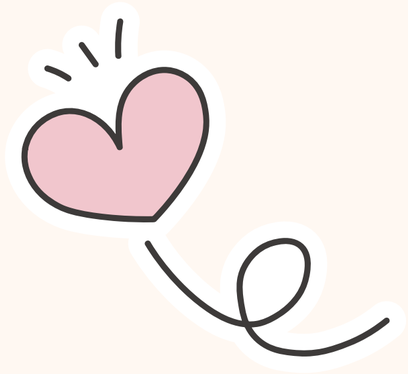
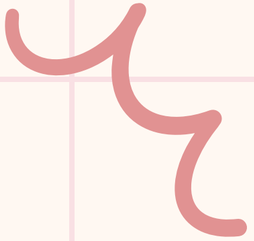
COMP1511 Programming Fundamentals





POINTERS

WEEK 5 LECTURE 2

with Tammy

Announcements

HELP SESSIONS

THEY WILL RUN IN WEEK 6!
STAGE-SPECIFIC FOR ASSN1
CHECK COURSE WEBSITE FOR TIMETABLE :)

WEEK 6 REVISION SESSIONS

MONDAY
13:00-15:00
STRING LAB J17

WEDNESDAY
16:00-18:00
VIA MICROSOFT TEAMS
(SIGN UP VIA LINK ON FORUM*)

WEEK 6 FLEXIBILITY WEEK NEXT WEEK

NO LECTURES
NOR TUT-LABS!

*Will be announced today

LIVE CODE HERE:

https://cgi.cse.unsw.edu.au/~cs1511/24T2/live/week_5/

THIS LECTURE, NEW TOPIC:

Introduction to Pointers!

Memory

- All the data in your code are stored in the computer memory
- Visualise it as a grid with values and each slot in the grid has a unique memory address (sequential hexadecimal values) (like how we have our home addresses, they live somewhere in memory!)
- Each slot have a unique memory address with the relevant data in it e.g. an integer value

Who has heard of the term “Pointers” ?

Who has heard of the term “Pointers” ?

Who has heard of the term “Pointers” ?

What are pointers?

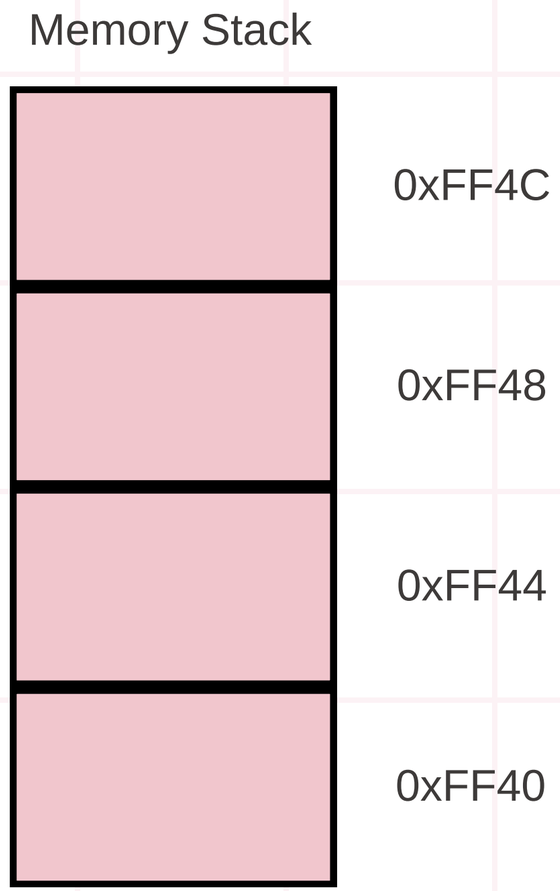
- a variable that stores the *memory address* of another variable
  - aka. “a variable that points to another variable”
- gives us the power to modify things at the source (especially when working with functions)
- to declare a pointer in our code - specify the type the pointer points to with an asterisk:

type_pointing_to ***variable_name;**

E.g.

int ***num_ptr;**

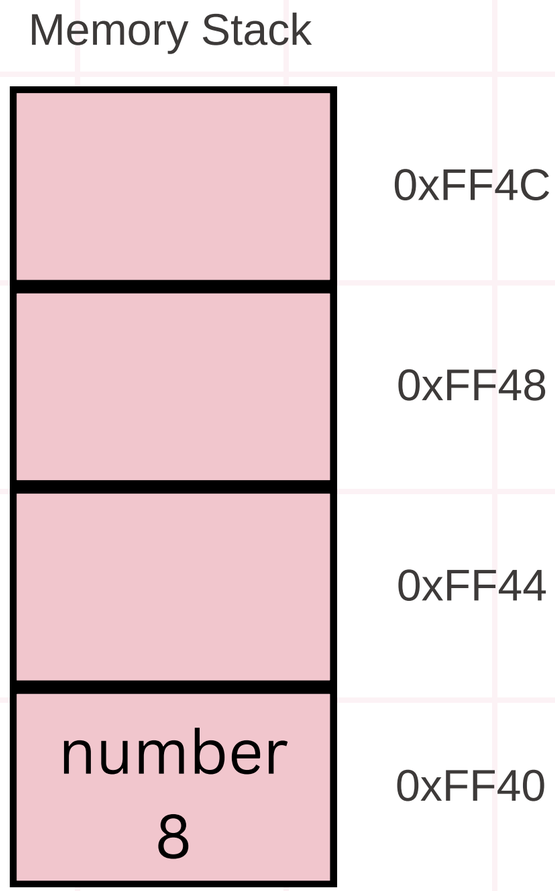
(a pointer variable that can store an address to an integer variable)


Memory Stack
0xFF4C
0xFF48
0xFF44
0xFF40

LET' S...

1. Declare and initialise an integer variable

```
int number = 8;
```



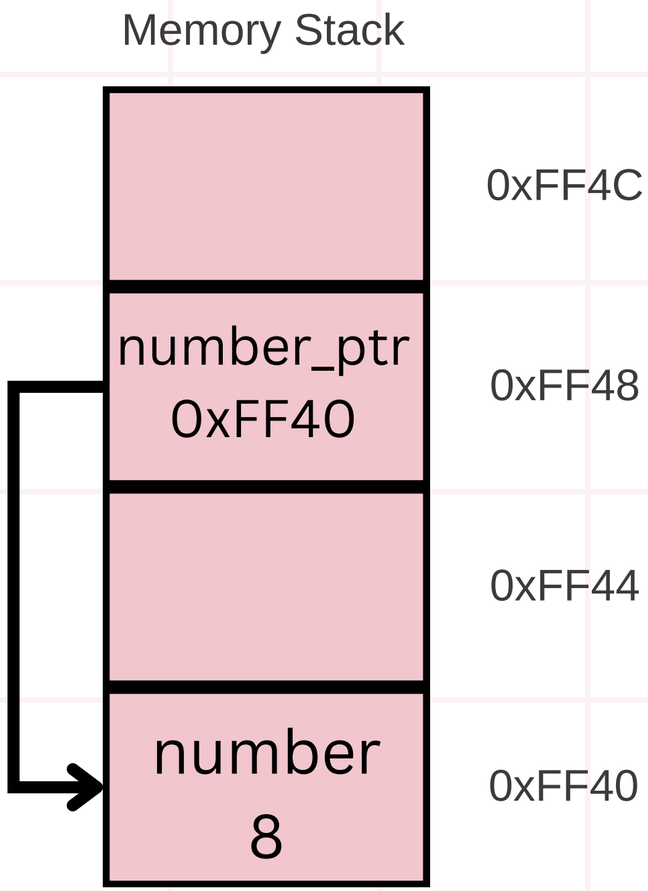
LET' S...

1. Declare and initialise an integer variable

```
int number = 8;
```

2. Declare an integer pointer variable and assign the address of `number` to it

```
int *number_ptr = &number;
```



LET' S...

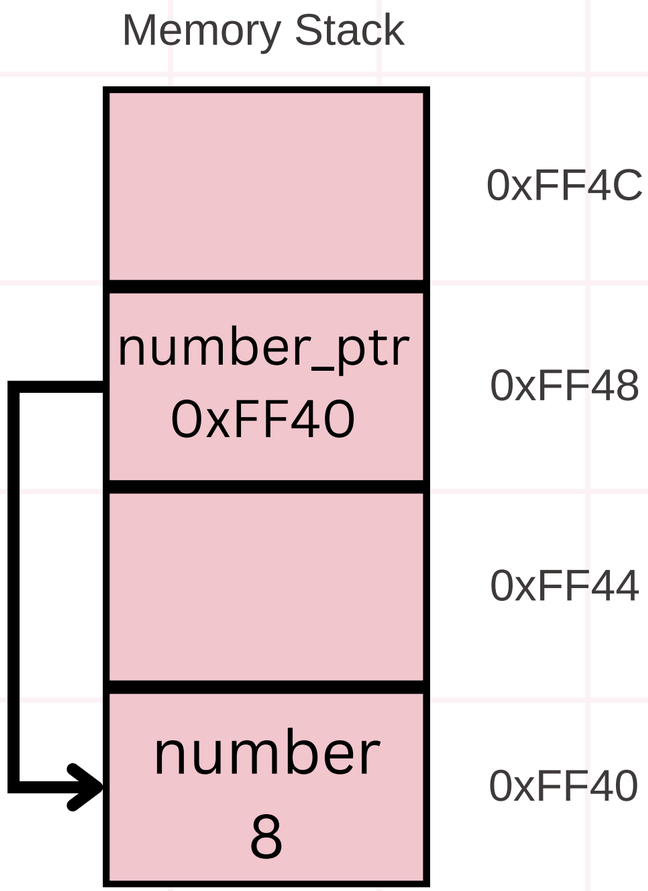
1. Declare and initialise an integer variable

```
int number = 8;
```

2. Declare an integer pointer variable and assign the address of `number` to it

```
int *number_ptr = &number;
```

(the & might look familiar from scanf!)



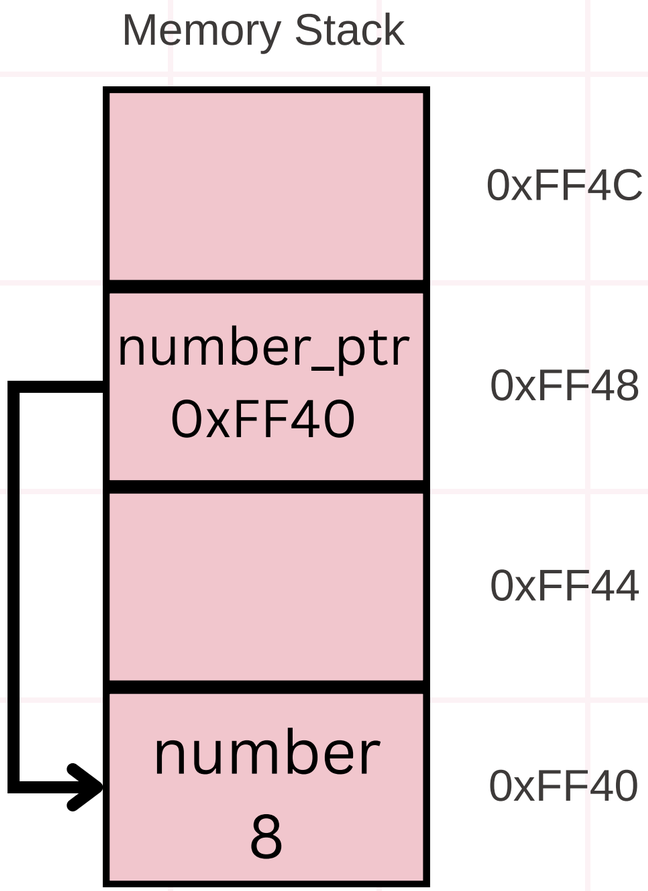
LET' S...

1. Declare and initialise an integer variable

```
int number = 8;
```

2. Declare an integer pointer variable and assign the address of `number` to it

```
int *number_ptr = &number;
```



SO NOW...

number == 8

number_ptr == 0xFF40

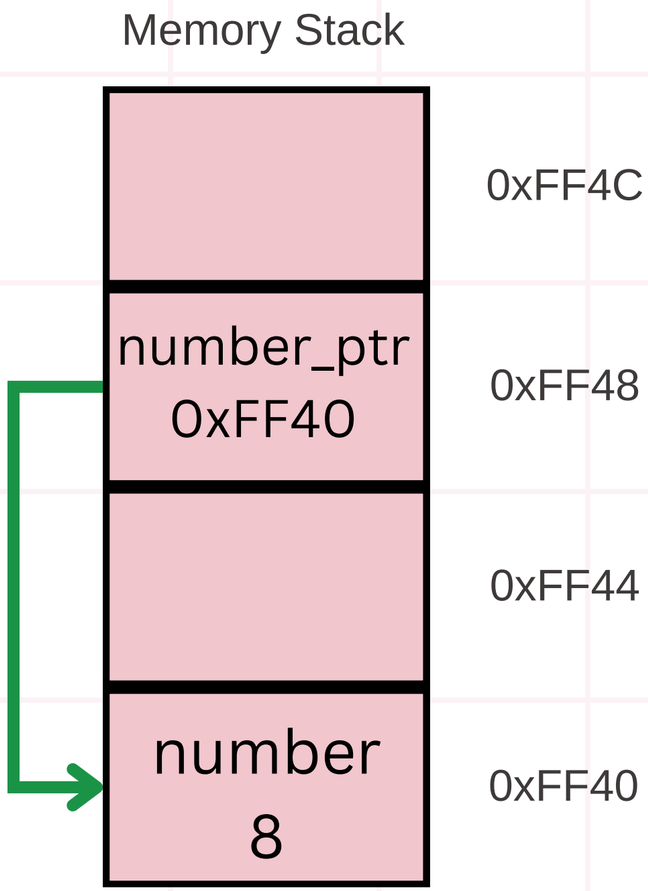
LET' S...

1. Declare and initialise an integer variable

```
int number = 8;
```

2. Declare an integer pointer variable and assign the address of `number` to it

```
int *number_ptr = &number;
```



SO NOW...

number == 8
number_ptr == 0xFF40

AND

*number_ptr == number == 8

LET'S...

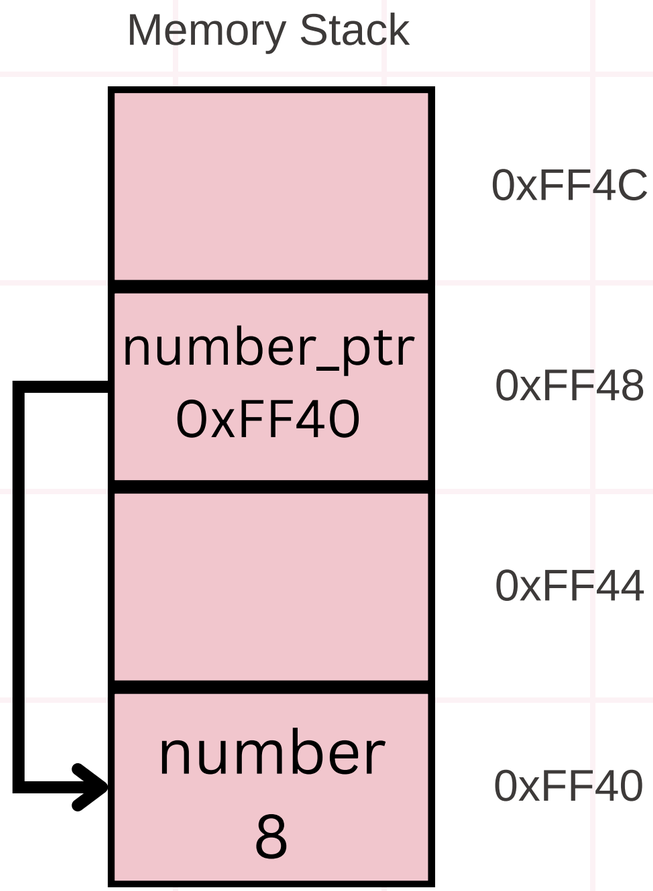
3 KEY COMPONENTS!

1. Declare and initialise an integer variable

```
int number = 8;
```

2. Declare an integer pointer variable and assign the address of `number` to it

```
int *number_ptr = &number;
```



SO NOW...

number == 8

number_ptr == 0xFF40

AND

*number_ptr == number == 8

LET'S...

3 KEY COMPONENTS!

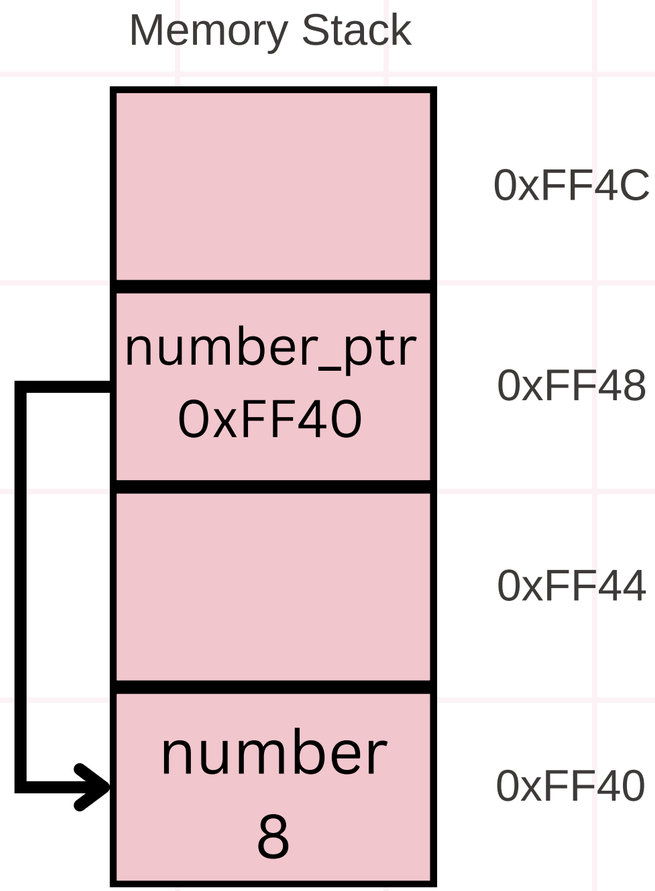
1. Declare and initialise an integer variable

```
int number = 8;
```

2. Declare an integer pointer variable and assign the address of `number` to it

```
int *number_ptr = &number;
```

Declare



SO NOW...

number == 8

number_ptr == 0xFF40

AND

*number_ptr == number == 8

LET'S...

3 KEY COMPONENTS!

1. Declare and initialise an integer variable

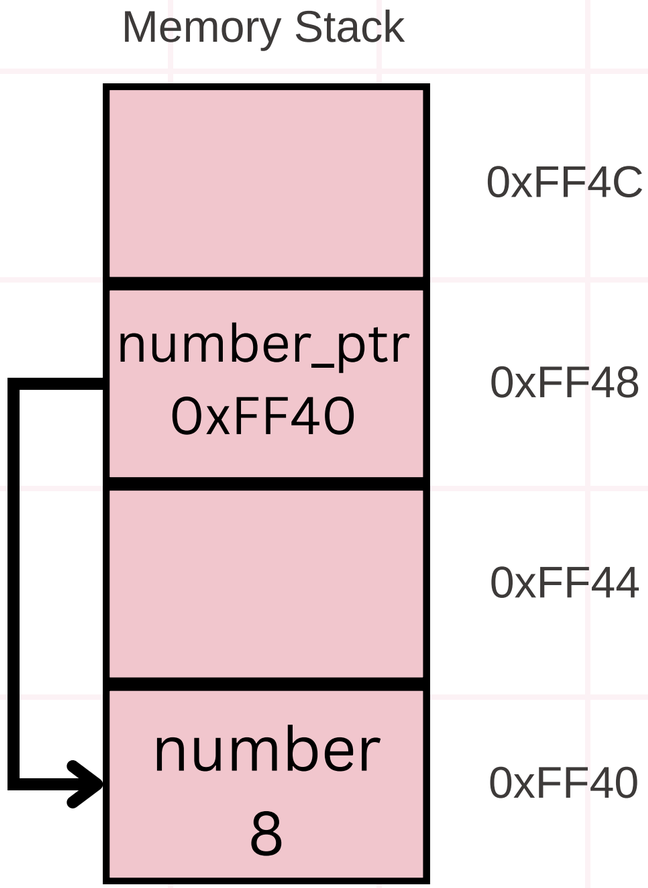
```
int number = 8;
```

2. Declare an integer pointer variable and assign the address of `number` to it

```
int *number_ptr = &number;
```

Declare

Assign the address



SO NOW...

number == 8

number_ptr == 0xFF40

AND

*number_ptr == number == 8

LET'S...

3 KEY COMPONENTS!

1. Declare and initialise an integer variable

```
int number = 8;
```

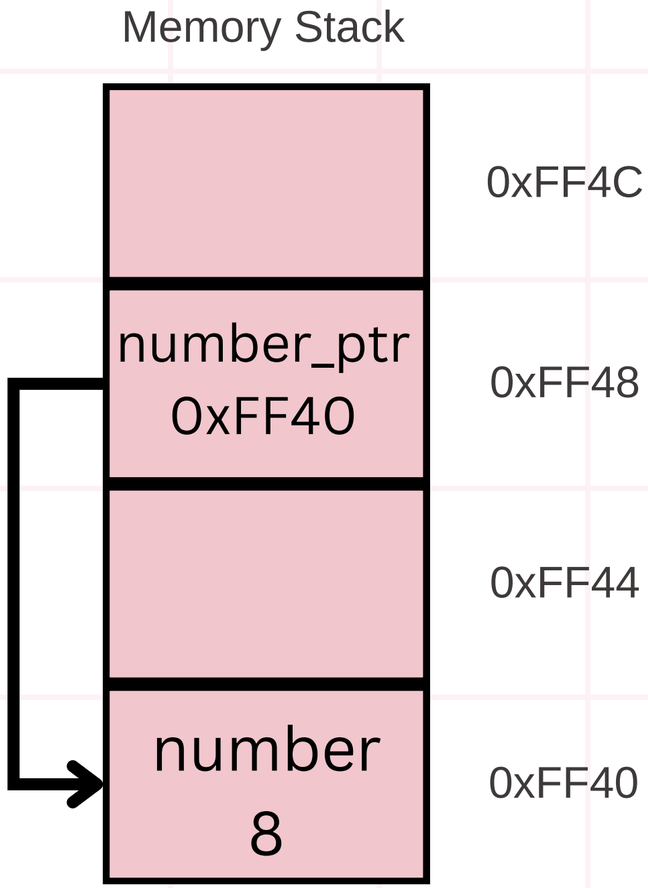
2. Declare an integer pointer variable and assign the address of `number` to it

```
int *number_ptr = &number;
```

Declare

Assign the address

Dereference



SO NOW...

number == 8

number_ptr == 0xFF40

AND

*number_ptr == number == 8

LET'S...

3 KEY COMPONENTS!

1. Declare and initialise an integer variable

```
int number = 8;
```

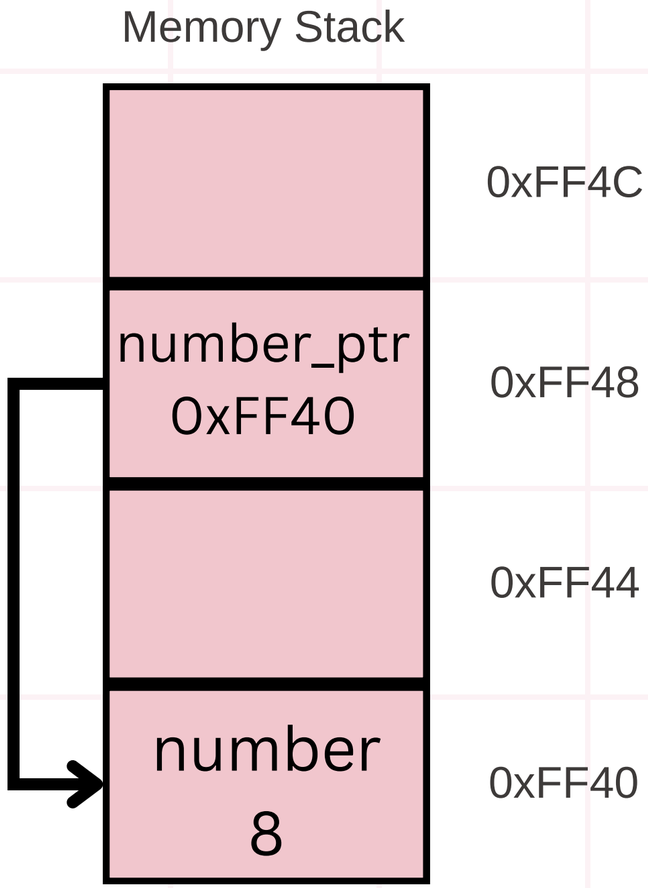
2. Declare an integer pointer variable and assign the address of `number` to it

```
int *number_ptr = &number;
```

Declare

Assign the address

Dereference



SO NOW...

number == 8

number_ptr == 0xFF40

AND

*number_ptr == number == 8

In Short

1. Delcare a pointer variable - using type_pointing_to *

type_pointing_to ***variable_name;**

int ***variable_name;**

2. Assign pointer variable with address of another variable - using &

number_ptr = **&**number;

3. Dereference pointer variable - using *

***number_ptr**

(go to the address that this pointer variable is assigned and find what is at that address)

CODE DEMO!

pointer_intro.c

- fundamentals of the use of pointers
- modifying values when pointers are involved

Mini Quiz: Will the following work in code?

```
int number;
int *number_ptr;

number_ptr = number; // 1
*number_ptr = &number; // 2
number_ptr = &number; // 3
*number_ptr = number; // 4
```

NO - THEY ARE DIFFERENT TYPES

NO - LHS IS AN INT, RHS IS A POINTER (ADDRESS)

YES!

DEPENDS - IS NUMBER_PTR INITIALISED?

BREAK TIME!

What’s the point? 

- We have a problem without it:
  - We cannot return multiple values from a function… cannot return an array…
- This will cause issues in tasks like swapping two variable values in a function (code demo)
- Food for thought: how would you hack your way around this without pointers?

CODE DEMO!

pointer_in_function.c

- demonstrate the purpose of pointers
- using pointers in functions

EXTRA CODE DEMO!

array_addresses.c

- demonstrate array decaying into a pointer
- demonstrate addresses in an array

Don’t forget, we can actually have different types of pointers

```
type_pointing_to *variable_name;
```

```
int value = 8;
int *ptr = &value;

printf(“%d\n”, *ptr);
```

Don’t forget, we can actually have different types of pointers

```
type_pointing_to *variable_name;
```

```
double value = 8.8;
double *ptr = &value;

printf(“%lf\n”, *ptr);
```

Don’t forget, we can actually have different types of pointers

```
type_pointing_to *variable_name;
```

```
char value = ‘t’;
char *ptr = &value;

printf(“%c\n”, *ptr);
```

Don’t forget, we can actually have different types of pointers

type_pointing_to ***variable_name;**

Struct Pointers!

CODE DEMO!

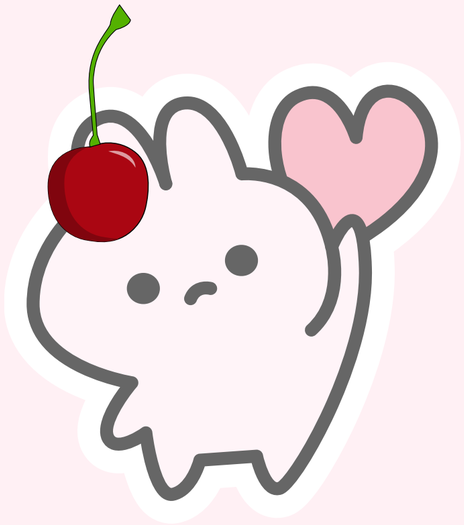
struct_pointer.c

- demonstrate the syntax for struct pointers
- . vs. ->

Why struct pointers?

- Linked List after flex week :)

KAHOOT!



FEEDBACK
(PRETTY PLEASE
WITH A CHERRY
ON TOP)

https://forms.office.com/r/FTgRVnZuRU

SUMMARY OF TODAY

Pointers :)

FLEX WEEK NEXT WEEK!

==

NO CLASSES

(BUT YES HELP SESSIONS + REVISION SESSIONS)

If you have any questions:

COURSE RELATED

COURSE FORUM + HELP SESSIONS!

ADMIN RELATED

CS1511@UNSW.EDU.AU

thank you

And come say hi if you see me around on campus :D