#define (cont)

- used well, makes programs more readable
  - good usage: #define MAX_STR 100
  - poor usage: #define ONE 1
- if *Name* used multiple times in program
  - changing it only needs change in one place

Functions

A function packages a small computation

- provides ways to pass data in (parameters)
- provides ways to get data out (return value)
- contains code and *local* data objects

Examples (that you've already seen):

- scanf(), printf(), sin(), cos()

Functions provide abstraction (a VIP concept)

Functions (cont)

Functions are defined by giving

- return type, function name, parameter names/types
- and, of course, code to compute and return result

Example: function to return sum of two ints

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

Functions (cont)

Function signatures are defined by giving

- return type, function name, parameter types

Example: function to return sum of two ints

```
int add(int, int);
```

Users of the function need to know at least the signature, before they can use it.

Functions (cont)

Most functions return a result (of type)

- each function contains a return statement
- usage: return *Expression*;
- the result returned by the function is the value of the *Expression*

Functions (cont)

Functions are typically used like x = fun(y);

- this assigns the function result to variable x
- x's type must match function's return type

If a function does not return any result

- declare return type as void
- e.g. void vfun(int a) {…} // returns no result
- the function call is a statement: vfun(y);