C
Functions, Arrays, Pointers

COEN 10

C - Lecture 9

Function Arguments

★ When passing a simple argument to a function

- ◎ a copy of the value is passed and changes to the argument do not affect the original value

Function Arguments

★ Example

```
x = max (a, b);
...
int
max (int x, int y)
{
        if (x > y)
            return x;
        else
            return y;
}
```

Functions, Arrays, and Pointers

★ When passing an array (or string) as an argument to a function

- ◎ the address is being passed

Functions, Arrays, and Pointers

★The function receiving the array (or string)

◎needs to receive the address in a pointer

◎has access to the original array (or string) and is able to change it

Functions, Arrays, and Pointers

★Example

◎Call

```
total = sum (array);
```

◎Prototype

```
int sum (int *);
or
int  sum (int []);
```

Functions, Arrays, and Pointers

★Example

◎Definition

```
int
sum (int *arg)  //  or: sum (int arg[])
{
    int i, sum = 0;
    for (i = 0; i < SIZE; i++)
        sum += arg[i];
    return sum;
}
```

Functions, Arrays, and Pointers

★Example in which the array or string is changed

```
void
init (int *arg)  //or: init (int arg[])
{
    int i;
    for (i = 0; i < SIZE; i++)
        arg[i] = i;
    return;
}
```

Functions, Arrays, and Pointers

- ★Important
  - ◎The size of the array is not known in the function
    - ✧sizeof of a pointer is 4
    - ✧The size of the array can be given by a constant or by another argument

Functions, Arrays, and Pointers

- ★Protecting Array Contents
  - ◎A function receiving a pointer, has access to the memory pointed by that pointer
  - ◎If a function is not supposed to change the original data, use const to protect the array

Functions, Arrays, and Pointers

- ★Example

```
int sum (const int []);
…
int sum (const int arg[])
{
   int i, sum = 0;
   for (i = 0; i < SIZE; i++)
      sum += arg[i];
   return sum;
}
```