THE SUBTASKS MAY BE FARMED OUT TO OTHER PROGRAMMERS, OTHER DEPARTMENTS, OR OTHER COMPANIES.

AN ABSTRACT, HIGH LEVEL (i.e. SYSTEMS ENGINEERING) UNDERSTANDING OF T IS NOT CONCERNED WITH THE PRECISE DETAILS OF HOW THE SUBTASKS ARE CARRIED OUT. INSTEAD ONE NEED ONLY CONSIDER HOW THE SUBTASKS WORK TOGETHER TO ACCOMPLISH T. THIS IS THE INFORMATION CONVEYED BY THE STRUCTURE CHART.

THE DIVIDE AND CONQUER TECHNIQUE CAN BE USED TO CARRY OUT ANY COMPLEX TASK (NOT JUST PROGRAMMING.)

IN COMPUTER PROGRAMMING THE SUBTASKS ARE PERFORMED BY SUBPROGRAMS OR MODULES. IN C/C++ THESE MODULES ARE CALLED FUNCTIONS.

EVERY C/C++ PROGRAM CONTAINS AT LEAST ONE FUNCTION CALLED MAIN.

```
int main(void) {
    // FUNCTION BODY
}
```

C/C++ FUNCTIONS APPEAR IN THREE CONTEXTS.

1.) FUNCTION PROTOTYPES.
2.) FUNCTION CALLS
3.) FUNCTION DEFINITIONS

FUNCTION PROTOTYPES TELL THE COMPILER HOW MANY, AND WHAT TYPE OF PARAMETERS THE FUNCTION TAKES, AS WELL AS ITS RETURN TYPE. PROTOTYPES SHOULD COME AFTER PREPROCESSOR DIRECTIVES AND BEFORE FUNCTION MAIN.

A TYPICAL PROTOTYPE TAKES THE FOLLOWING FORM:

DATA_TYPE name (DATA_TYPE VAR1, DATA_TYPE VAR2, ...);

RETURN TYPE — FUNCTION NAME — PARAMETER LIST

EX.
OUR CIRCLE PROGRAM COULD USE A FUNCTION TO COMPUTE AREA.

```
double find_area (double radius);
```

THE RETURN TYPE, AND THE PARAMETER LIST DATA TYPES, CAN BE ANY BUILT IN (OR USER DEFINED) DATA TYPE SUCH AS int, double, char, bool, OR void TO INDICATE THE ABSENCE OF A RETURN VALUE OR PARAMETER.

THE FUNCTION NAME MUST ADHERE TO THE RULES FOR C/C++ IDENTIFIERS.

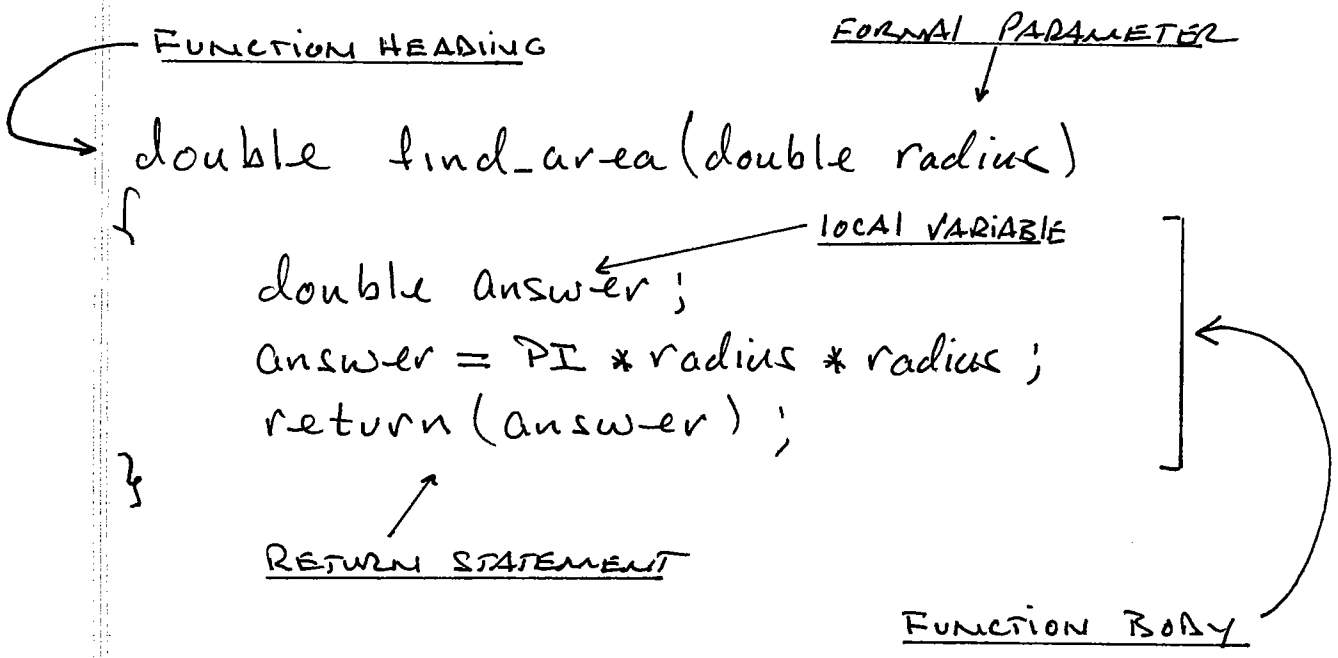
FUNCTION CALLS OCCUR WITHIN FUNCTION main, OR WITHIN SOME OTHER FUNCTION.

```
int main(void)
{
    double a, r;
    :
    a = find_area(r);
    :
}
```

FUNCTION ARGUMENT

IN THIS CONTEXT, FUNCTION find_area CAN BE THOUGHT OF AS A BLACK BOX. THE VALUE r GOES IN, AND WHAT COMES OUT IS THE VALUE OF THE EXPRESSION find_area(r), WHICH IS ASSIGNED TO a. WE DON'T KNOW WHAT IS IN THE BOX.

FUNCTION DEFINITIONS TELL THE COMPILER WHAT OPERATIONS TO PERFORM DURING THE FUNCTION CALL. THEY OCCUR AFTER FUNCTION MAIN.



THIS IS WHAT HAPPENS WHEN THE FUNCTION CALL

```
a = find_area(r);
```

IS ENCOUNTERED; SPACE IS RESERVED FOR THE FORMAL PARAMETER radius, AND THE LOCAL VARIABLE ANSWER. THE FUNCTION ARGUMENT r IS COPIED TO THE FUNCTION PARAMETER RADIUS. PROGRAM EXECUTION THEN TRANSFERS TO THE EXECUTABLE STATEMENTS IN THE FUNCTION DEFINITION, WHERE ANSWER IS ASSIGNED A VALUE

THE STATEMENT

```
return (answer);
```

CAUSES THE EXPRESSION find_area(r) TO BE EVALUATED AS answer. THE SPACE USED FOR radius AND answer IS DEALLOCATED. PROGRAM EXECUTION THEN RETURNS TO FUNCTION main, WHERE a IS ASSIGNED A VALUE.

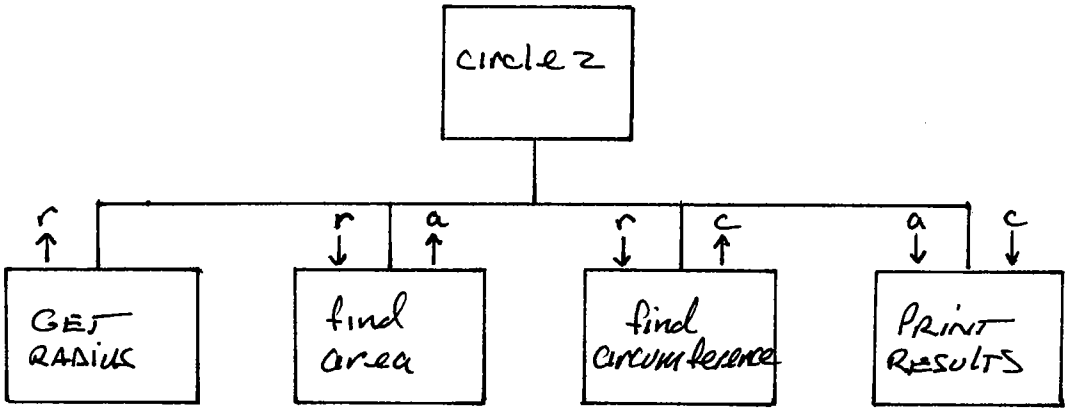
— SEE EXAMPLE: circle2.cpp —

NOTE THE local VARIABLE answer, AND FORMAL PARAMETER radius APPEAR IN BOTH FUNCTION find_area AND find_circumference. THESE ARE REALLY SEPARATE VARIABLES. IN FACT THEY ARE NOT CREATED UNTIL THE FUNCTION IS CALLED, AND ARE DESTROYED WHEN IT RETURNS.

IN THIS EXAMPLE FUNCTIONS find_area AND find_circumference WERE USED TO REFINE ALGORITHM STEPS.

```
//////////////////////////////////////////////////////////////
//   File: circle2.cpp
//   Description: Get input value for the radius of a circle,
//   print out the circumference and area.
//   Compile:  g++  -o circle2  circle2.cpp
//////////////////////////////////////////////////////////////

#include<iostream>
#include<cmath>
using namespace std;

#define  PI  (3.14159)

double find_area(double radius);
double find_circumference(double radius);

int main(void)
{
     double r,          // Input: radius of circle.
            a,          // Output: area of circle
            c;          // Output: circumference of circle

     // Get radius from user.
     cout << "Enter the circle radius: ";
     cin >> r;

     // Compute area.
     a = find_area(r);

     // Compute circumference.
     c = find_circumference(r);

     // Print out area and circumference.
     cout << "The area is " << a << ", and the circumference"
          << " is " << c << "." << endl;

     return (0);
}

double find_area(double radius)
{
        double answer;
        answer = PI*pow(radius,2);
        return answer;
}

double find_circumference(double radius)
{
        double answer;
        answer = 2*PI*radius;
        return answer;
}
```

A STRUCTURE CHART FOR circle2.cpp WOULD LOOK LIKE:



RECALL THAT THE SELECTION SORT ALGORITHM HAD FIND MAX EMBEDDED WITHIN IT. MAIN WOULD BE MORE READABLE IF WE PLACE FIND MAX IN A FUNCTION.

— SEE EXAMPLE: SelectionSort2.cpp —

NOTE THAT IN THE PROTOTYPE FOR FUNCTION find_imax:

```
int find_imax(double L[], int len);
```

ONE OF THE INPUT PARAMETERS L, IS AN ARRAY OF DOUBLES OF UNDETERMINED LENGTH.

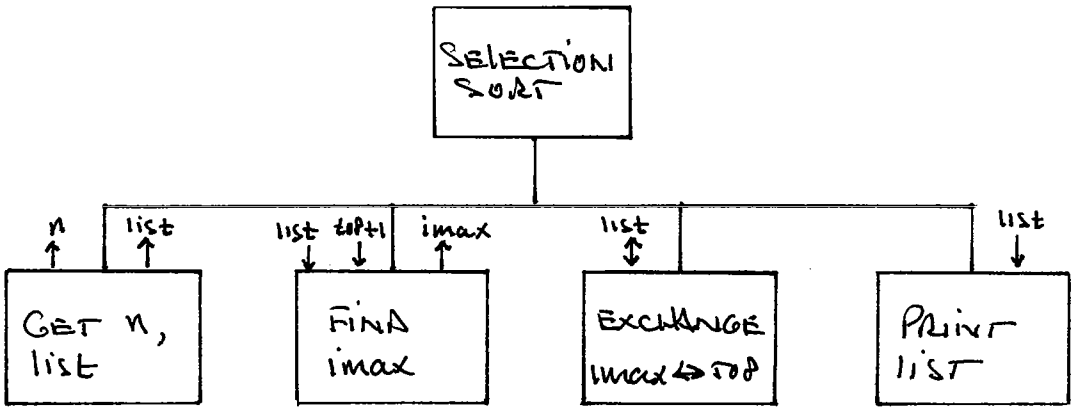
```
//////////////////////////////////////////////////////////////////////
//      File: SelectionSort2.cpp
//      Input:  list length n, and array list.
//      Output: array list in increasing order.
//////////////////////////////////////////////////////////////////////

#include<iostream>
using namespace std;

#define MAX_LIST_LEN  100

int find_imax(double L[], int len);

int main(void)
{
        int    n,                       // list length
               j,                       // loop control
               imax,                    // index of max
               top;                     // unsorted marker
        double temp,                    // temporary storage
               list[MAX_LIST_LEN];      // array to be sorted

        // Get values for n and array list.
        cout << "Enter the list length "
             << "(must be less than or equal to "
             << MAX_LIST_LEN << "): ";
        cin  >> n;
        cout << "Enter " << n << " numbers:\n";
        for(j=0; j<n; j++)
                cin >> list[j];

        // Perform selection sort on array list.
        for(top=n-1; top>0; top--)
        {
                // Find imax, index of max element in unsorted section
                imax = find_imax(list, top+1);

                // Exchange list[imax] with list[top].
                temp = list[imax];
                list[imax] = list[top];
                list[top] = temp;
        }

        // Print out sorted list.
        for(j=0; j<n; j++)
                cout << list[j] << ' ';
        cout << endl;

        return(0);
}
```

```
////////////////////////////////////////////////////////////////////////
//      find_imax
//      Input : array L, length len of section to search.
//      Output : Index i of maximum array element.
////////////////////////////////////////////////////////////////////////
int find_imax(double L[], int len)
{
        double max;
        int i, j;

        max = L[0];
        i = 0;
        for(j=1; j<len; j++)
        {
                if( L[j] > max )
                {
                        max = L[j];
                        i = j;
                }
        }
        return(i);
}
```

RECALL THAT THE COMPILER DOES NOT KNOW WHERE THE END OF THE ARRAY IS. ONE WAY TO DEAL WITH THIS WHEN PASSING AN ARRAY PARAMETER IS TO PASS THE ARRAY LENGTH AS A SEPARATE PARAMETER.

IN THIS WE ARE PASSING NOT THE LENGTH OF THE FULL ARRAY, BUT THE LENGTH OF THE SUBARRAY TO BE PROCESSED.

STRUCTURE CHART:



IN THIS EXAMPLE ONLY THE SUBTASK FIND IMAX IS IMPLEMENTED AS A FUNCTION.

NOTE THAT A STRUCTURE CHART IS NOT A FLOW CHART. SOME SUBTASKS ARE PERFORMED MANY TIMES WHILE OTHERS ARE PERFORMED ONLY ONCE.

OBSERVE THAT IN THE FUNCTION CALL

```
imax = find_imax (list, top+1);
```

WE PASS THE ARRAY list BY JUST USING ITS NAME AS THE FUNCTION ARGUMENT, WITH NO PARAMETERS.

THERE IS A FUNDAMENTAL DIFFERENCE BETWEEN THE SIMPLE PARAMETER len AND THE ARRAY PARAMETER L.

WHEN THE ABOVE FUNCTION CALL IS EXECUTED, THE VALUE top+1 IS <u>COPIED</u> INTO THE LOCAL VARIABLE len. FUNCTION find_imax MAY CHANGE THE VALUE OF len, BUT THIS HAS NO EFFECT ON THE VALUE OF THE ARGUMENT top+1 (OR top). THIS IS CALLED <u>PASSING BY VALUE</u>.

find_imax DOES NOT RECIEVE MERELY A COPY OF list HOWEVER, IT RECIEVES THE ACTUAL ARRAY, KNOWN LOCALLY AS L. IF find_imax MADE CHANGES TO L, THIS WOULD CHANGE THE VALUES IN list. THIS IS CALLED <u>PASSING BY REFERENCE</u> OR <u>PASSING BY ADDRESS</u>.