LECTURE 2: MORE ABOUT C

Last Time: Program to sum integers from 1 to n

```
#include<stdio.h>
void main()
{ int i, n, sum;

   sum = 0;
   printf("Input the integer\n");
   scanf("%d", &n);
   for(i=1; i <= n; i=i+1)
        {
          sum = sum + i;
        }
   printf("The sum is %d", sum);
}
```

header

variable declarations

"for" loop

The `if` and `if-else` statements

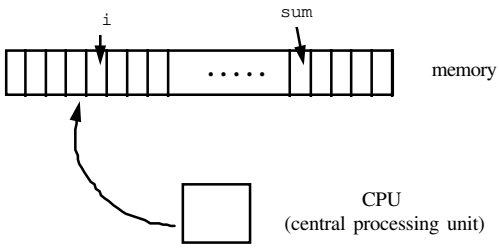
used by programs to "branch" depending upon whether or not a certain condition is met.

```
void main()
{
  int grade;

  scanf("%d", &grade);
  if (grade >75)
       printf("This student passes");
  else
       {
       printf("This student may not pass\n");
       printf("Check student's partying record");
       }
}
```

How variables are stored inside the computer

(a rough picture)




```
sum = sum + i;
```

(CPU reads current values of `i` and `sum`
and stores their sum in the location for `sum`)

variables of type `int` can only store integers between
- 2 billion and + 2 billion

Goal: Write a program to compute e^x

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \ldots$$

($n! = n(n-1)(n-2)\ldots 3.2.1.$)

Notes:

(i) $e = 2.718\ldots\ldots\ldots$

(ii) In a finite amount of time/space, we can only hope to compute e^x upto some finite precision.

```
#include <stdio.h>
void main()
{
  int       i, n;
  float      x, current_term, epowx;

  printf("Input x \n");
  scanf("%f", &x);
  printf("Input the desired precision\n");
  scanf("%d", &n);

  epowx=1.0; current_term=1.0;

  for(i=1; i <= n; i=i+1)
    {
       current_term= (current_term * x)/i;
       epowx = epowx + current_term;
     }

   printf("exp(x) is %f \n", epowx);
}
```


function: a "subprogram" that, given an input, produces an output.

Can be reused ("called") any number of times

```
#include <stdio.h>
float epowerx(float, int);
main()
{
  float a, b;
  int n;

  b=5.5; n=100;

  a = epowerx(b, n);
  printf("%f", a);
  b = epowerx(a, 5);
}

float epowerx(float x, int n)
{
  float epowx, current_term;
  ....
  just the for loop from the previous slide
  ....
  return epowx;
}
```

function declaration

function call

function definition

returned value