The University Of Sheffield.

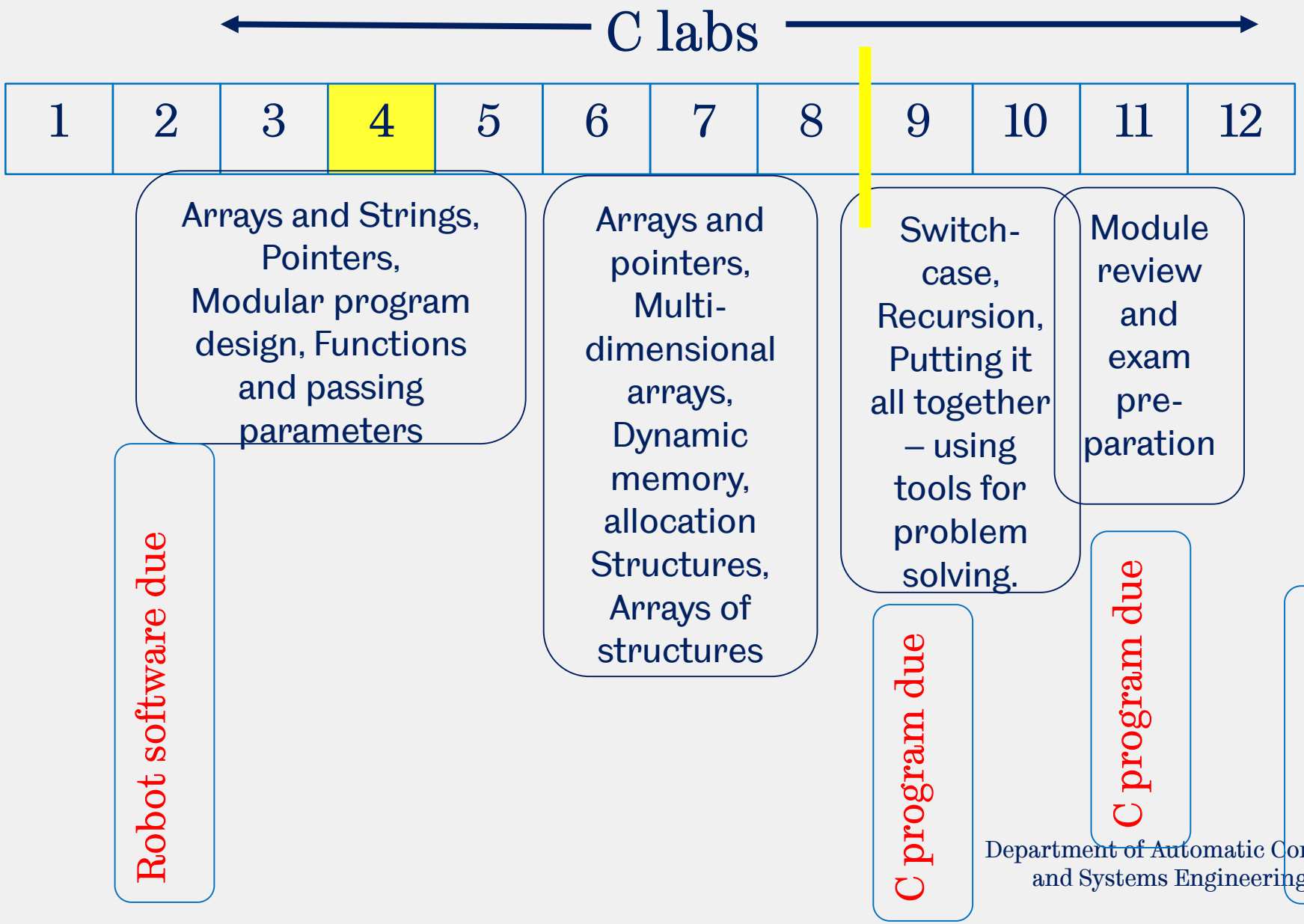
ACS126 Computing and Systems Design

Lecture 12. Modular Program Design and Functions

Department of Automatic Control and Systems Engineering

The University Of Sheffield.

Module roadmap



Learning Outcomes

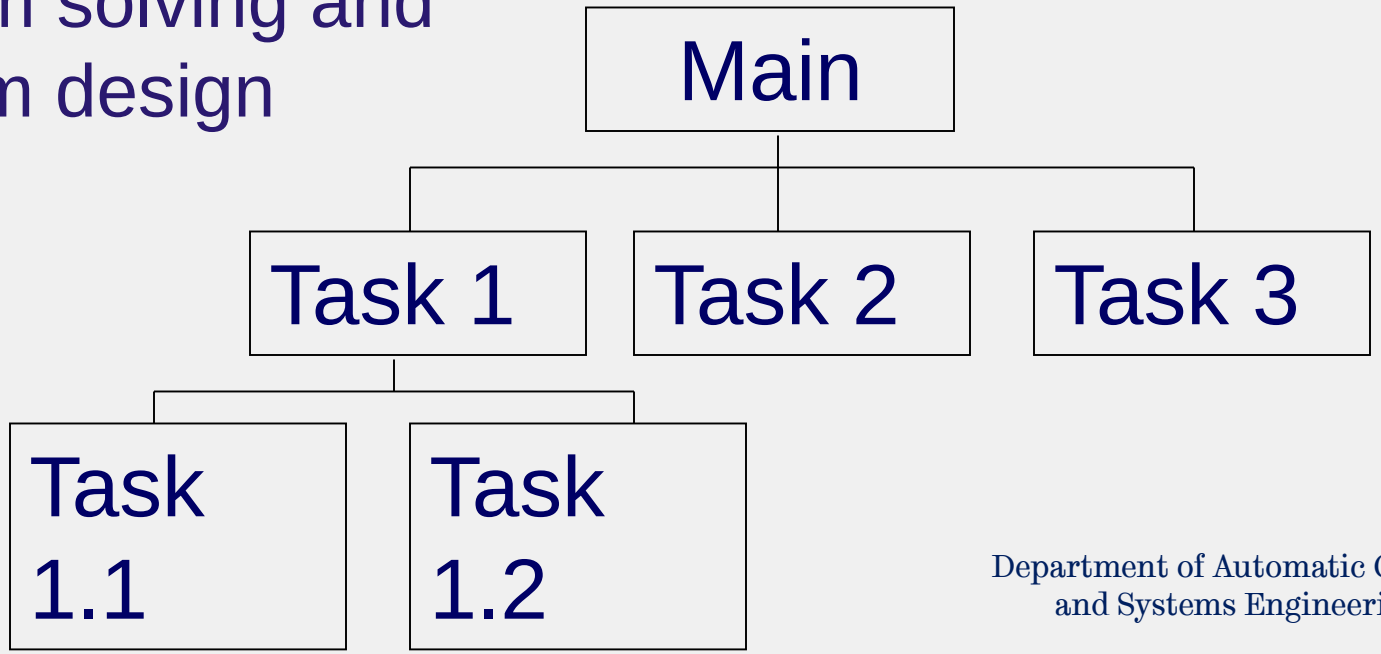
By the end of this session, students should be able to:

- Solve C software problems systematically and hierarchically, which is reflected in the program design.
- Determine the scope of a variable declared in a C program
- Construct and use C functions using:
  - Pass by value
  - Pass by address

The University Of Sheffield.

Modularisation

- Subdivision of a large task into manageable chunks.
- The chunks may be simple tasks or complex tasks.
- **Top- down design**: a systematic approach to problem solving and program design



The University Of Sheffield.

Advantages in using Modularisation

- Easy to test
- Make it easier for software to be developed by more than one person
- Easy to change
- Readable
  - Reader of main() code can see that all the necessary tasks have been included in the program by looking at the references to the modules

Given a problem, how do we produce a modular solution?

- First determine what are the tasks that must be done.
- Look for verbs/action phrases in a problem description.
- If small tasks are related they can be grouped into a single module.

The University Of Sheffield.

Example

- What were the tasks your robot software needed to carry out? What functions did/could you use?

The University Of Sheffield.

Modularisation – C functions

- A C function is a module.
- A C function is a block of code with a name. The name allows the function to be used from more than one point in a program, or even from different programs.
- A **block** of code is zero or more programming statements delimited by braces: { …. }
- Note that main() is delimited by braces, and main() is a function. It is the top level function in the top-down design of a program.
- A good function deals with a single processing task.
  - Eg, what would you expect int square(int a) to do?

The University Of Sheffield.

FUNCTIONS

Function Prototypes/Declarations

Function Calling

Function Definitions

-> **Passing by value**

Department of Automatic Control
and Systems Engineering

The University Of Sheffield.

Recap: Function Prototype

- Creates the function:
  - Goes after #include statements, and before `main( )`
  - Gives the definition of 3 aspects of the function:
    1. the type of the return value,
    2. the function name and
    3. the parameter list.

e.g. `int max(int a, int b);`

Always ends with a semi-colon

Type of return value

Function name

Parameter list

The University Of Sheffield.

Recap: Function Calling

- The function is called from within main()

e.g.

```
#include <stdio.h>
int max(int a, int b);
main(){
      int int1=2, int2=5, maxValue;
      maxValue= max(int1, int2);
…
}
```

The University Of Sheffield.

Recap: Function Definition

- Creating a function- the function body.
- The function body is a block of program code which performs some task(s).
- The function body is headed by a line containing the function return value, name and parameter list.
- The function body header looks like the function prototype, except the function prototype concludes with a ";", and the function body header does not. The program statements that follow the function body header are delimited by { } (just as in main).
- The function body **cannot** go within the { } of main.

```
#include <stdio.h>
int max(int a, int b);
main(){
      int int1=2, int2=5, maxValue;
      maxValue= max(int1, int2);
      printf("Max value is %d.",maxValue);
} //end main
int max(int a, int b){
      if(a>b)
            return a;
      else
            return b;
}
```

Function Prototype

Function Call

Function Definition

The University Of Sheffield.

SCOPE OF VARIABLES

Local variables versus Global variables

Department of Automatic Control
and Systems Engineering

Variables

- Scope of variables
  - Global: variable defined throughout entire program
    - Declared before main().
  - Local: variable defined in a limited section of the program
    - Declared in main(), and in any other functions.
- Within main or a function, a variable can be given a value via:
  - Parameter list
  - Reading from the keyboard/file/other device
  - Setting values in an assignment statement (eg, a=1)

```
/* program to illustrate scope, written March 2017, TB*/
#include <stdio.h>
void swap(int a, int b);
main()
{ int i=100, j=200;
  printf("before call to swap in main, i is %d, j is
  %d\n",i,j);
  swap(i,j);
  printf("after call to swap in main, i is %d, j is
  %d\n",i,j);
}
/* function void swap(int a, int b) swaps a and b internal to swap */
void swap(int i, int j)
{ int temp;
  temp=i;
  i=j;
  j=temp;
  printf("in swap function, i is %d, j is %d\n",i,j);
}
```

i and j are local to main function

i and j are local to swap function

```
/* program to illustrate global vars and scope, written March 2017, TB */
#include <stdio.h>
int a,b;  /* global vars */
void swap();
main()
{   a=100; b=200;
    printf("before call to swap in main, a is %d, b is
   %d\n",a,b);
    swap();
    printf("after call to swap in main, a is %d, b is
   %d\n",a,b);
}
/* function void swap() swaps a and b, global vars */
void swap()
{    int temp;
     temp=a;
     a=b;
     b=temp;
     printf("in swap function, a is %d, b is %d\n",a,b);
}
```

Alert: Global variables are dangerous. Avoid them!

The University Of Sheffield.

USING POINTERS AS PARAMETERS IN FUNCTIONS

Pass by address

Department of Automatic Control
and Systems Engineering

The University Of Sheffield.

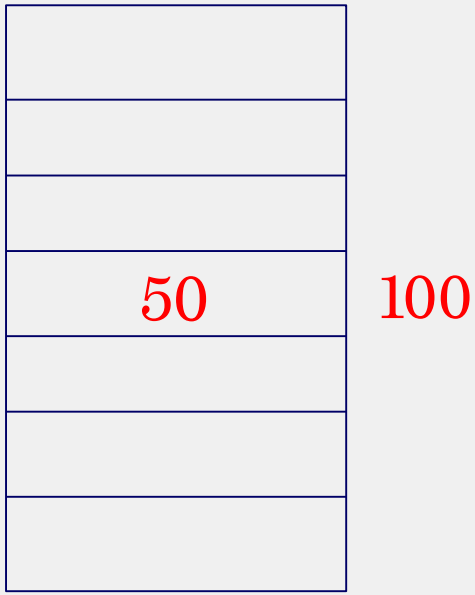
Pass by Value

- The functions we have encountered use **pass by value**.
  - The function makes a local copy of the parameter in memory, and
  - the value of the variable in the calling function remains unchanged.

The University Of Sheffield.

Pass by Value

Memory in main( )

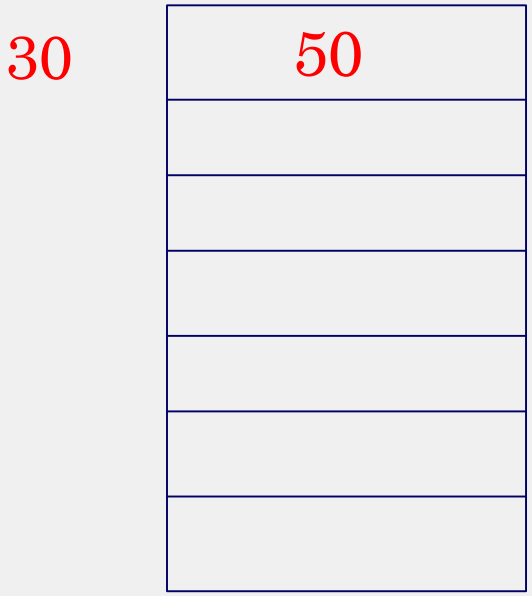


Let's say in main() you have int A=50, stored at memory loc 100.

Function Call

F(A)

Memory in F(int A)



If you then use A in F, you would have access to the value of A only.

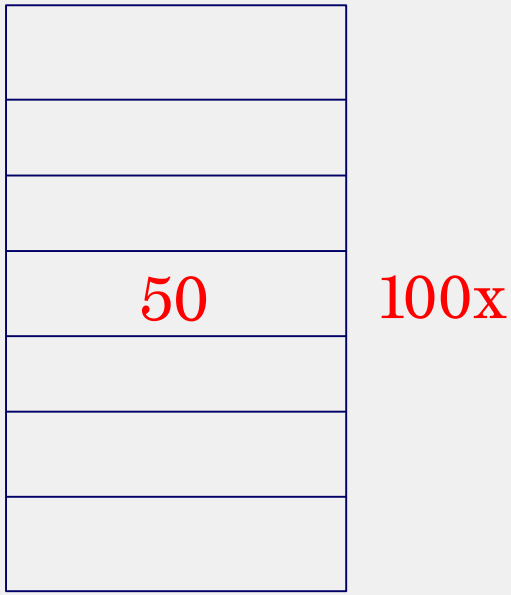
Pointers as Parameters

- Functions can also use **pass by address**.
  - To pass a parameter by address, the parameter must be a pointer to a variable.
  - The value of a parameter changed in the function also changes the value of the variable in the calling function.

The University Of Sheffield.

Pass by Address

Memory in main( )



Let's say in main() you have int A=50, stored at memory loc 100.

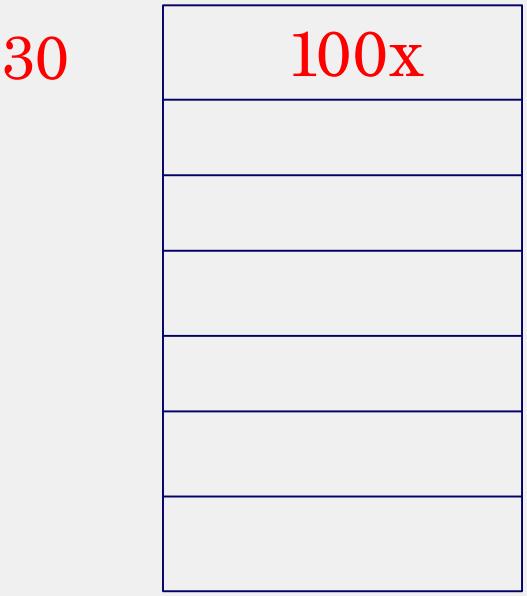
Function Call

F(&A)

Address of A

dereferencing

Memory in F(int *A)



If you then use *A in F, you would have access to the contents of the memory location 100x.

Function Prototype

```
void swap(int *a, int *b);
```

- `int *a, int *b` are pointers to type int
- Function prototype goes before main.

Function Calling

```
swap(&i,&j);
```

- The parameter list consists of the addresses of i and j.
- Function is called within main() or another function.

The University Of Sheffield.

Function Definition

```
void swap(int *a, int *b)
{   int temp;
    temp=*a;
    *a=*b;
    *b=temp;
    printf("in swap function, a is
  %d, b is %d\n",*a,*b);
}
```

Same as function prototype

Dereferencing

Example

```
/* program to illustrate pass by address, written March 2017, TB */
#include <stdio.h>
void swap(int *a, int *b);
main()
{   int i=100, j=200;
    printf("before call to swap in main, i is %d, j
       is %d\n",i,j);
    swap(&i,&j);
    printf("after call to swap in main, i is %d, j
       is %d\n",i,j);
}
/* function void swap(int *a, int *b) swaps contents of a and b */
void swap(int *a, int *b)
{   int temp;
    temp=*a;
    *a=*b;
    *b=temp;
    printf("in swap function, a is %d, b is %d\n"
      ,*a,*b);
}
```

```
#include <stdio.h>
void roundFloat(float v, int *r);
main()
{
     float val1=0.3, val2=-24.57;
     int v1,v2;
     roundFloat(val1,&v1);
     roundFloat(val2,&v2);
     printf("%d %d\n",v1,v2);
}
/* function roundFloat
void roundFloat(float value, int *r)
{
   float a,b;
   a=value;
   if(a>=0.0)
        b=value+0.5;
   else
        b=value-0.5;
  *r=b;
 }
```

Test: What will be printed by this program?

Characteristics of a Good Function

- It performs a single task, and it is clear what that task is.
- Only one way in
- Only one way out
- It has a descriptive name.
- The interface (header) to the function is well defined.

Note that the interface to a function using global variables is badly defined.

The University Of Sheffield.

Learning Outcomes

By the end of this session, students should be able to:

- Solve C software problems systematically and hierarchically, which is reflected in the program design.
- Determine the scope of a variable declared in a C program- global versus main
- Construct and use C functions using:
  - Pass by value
  - Pass by address