computer?
- Kernighan and Ritchie: The C Programming Language.

Acknowledgements

- Slides for the course are based on material prepared by faculty of CSE department IITM.
- Ideas will also be drawn from a book by Prof. Abhiram Ranade (IITB) (Introduction to programming using C++).
- All images – courtsey Google Images.
- This applies for all slides throughout the course.

Rest of this week..

- More on turtle graphics. (today !)
- A brief history about computers. (some of them today !)

Rest of this week..

- More on turtle graphics. (today !)
- A brief history about computers. (some of them today !)
- What is a computer made of?

Rest of this week..

- More on turtle graphics. (today !)
- A brief history about computers. (some of them today !)
- What is a computer made of?
    - Do we need to know internals of a computer to be able to program it?

Rest of this week..

- More on turtle graphics. (today !)
- A brief history about computers. (some of them today !)
- What is a computer made of?
    - Do we need to know internals of a computer to be able to program it?
- How does a computer perform so many diverse tasks (number crunching, weather prediction, playing chess, ...)?

Rest of this week..

- More on turtle graphics. (today !)
- A brief history about computers. (some of them today !)
- What is a computer made of?
    - Do we need to know internals of a computer to be able to program it?
- How does a computer perform so many diverse tasks (number crunching, weather prediction, playing chess, ...)?
    - Convert every task into a task on numbers.
    - How to represent numbers on computers?

More on the Turtle Language

Question : What do we get by this program?

```
#include simplecpp
main_program
{
  turtleSim();
  forward(100); left(72);
  forward(100); left(72);
  forward(100); left(72);
  forward(100); left(72);
  forward(100);
  wait(5);
}
```

What about a Decagon?

Turtle knows more ...

- `forward(n)`
- `right(d)`
- `left(d)`
- `wait(t)`
- `repeat(k)` { commands }
  repeats the commands *k* times.

What about a Decagon?

Turtle knows more ...

- `forward(n)`
- `right(d)`
- `left(d)`
- `wait(t)`
- `repeat(k)` { commands } repeats the commands *k* times.

```
#include <simplecpp>
main_program
{
  turtleSim();
  repeat(10)
  {
    forward(100);
    left(36);
    wait(1);
  }
  wait(5);
}
```

More fun with Turtle ...

What will the following program draw?

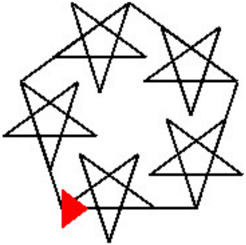
```
#include <simplecpp>
main_program
{
  turtleSim();
  left(72);
  repeat(5)
  {
    forward(200);
    wait(1);
    left(144);
  }
  wait(20);
}
```

More fun with Turtle ...

What will the following program draw?

```
#include <simplecpp>
main_program
{
  turtleSim();
  left(72);
  repeat(5)
  {
    forward(200);
    wait(1);
    left(144);
  }
  wait(20);
}
```

Make the turtle draw this !



Turtle knows more ...

- Turtle can print messages. `cout << ‘‘Hello World";`
- Turtle can wait for an input to be typed by you and use it for the drawing (computation). Command is : `cin >> n;` where `n` is a “variable”.

Turtle knows more ...

- Turtle can print messages. `cout << ‘‘Hello World";`
- Turtle can wait for an input to be typed by you and use it for the drawing (computation). Command is : `cin >> n;` where `n` is a “variable”.
- penUp(): Causes the pen to be raised.
- penDown(): Causes the pen to be lowered.
- sqrt(x) : square root of x.
- sine(x), cosine(x), tangent(x) : trigonometric functions, x is in degrees.

Text-only Turtle

Predict the output:

```
#include <simplecpp>
main_program
{
  cout << "a";
  repeat(5)
  {
    cout << "b";
    repeat(2){  cout << "c"; }
    cout << "d";
  }
}
```

Text-only Turtle

Predict the output:

```
#include <simplecpp>
main_program
{
  cout << "a";
  repeat(5)
  {
    cout << "b";
    repeat(2){  cout << "c"; }
    cout << "d";
  }
}
```

The program will print

```
abccdbccdbccdbccdbccd
```

Drawing a polygon with “given” number of sides

```
#include <simplecpp>
main_program
{
  turtleSim();
  cout << "How many sides?";
  int nsides;
  cin >> nsides;
  repeat(nsides){
    forward(100);
    right(360.0/nsides);
    wait(1);
  }
  wait(10);
}
```

Drawing a polygon with “given” number of sides

```
#include <simplecpp>
main_program
{
  turtleSim();
  cout << "How many sides?";
  int nsides;
  cin >> nsides;
  repeat(nsides){
    forward(100);
    right(360.0/nsides);
    wait(1);
  }
  wait(10);
}
```

`cout << msg;` : Print message msg on the screen.

Drawing a polygon with “given” number of sides

```
#include <simplecpp>
main_program
{
  turtleSim();
  cout << "How many sides?";
  int nsides;
  cin >> nsides;
  repeat(nsides){
    forward(100);
    right(360.0/nsides);
    wait(1);
  }
  wait(10);
}
```

`cout << msg;` : Print message msg on the screen.

`int nsides;` : “Reserve a space in the “blackboard” in which I will store some integer value, and call that cell nsides”.

Drawing a polygon with “given” number of sides

```
#include <simplecpp>
main_program
{
  turtleSim();
  cout << "How many sides?";
  int nsides;
  cin >> nsides;
  repeat(nsides){
    forward(100);
    right(360.0/nsides);
    wait(1);
  }
  wait(10);
}
```

`cout << msg;` : Print message msg on the screen.

`int nsides;` : “Reserve a space in the “blackboard” in which I will store some integer value, and call that cell nsides”.

`cin >> nsides;` : Read an integer value from the keyboard and put it in the cell nsides.

Drawing a polygon with "given" number of sides

```
#include <simplecpp>
main_program
{
  turtleSim();
  cout << "How many sides?";
  int nsides;
  cin >> nsides;
  repeat(nsides){
    forward(100);
    right(360.0/nsides);
    wait(1);
  }
  wait(10);
}
```

`cout << msg;` : Print message msg on the screen.

`int nsides;` : "Reserve a space in the "blackboard" in which I will store some integer value, and call that cell nsides".

`cin >> nsides;` : Read an integer value from the keyboard and put it in the cell nsides.

`360.0/nsides` : represents the value obtained after dividing 360 by whatever is in `nsides`.

Drawing a polygon with “given” number of sides

```
#include <simplecpp>
main_program
{
  turtleSim();
  cout << "How many sides?";
  int nsides;
  cin >> nsides;
  repeat(nsides){
    forward(100);
    right(360.0/nsides);
    wait(1);
  }
  wait(10);
}
```

`cout << msg;` : Print message msg on the screen.

`int nsides;` : “Reserve a space in the “blackboard” in which I will store some integer value, and call that cell nsides”.

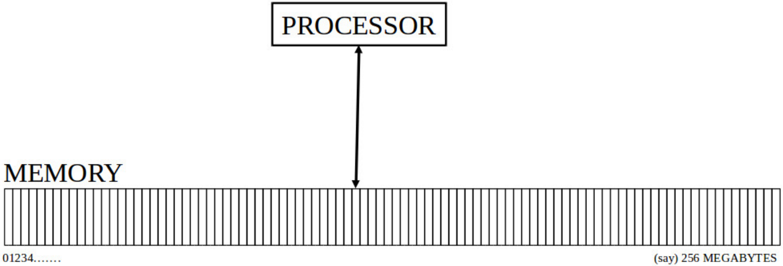
`cin >> nsides;` : Read an integer value from the keyboard and put it in the cell nsides.

`360.0/nsides` : represents the value obtained after dividing 360 by whatever is in `nsides`.

A few general ideas . . .

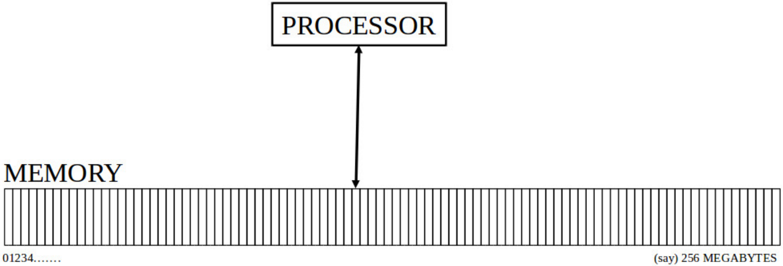
- *Control is at statement w*: Computer is currently executing statement *w*.
- *Control flow*: The order in which statements get executed. Execution starts at top and goes down. Retraced if there is a repeat statement.
- *Variable* used for storing data.
- *Computer memory*: blackboard
- Variable : Region on the board in which you can write a value.
- Variables have names, e.g. nsides. We can use the name to refer to the value written in the variable. Details later.

The Computing Machine



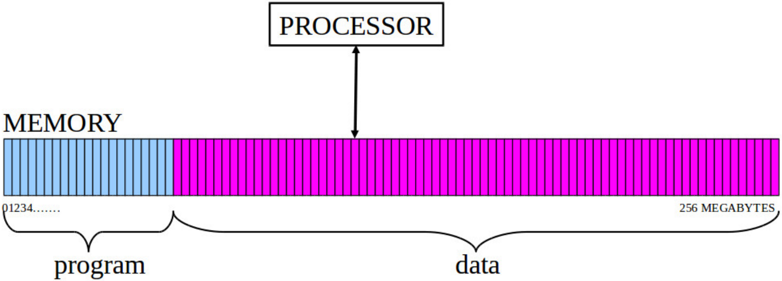
- The computer is made up of a processor and a memory.

The Computing Machine



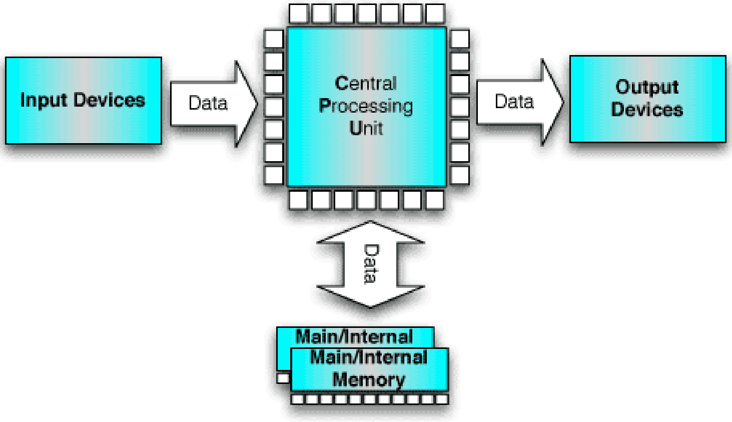
- The computer is made up of a processor and a memory.
- The memory can be thought of as a series of locations to store information.

The Computing Machine



- A program is a sequence of instructions assembled for some given task.
- Most instructions operate on data.
- Some instructions control the flow of the operations.

The Computing Machine : von Neuman Architecture



Coming up...

- How does the computer execute a program?
- How does the computer represent data/programs?
- Introduction to C programming language.