Function Pointer

Alon Efrat

Computer Science Department
University of Arizona

Pointer to function = name of a function

- int add(int x, int y) { return x+y }
- Writing y=add(3,19) is like writing y= (*add)(3,19)
- So what ???
- We can pass names to functions as arguments

Example

```
int add( int x, int y) { return x+y ;  }
int mul(int x, int y) { return x*y ; }

int DoSomething( int x, int y , int (*opt)(int x, int y))
{     return (*opt)(x,y);  }

main(){
      printf("%d",    DoSomething(8,100, add ) )
}
```

- Note: DoSomething receives 3 arguments:
  - 2 ints, and a point to a function, which receives 2 arguments, and returns an int.

Example 2

```
int add( int x, int y) { return x+y ;  }
int mul(int x, int y) { return x*y ; }

main(){
  int (*ptr)(int, int);
  if(condition) ptr = add; else ptr=mul ;
  (*ptr)(4,19);
}
```

Function Pointer

```
int (*func)(int foo);
```

- func: a function pointer
- "func" points to a function that takes an int as parameter and returns int
- The base type of "func" is a function that takes an int and returns an int
- Compare: `int *func(int foo);`

5

qsort

```
void qsort(void *base, size_t nel, size_t width,
           int (*compar) (void *, void *));
```

- qsort() routine in the C library
- The data items to be sorted can be of any type and size. How does qsort compare them during the sorting?
- base: address of the arrary of elements to be sorted
- nel: number of elements in the array
- width: size (in bytes) of each element
- compar: pointer to the compare function

6

Object

```
struct __db {
    char *filename;
    int type;
    int (*get) (struct __db *dbp, void *key, void *data);
    int (*put) (struct __db *dbp, void *key, void *data);
    … …
};
```

- Wrap data and functions in a structure
- Need to initialize the function pointers when initializing the object
  - struct __db dbp;
  - dbp.get = my_get_func;

7