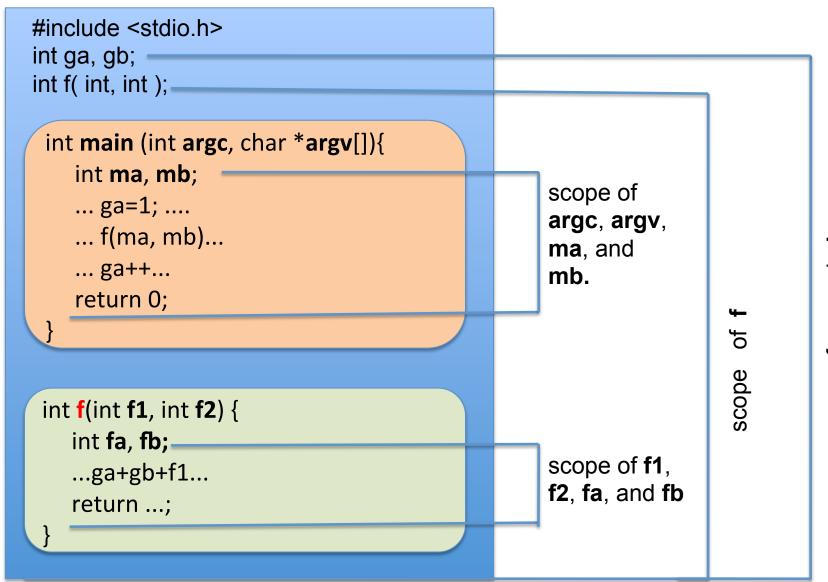
COMP10002 Workshop Week 4

Outlook:

1	Functions & scopes: Ex 6.2		
2	Pointers & Functions		
3	Arrays, pointers and arrays. Discuss Ex 7.3.		
4	Implement 6.9 and 7.4		
github this week	 insertion_sort.c and data skeleton for all exercises this week 		

Scopes, local & global variables



scope of ga and gb

```
6.2: For each of
               1
                   int bill(int jack, int jane);
the 3 marked
               2
                   double jane(double dick, int fred, double dave);
points, write
               3
down a list of all
                   int trev;
of the program-
               5
declared
               6
                   int main(int argc, char *argv[]) {
variables and
               7
                         double beth;
                         int pete, bill; /* -- point #1 -- */
functions that
               9
are in scope at
                         return 0;
that point, and
               10
for each
identifier, its
               11
                   int bill (int jack, int jane) {
type. Don't
               12
                         int mary;
                                                /* -- point #2 -- */
forget main,
               13
                         double zack;
argc, argv.
               14
                         return 0;
Where there are
               15
                   }
more than one
choice of a given
               16
                   double jane(double dick, int fred, double dave) {
name, be sure to
               17
                         double trev; /* -- point #3 -- */
indicate which
               18
                         return 0.0;
one you are
               19
                   }
referring to.
```

Function: can it have more than one output? how?

functions can have more than one outputs

```
int n;

scanf("%d", &n);

How many output scanf give in this case?

print("n= %d\n", n);

Why not &n here?
```

What is &n? what's its data type?

Data type: "pointer to int", "address of int"

int a= 18;		18	a is a location in the memory, interpreted as an int , with value of 18
		a	
int *pa;	?	18	pa is an int pointer , it can hold the address of an int
	pa	a	
pa= &a	&a	18	<pre>pa now holds the address of a, or, it "points" to a. pa is another method to access a</pre>
	pa	a	using pa alone, we can both get and change the value of a
*pa= *pa + 1;			These statements are all equivalent:
	&a	19	*pa = *pa + 1;
			a = a + 1;
	pa	a	*na - a + 1.
			*pa = a + 1; a = *pa + 1;
			6

pointers as function parameters

```
Pointers can
            1
                 int main(...) {
                   int a=2, b=4, sum, product;
be used to
            3
change the
value of
                   sAndP(a, b, &sum, &product);
            4
variables
indirectly
                   printf("sum=%d"
            6
                                                     sum);
                   printf("prod=%d",
                                                         product);
Example:
Function call
in line 4 leads
                        &sum is copied to ps, and so
to the
                           *ps is the same as sum
            11
change of
value of sum
            12
                 void sAndP(int m, int n, int *ps, int *pp ) {
and
            13
                   // now ps has the value &sum, *ps is sum
product.
                   *ps = m + n ;
                   *pp = m * n ;
            14
```

Quiz 1

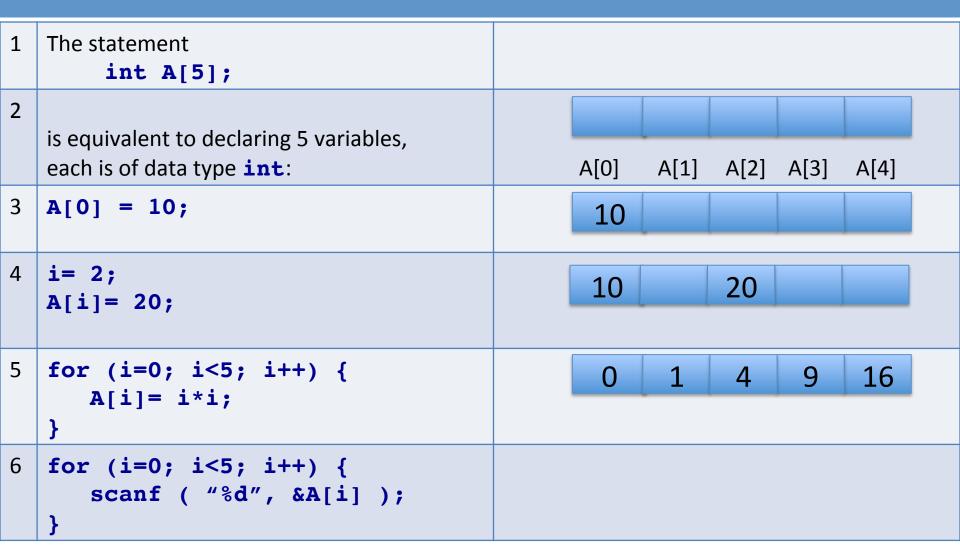
```
With the fragment:
int x=10;
f(&x);
which function below will set x to zero?
A:
                            B:
                            void f( int *n) {
int f(int n) {
   return 0;
                              &n=0;
C:
                            D:
void f (int *n) {
                            void f( int *n) {
                              *n=0;
   n=0;
```

Quiz 2

```
Given function:
void f(int a, int *b) {
    a = 1;
    *b = 2;
what are values of m and n after the following fragment:
m=5;
n = 10;
f(m, &n);
A) 5 and 10
               B) 1 and 2
                               C) 5 and 2
                                               D) 1 and 10
```

Arrays

(one-dimensional) arrays



Arrays

With declaration:

```
#define MAX N 1000
int A[MAX N] = \{10, 20, 30\}, n = 3, i = 1;
int *p= A;
```

- A[i] is just an int variable
- A is: a constant? an array? a pointer? an address?
- p is a variable, and now p can be used as a substitute for A

- Passing array to a function? We can just pass:
 - A: the array name, and
 - n: the actual number of elements in the array A.

Functions with arrays: examples

Write a function that:

- computes sum of a float array
- double the value of all elements of an int array
 How to use these functions?

Arrays: example using insertion_sort.c from github

Insertion sort algorithm:

Input: an array of n (say, int) elements

Output: the same array, but with elements re-arranged

in some (say, increasing) order.

At home: copy insertion_sort.c from github and play with it.

7.3: Modify so that only distinct values are retained after sorting Use **c7.3.c** from **github** for skeleton and hints

```
for (i= 1; i<n; i++) {
     /* swap A[i] left into correct position */
     for (j=i-1; j>=0 && A[j+1]<A[j]; j--) {
3
4
        /* not there yet */
5
       int_swap( &A[j] , &A[j+1] );
6
 H: ./7.3.exe
 Enter up to 1000 values, ^Z to end
  1 8 15 3 17 12 4 8 4
  ^ Z
  9 value read into array
 Before: 1 8 15 3 17 12 4 8 4
 After: 1 3 4 8 12 15 17
                                                15
```

Lab: Ex 6.9

Target: Read integer amounts of cents between 0 and 999. Print out the coin changes, using coins from \$2 to 5c. For example, for cents = 393, the printout should be:

200c, 100c, 50c, 20c, 20c, 5c (for simplicity, we print "200c" instead of "\$2", but you can try \$2!)

Note: valid coins are 200c, 100c, 50c, 20c, 10c, and 5c

Design: supposing we want to write a function for print changes

Target: ... For example, for cents = 393, the printout should be: 200c, 100c, 50c, 20c, 20c, 5c

Lab: Ex 6.9 – use 6.9.c from github

```
Target: ... For example, for cents = 393, the printout should be:
  200c, 100c, 50c, 20c, 20c, 5c
(for simplicity, we print "200c" instead of "$2", but you can try "$2")
Write & use functions:
int round to 5(int cents)
 that returns the value rounded off to the nearest multiple of 5.
int try one coin(int *pcents, int coin)
 that returns number of coins with face value "coin",
 does the corresponding printout,
 and also reduces the value of "*pcents" accordingly.
void print change(int cents)
  that employs try one coin to print out the coin
  changes for the amount "cents".
```

Lab

Finish implementing Ex 6.9.

Implement Ex 7.4(output values and frequencies of an array). Note that we need 2 solutions for this task: one trivial and general, and the other one which imposes some constraints on input data.

Use github as you wish:

- e7.4.Soll.c : skeleton for solution 1
- e7.4.Sol2.c : skeleton for solution 2
- Think: The 2nd solution better? When? Why?