The University Of Sheffield.

Review of last week's content

- What is an array?
- What are the main differences between an array and a string?
- Comment the C code to explain what is happening.

The University Of Sheffield.

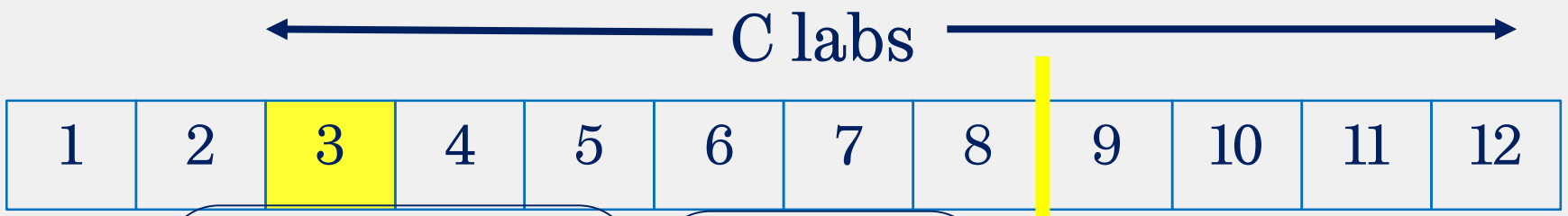
ACS126 Computing and Systems Design

Lecture 11. Pointers

Department of Automatic Control
and Systems Engineering

The University Of Sheffield.

Module roadmap



Arrays and Strings, Pointers, Modular program design, Functions and passing parameters

Robot software due

Arrays and pointers, Multi-dimensional arrays, Dynamic memory, allocation Structures, Arrays of structures

C program due

Switch-case, Recursion, Putting it all together – using tools for problem solving.

Module review and exam pre-paration

C program due

Exam

The University Of Sheffield.

Learning Outcomes

At the end of this session, you should be able to:

- Explain what a pointer is,
- Use pointers in simple C programs
- Use pointers with arrays and strings

The University Of Sheffield.

What is a Pointer?

- A pointer is a variable containing the address (in memory) of a variable, i.e. a pointer holds the memory location where a variable is stored.
- A pointer is declared to point to a particular type of variable. A declaration of a pointer gives the type and the variable name preceded by an '*'.
- Examples:

```
int *ptrA;     // ptrA is a pointer to an int variable
char *ptrB;    // ptrB is a pointer to a char variable
float *ptrC;   // ptrC is a pointer to a float variable
```

The University Of Sheffield.

Working with Pointer Variables

```
int *ptrA;      /* ptrA is a pointer to an int variable */
int A;
```

...would result in

We don't yet know where ptrA points.

The names A and ptrA imply that we want ptrA to point to A, but we need to explicitly tell the C compiler this.

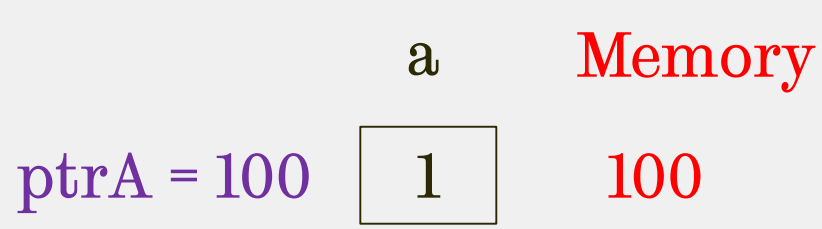
The & address operator is used to get the address of A. We can then set ptrA equal to the address of A:

```
ptrA = &A;
```

The University Of Sheffield.

Using Pointers

```
int a=1;
int *ptrA;
ptrA = &a;
```



To get or set the contents of the memory location which a pointer points to, use the dereferencing operator *.

```
printf("Element of variable a is %d\n", *ptrA);
```

```
*ptrA = 25; /* sets the contents at memory location ptrA to 25 */
```

```
int B = *ptrA; /* the integers A and B now both equal 25 */
```

The University Of Sheffield.

Working with Pointer Variables

Notice that:

- in a declaration, * is used to declare a pointer (eg, int *ptrA),
- in an assignment statement, * is used to dereference a pointer (*ptrA = 25)
  - Dereference means to get the contents of the memory location.

The University Of Sheffield.

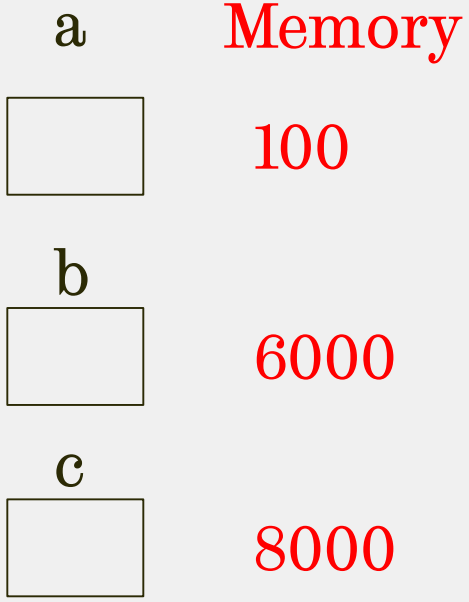
Example: Using Pointers

```
int a=1;
int *ptrA;
ptrA = &a;

float b=3.5;
```

//Add your code for pointer ptrB

```
char c='!';
```

//Add your code for pointer ptrB



Dereferencing: use

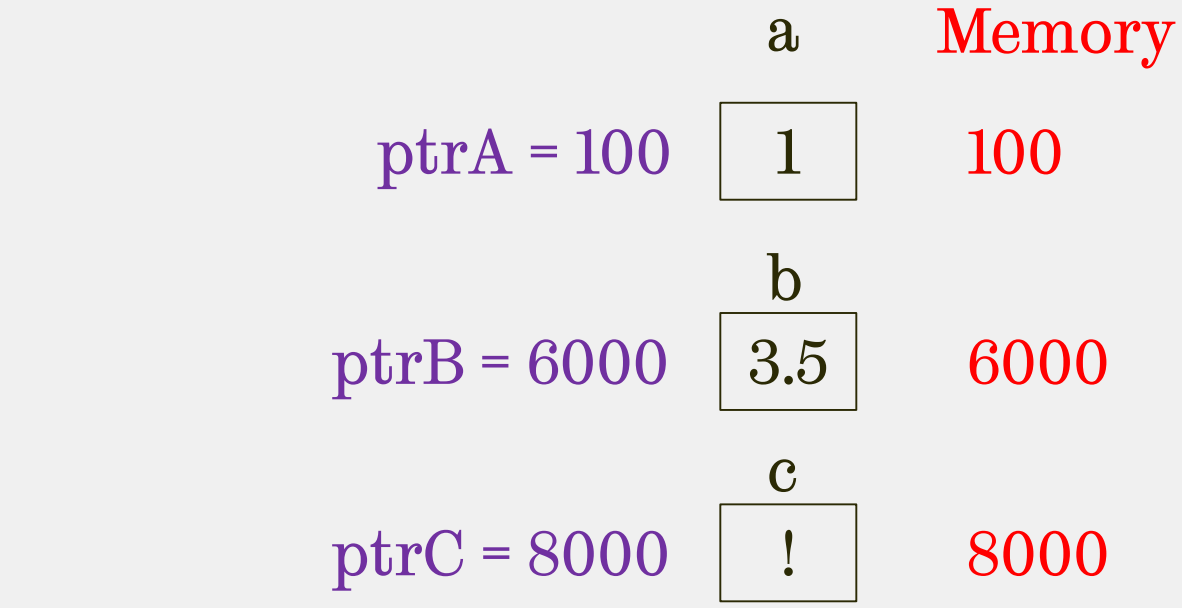
`*ptrA,*ptrB,*ptrC.`

What are ptrA, *ptrA, ptrB, *ptrB, ptrC, *ptrC?

The University Of Sheffield.

Solution: Using Pointers

```
int a=1;
int *ptrA;
ptrA = &a;

float b=3.5;
float *ptrB;
ptrB = &b;

char c='!';
char *ptrC;
ptrC = &c;
```



Dereferencing: use `*ptrA, *ptrB, *ptrC.`

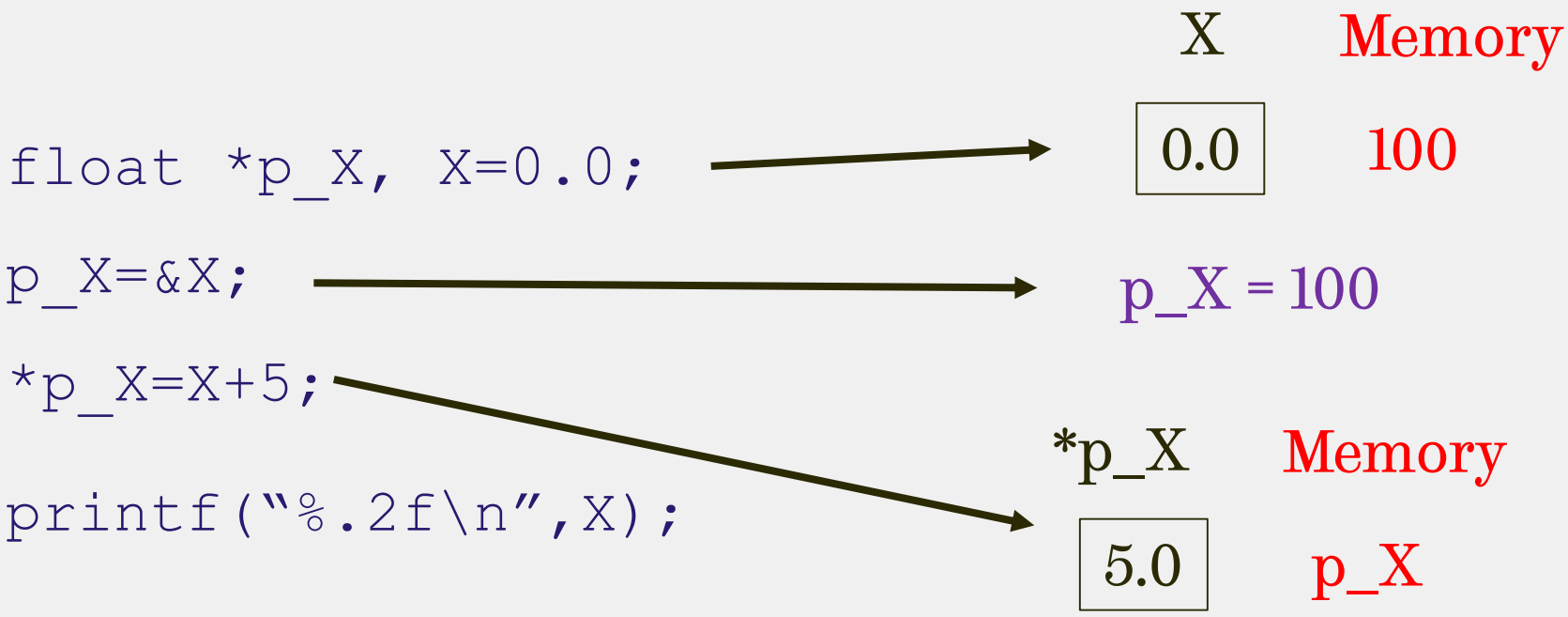
*ptrA = 1, *ptrB=3.5, *ptrC='!'

Example: Using Pointers

```
#include <stdio.h>
main(){

    char c = 'A';    /* from the ASCII table, this is 41 hex. */
    char *ptrC;
    ptrC = &c;

    printf("Element is %c\n",c);
    printf("Address is %x\n",ptrC);
    printf("Using pointers:\n");
    printf("Element is %c and in hex it is %x\n",*ptrC,
                                                 *ptrC);
}
```

The University Of Sheffield.

Example



What would be printed?

5.00

The University Of Sheffield.

Working with Pointer Variables

- When would you want to use pointers to access memory locations, rather than manipulating variables directly?
  - Mainly used when passing parameters to functions.
  - Can be used as part of dynamic allocation of memory.
  - Can be used as an alternative way of accessing array contents.
- Pointer variables can be used with +, -, ++, -- (that is, you can add to, subtract from and increment and decrement an address)

The University Of Sheffield.

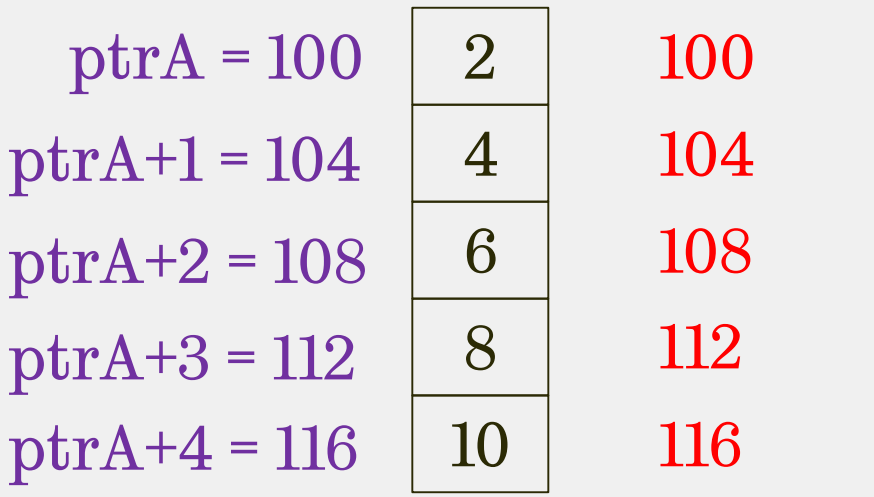
Pointers and Arrays

- An array is stored in a block of contiguous memory locations.
- **The array name on its own is a pointer to the first location in the array (eg, A[0]).** This can be used in assignment statements **iff** it is assigned to a pointer variable of the same base type.

The University Of Sheffield.

Using Pointers with Arrays

```
int a[]={2,4,6,8,10};
int *ptrA;
ptrA = a;
```



Dereferencing: use `*ptr`.

e.g.

```
for(i=0;i<5;i++){
     printf("The elements of the array are
     %d\n", *(ptrA+i)); }
```

Example

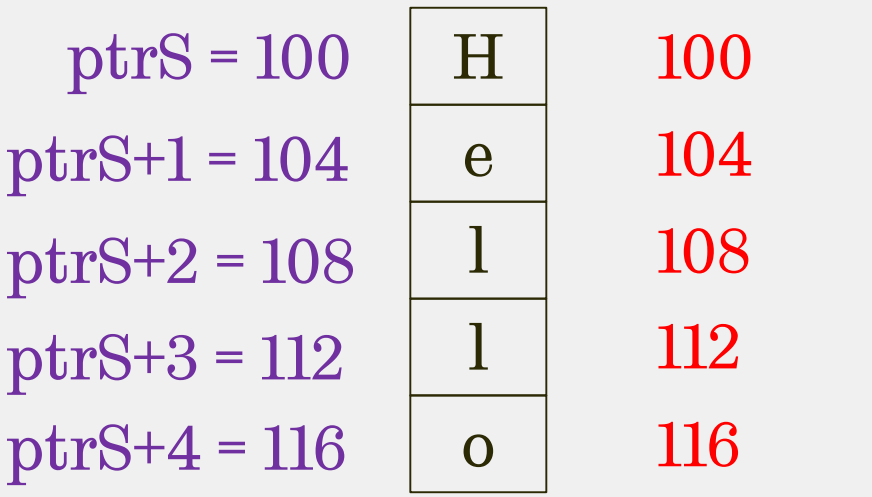
- What gets printed to the screen?

```
int array[]={0,2,4,8,16},*pA, *pB;

pA=array;

pB=array+3;

printf("array: %d %d %d %d %d; *pA %d,
  *pB %d", array[0], array[1],
  array[2], array[3], array[4], *pA,
  *pB);
```

The University Of Sheffield.

Using Pointers with Strings

```
char s[6]="Hello";
int *ptrS;
ptrS = s;
```



e.g.

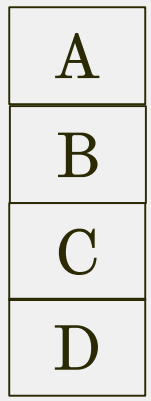
```
printf("The string is %s", ptrS);
```

The implementation of printf() sees the "%s", assumes that the corresponding argument is a pointer to char, and uses that pointer to traverse the string and print it.

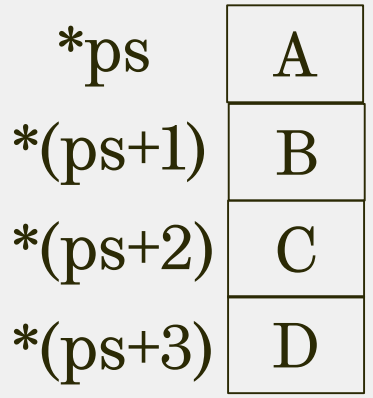
The University Of Sheffield.

What Would be Printed?

```
char string[25],*ps;

scanf("%s",string);
/* say you enter ABCD */

ps=string;

printf("%c\n",*(ps+2));
```



ps = 100



Example Program with Pointers

```
#include <stdio.h>
main()
{
  int list1[10]={10,20,30,40,50,60,70,80,90,100};
  char list2[10];
  int *plist1;
  char *plist2;
  int i;
  plist1=list1;
  plist2=list2;
  scanf("%s",list2); //assume you enter abcdefghij
  for(i=0; i<10; i++)
  {
     printf("%c %d\n",*(plist2+i), *(plist1+i));
  }
  printf("%s\n", plist2);
}
```

The University Of Sheffield.

Learning Outcomes

At the end of this session, you should be able to:

- Explain what a pointer is,
  - A pointer is a variable containing the address (in memory) of a variable.
- Use pointers in simple C programs

```
int a=1;
int *ptrA;
ptrA = &a;
```

dereferencing operator:

```
*ptrA = 2;
```

- Use pointers with arrays and strings

```
int a[]={2,4,6,8,10};
int *ptrA;
ptrA = a;
```

```
char s[6]="Hello";
char *ptrS;
ptrS = s;
```