PLAN 396
Lecture 6

Dr. Hossen Asiful Mustafa

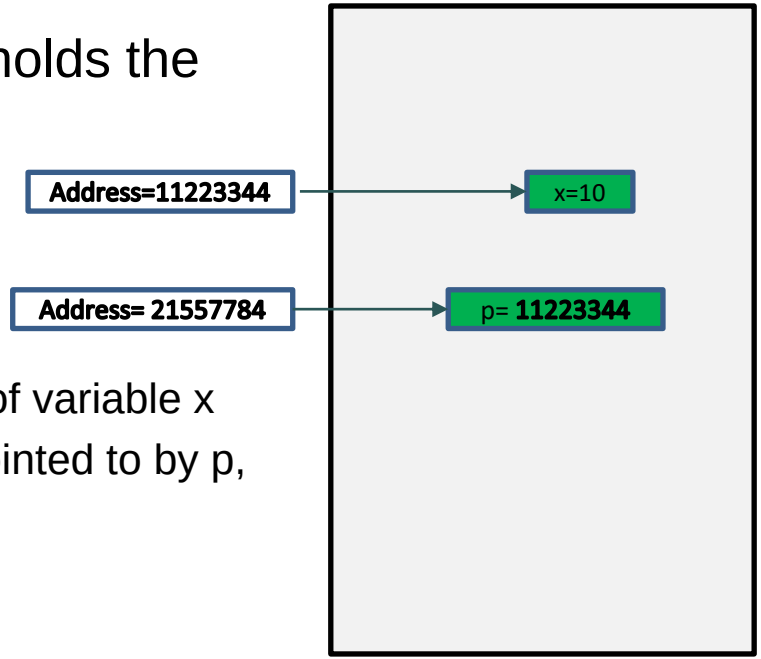
https://hossenmustafa.buet.ac.bd

Pointers

- A regular variable has 2 properties:
  - A value, and
  - An address
  - int x
  - x = 100
    - x denotes the value of the variable x, i.e., 100
    - &x denotes the address of the variable x in memory
    - The address may change each time the program is executed

Pointers

- A pointer variable is a variable which holds the address of another variable as value:
  - int x
  - int *p
  - p = &x;
    - p denotes the value which is address of variable x
    - *p denotes the value of the variable pointed to by p, i.e., the value of x
    - *p = 10 is same as x = 10
    - p is analogous to &x



Pointers

- int x = 1, y = 2, z[10];
- int *ip; // ip is a pointer to int
- ip = &x; // ip now points to x
- y = *ip; // y is now 1
- *ip = 0; // x is now 0
- ip = &z[0]; // ip now points to z[0]
- *ip = 20; // z[0] is now 20

Pointers

- Multiple pointers can point to the same variable
- Example:

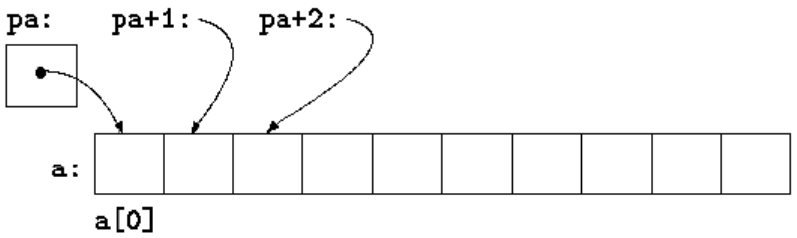
```
int *p, *q;
int num = 100;
p = &num;
q = &num;
```

- What will be the output of the following:

```
*p = 200;
printf("%d -- %d -- %d\n", num, *p, *q);
```

Pointers and Arrays

- Pointers can be used to traverse an array
- int a[10]
- int *pa;
- pa = a;
- *(pa+1) is a[1]



Functions

- A function provides a convenient way to encapsulate some computation
  - it can then be used without worrying about its implementation.
- With properly designed functions, it is possible to ignore how a job is done; knowing what is done is sufficient.
- Example:
  - printf, scanf
- A function name generally starts with a verb

Function Definition

- A function definition has this form:

```
return-type function-name(parameter declarations, if any)
{
    declarations
    statements
}
```

Example

```
#include <stdio.h>
int power(int m, int n); // forward declaration
/* test power function */
int main() {
	int i;
	for (i = 0; i < 10; ++i)
		printf("%d %d %d\n", i, power(2,i), power(-3,i));
	return 0;
}
/* power: raise base to n-th power; n >= 0 */
int power(int base, int n){
	int i, p;
	p = 1;
	for (i = 1; i <= n; ++i)
		p = p * base;
	return p;
}
```

Arguments - Call by Value

- In call by value, the called function is given the values of its arguments in temporary variables rather than the originals.
- Change in the argument doesn’t affect the original variables
- Example
  - Function: int power(int base, int n)
  - If we call,

    x = power(b, num)

  change in base doesn’t affect b.

Arguments - Call by Reference

- In call by reference, the called function is given the address of the original variables
- Change in the argument variable affect the original variables
- Example
  - Function: int power(int *base, int *n)
  - If we call,

    x = power(&b, &num)

  change in base will affect b because b and base is the same instance.

Arguments - Call by Reference

- If the argument is an array, then it is always passed as reference
- Example

  int toUpper(char str[ ]) or int toUpper(char *str)
- Classic example of call by reference is swap function

Class Assignment

- Write a program named assignment9.c
- The program should take a string as input and save it in an array of char.
- The program should use a pointer to count total number of vowels, consonants, numbers and symbols of a string.
- The program should output the counts.
- Example:
  - Input = “This is C programming class of July, 2021!”
  - Output: vowels =8
    consonants=21,
    numbers=4
    symbols=2

Mid Quiz

- Syllabus: Lecture 1 – 6
- Date: 21/12/2021
- Written Exam