CS35L – Spring 2019

Slide set:	4.1
Slide topics:	C programming
Assignment:	4

Assignment 10 Registration

- Presentations begin on Monday, 29 April 2019
- **Choosing a topic**

C Programming

Basic Data Types

- **int**
  - Holds integer numbers
  - Usually 4 bytes
- **float**
  - Holds floating point numbers
  - Usually 4 bytes
- **double**
  - Holds higher-precision floating point numbers
  - Usually 8 bytes (double the size of a float)
- **char**
  - Holds a byte of data, characters
- **void**

A simple C Program

```
#include <stdio.h>

int main() {
	printf("Hello World\n");
	return 0;
}
```

Format Specifiers in C

Defines the type of data to be printed on standard output - printf("%d", 4); //%d is integer

Data Types Revised

Data Type	Range	Bytes	Format Specifiers
char	-128 to 127	1	%c
unsigned char	0 to 255	1	%c
short signed int	-32,768 to 32,767	2	%d
short unsigned int	0 to 65,535	2	%u
signed int	-32768 to 32767	4	%d
unsigned int	0 to 65535	4	%u
long signed int	-2147483648 to 2147483647	4	%ld
long unsigned int	0 to 4294967295	4	%lu
float	-3.4e38 to 3.4e38	4	%f
double	-1.7e308 to 1.7e308	8	%lf
long double	-1.7e4932 to 1.7e4932	8	%Lf
Note: The sizes in this figure are for 32 bit compiler			

www.ecti.co.in

ENVISION Computer Training Institute

Pointers

- Variables that store memory addresses

Declaration

- <variable_type>* <name>;
  - int* ptr; //declare ptr as a pointer to int
  - int var = 77; // define an int variable
  - ptr = &var; // let ptr point to the variable var

Dereferencing Pointers

- Accessing the value that the pointer points to

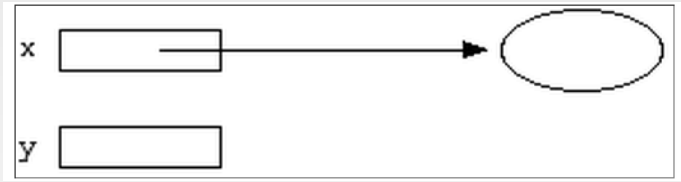
- Example:
  - double x, *ptr;
  - ptr = **&**x; // let ptr point to x
  - *****ptr = 7.8; // assign the value 7.8 to x

Pointer Example

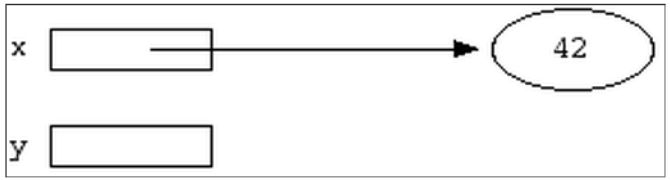
int* x;

int* y;

int var; x = &var;

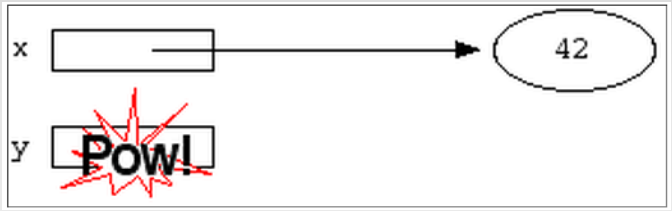


*x = 42;

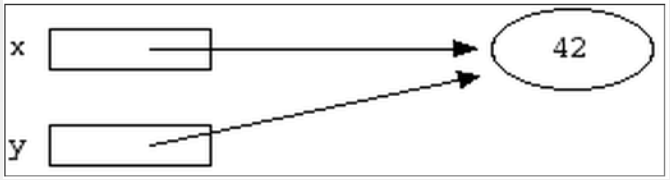


Pointer Example

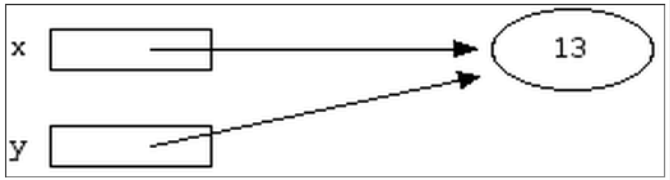
*y = 13;



y = x;

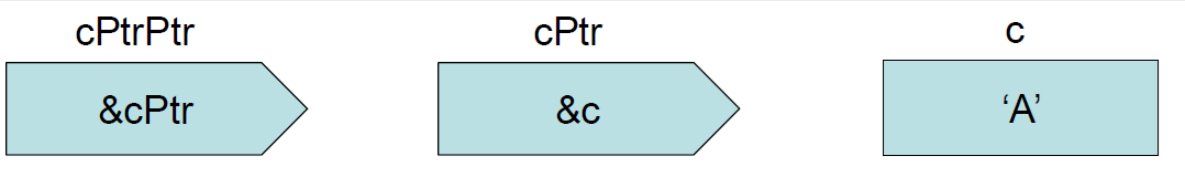


*x = 13; or
*y = 13;



Pointers to Pointers

char c = ‘A’ char* cPtr = &c char** cPtrPtr = &cPtr



Loops

```
int i;                  //initialize outside the loop unlike C++
for(i=0;i<10;i++){
}

int i=0;                //initialize outside the loop unlike C++
while(i<10)
{
        i++;
}
```

Functions

Function Name
Return Type
Arguments/Parameters
Function Body

```
int func (int a)
{
    //function body
    return 0;
}
```

Parameter Passing

- **Pass by value**
  - The data associated with the actual parameter is copied into a separate storage location assigned to the formal parameter.
  - Any modifications to the formal parameter variable inside the called function or method affect only this separate storage location and will therefore not be reflected in the actual parameter in the calling environment

Parameter Passing

```
int add(int a, int b)
{
    return a+b;
}
void main()
{
    int x=4,y=8;
    int z = add(x,y);
    printf("%d",x);
}
```

Parameter Passing...

- **Pass by reference (???)**

  The formal parameter receives a pointer to the actual data in the calling environment. Any changes to the formal parameter are reflected in the actual parameter in the calling environment.

```
void swap (int* a, int* b)
{
  int tmp = *a;
  *a = *b;
  *b = tmp;
}
```

```
void main()
{
   int a = 1;
   int b = 2;
   printf ("before swap a = %d\n", a);
   printf ("before swap b = %d\n", b);
   swap (&a, &b);
   printf ("after swap a = %d\n", a);
   printf ("after swap b = %d\n", b);
}
```

Task 1

Create a function such that it takes three numbers ‘a’, ‘b’ and ‘c’ as arguments, computes a^b and store the results in ‘c’. It should not return any value. Call this function from main() and print the answer in main().

Hint: pass by reference (?)

Hint: you may want to see the pow function [check the return type and library] (or compute the exponent yourself <- better)

Task 1 Solution

```
#include <stdio.h>
#include <math.h> //library import
void exponent(int a, int b, double* c)
{
        *c=pow(a,b); //pow returns a pointer
}
int main (void)
{
        int a=2;
        int b=2;
        double z;
        exponent (a, b, &z);
        printf ("%f", z);
        return 0;
}
```