

Computer System Fundamentals

DPST1092

Revision

Revision: What you need to know...

Overview

- Define variable i.e. `int x = 10;`
- Define and use arrays `int arr[10];`
- for loops, while loops and if statements
- Define and use function
- Pointers
- Dynamic Memory (Malloc)
- **NO LINK LISTS IN DPST1092**

I must do my revision
I must do my revision
I must do my revision
I must do my revision
I must do my revision



Arrays and Functions

What you need to know

- Must know how to define an array
- Must know how to access elements
- Must know how to define and call a function

Loops

- Must know how to use for loops, while loops and if statements

```
#include <stdio.h>

int sum(int x[], int length){
    int result = 0;
    for(int i = 0; i < length; i++){
        result = result + x[i];
    }
    printf("%d\n", result);
    return result;
}
```

```
int main(){
    int length = 5;
    int arr[5];

    for(int i = 0; i < length; i++){
        arr[i] = i;
    }

    sum(arr, length);
}
```

Pointers (*)(&)

What you need to know

- Define a pointer
- Use a pointer
- Change where the pointer is pointing to
- Access its address
- Define different types of pointers
- Pass by reference and pass by value

```
#include <stdio.h>
```

```
int main() {  
    int num = 10;  
    int *ptr;  
  
    // Assigning the address of num to ptr  
    ptr = &num;  
  
    // Accessing the value of num using ptr  
    printf("Value of num: %d\n", *ptr);  
  
    // Changing the value of num using ptr  
    num = 20;  
    printf("New value of num: %d\n", num);  
  
    //printing out the address of your pointer %p  
    printf("The address that your pointer is pointing to is: %p\n", &num);  
  
    return 0;  
}
```



Pointers (int vs char)

- **int *** is used when working with arrays or variables of type int
- **char *** is often used for strings, single characters, or binary data
- **int ***, ptr++ increments the pointer by the size of an int (4 bytes)
- **char ***, ptr++ increments the pointer by the size of a char (1 byte).

```
#include<stdio.h>
```

```
int main()
```

```
{
```

```
    int arr[3] = {1592, 1412, 5165123};
```

```
    int *int_ptr = arr; ← int pointer
```

```
    printf("%d\n", *int_ptr);
```

```
    char *char_ptr = (char *)arr; ← char pointer
```

```
    printf("%d\n", *char_ptr);
```

```
    int_ptr++;    // Moves ahead by 4 bytes (to the next int)
```

```
    char_ptr++;   // Moves ahead by 1 byte (to the next byte)
```

```
}
```

```
z3544032@vx16:~/testweek1$ ./a.out
```

```
1592
```

```
56
```

```
z3544032@vx16:~/testweek1$
```

WHY??



Pointers

```
int arr[3] = {1592, 1412, 5165123};
```



1592 in hex is 0x638 (stored in little endian, talk about it later!!!)

```
int *int_ptr = arr;
```

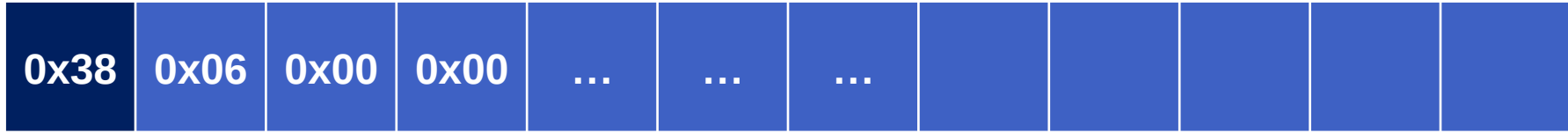
Pointers are always a 4 bytes address (e.g. 0x80808080)



When you define an int pointer you will be returning 4 bytes

Pointers

`char *char_ptr = (char *)arr;` *Pointers are always a 4 bytes address (e.g. 0x80808080)*



We are returning a single byte = 0x38 = 56

- **Don't worry if you don't understand this now!!!**
- In this course we will be looking at memory and pointers in much more detail
- You will learn how to manage memory and how to manipulate it

Dynamic Memory (Malloc)

int * a = malloc(size)

- Must be freed using:

free(a)

- Stored on the heap, known as a dynamic variable
- Can be resized during runtime using:

realloc(a, newsize)

```
1  #include <stdlib.h>
2
3  int main() {
4      int n, i;
5      int *arr;
6
7      // Get the number of elements for the array
8      printf("Enter number of elements: ");
9      scanf("%d", &n);
10
11     // Dynamically allocate memory using malloc
12     arr = (int*)malloc(n * sizeof(int));
13
14     // Check if memory allocation was successful
15     if (arr == NULL) {
16         printf("Memory allocation failed!\n");
17         return 1;
18     }
19
20     // Assign values to the allocated memory and print them
21     for (i = 0; i < n; i++) {
22         arr[i] = i + 1; // Assign values (1, 2, 3, ...)
23         printf("arr[%d] = %d\n", i, arr[i]);
24     }
25
26     // Free the allocated memory
27     free(arr);
28
29     return 0;
30 }
```

Revision: Variables and Formatting

```
#include <stdio.h>
```

```
int main(){  
    int x = 52;  
    //print decimal representation  
    printf("decimal(default):%d\n", x); //output: 52  
    printf("decimal(force 3 digits): %03d\n", x); // output: 052  
    //print hex representation  
    printf("hex: 0x%x\n", x); // ouput:  
    // print binary representation - can only be used with gcc  
    printf("binary: %08b\n", x); //output: 00110010  
}
```

```
z3544032@vx06:~/dpst1092/w1_review$ ./first  
decimal(default):52  
decimal(force 3 digits): 052  
hex: 0x34  
binary: 00110100  
z3544032@vx06:~/dpst1092/w1_review$
```

What you need to know

- Define a variable
- Formatted print
- Specify digits
- %b important for debugging (outputs binary)
- Compiling with both dcc and gcc
- Libraries allow us to use functions i.e. printf, scanf etc.

Memory segmentation

