

Pointers

- Pointers
 - Example in memory
 - Pointers
 - All possible cases
 - Finial version with no Errors and no Warnings

```
sizeof(char)=1
sizeof(short)
sizeof(int)= 4 (32 bit) or 8 (32bit)
sizeof(long)
sizeof(float)
sizeof(double)
sizeof(longdouble)
```

Example in memory

```
struct student{
    int id;
    char a;
    int id2;
    char b;
    float percentage
}
```

In memory:

.	1 byte	1 byte	1 byte	1 byte
id	x	x	x	x
a	x			
id2	x	x	x	x
b	x			

.	1 byte	1 byte	1 byte	1 byte
percentage	x	x	x	x

Pointers

Pointers are variables whose values are memory addresses

```
<type> *<pointer>;  
int *pointer;  
int number;  
  
pointer = &number;
```

This means that the pointer is equal to the number address, so pointer points to the number

```
#include <stdio.h>  
#include <stdlib.h>  
  
int main(int argc, char *argv[])  
{  
    float *ptr;  
    float f = 7.5;  
  
    int *ppp;  
    int a = 3;  
  
    ptr = &f;  
    ppp = &a;  
  
    printf("%.2f\n%.2f\n", f, *ptr);  
    printf("%d\n%d\n%d\n", &a, &(*ppp), ppp);  
  
    return 0;  
}
```

Run:

7.5
7.5
32324325525
32324325525
32324325525

All possible cases

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

int main(int argc, char *argv[])
{
    int v = 5;
    int *p;
    p = &v;

    printf("%d\n", v);
    /** printf("%d\n", *v); Error!*/
    printf("%d\n", &v);
    printf("%d\n", p);
    printf("%d\n", *p);
    printf("%d\n", &p);
    printf("%d\n", *(&v));
    /** printf("%d\n", &(*v)); Error!*/
    printf("%d\n", *(&p));
    printf("%d\n", &(*p));

    return 0;
}
```

5
//Error!
957891628
957891628
5
957891616
5
//Error!
957891628
957891628

Simbol	Meaning	Outcome
--------	---------	---------

Simbol	Meaning	Outcome
v	integer value	5
v	meaningless	Error!
&v	Address of v	Warning*! address 957891628
p	It is the address of v that points to v	Warning! address 957891628
*p	It's where p points. So it's the int value v	5
&p		Warning*! address 957891628
*(&v)	v (integer value)	5
&(*v)	meaningless	Error!
(&p)	p	Warning! address 957891628
&(*p)	p	Warning*! address 957891628

*The Warning is because the print is going to print an integer ("%d") but the simbol is the integer address

Finial version with no Errors and no Warnings

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

int main(int argc, char *argv[])
{
    int v = 5;
    int *p;
    p = &v;

    printf("%d\n", v);
    printf("%lu\n", (long unsigned int) &v);
    printf("%lu\n", (long unsigned int) &v);
}
```

```

printf("%lu\n", (long unsigned int) p);
printf("%d\n", *p);
printf("%lu\n", (long unsigned int) &p);
printf("%d\n", *(&v));
printf("%lu\n", (long unsigned int) *(&p));
printf("%lu\n", (long unsigned int) &(*p));

return 0;
}

```

```

5
140735223946540
140735223946540
140735223946540
5
140735223946528
5
140735223946540
140735223946540

```

Simbol	Meaning	Outcome
v	integer value	5
&v	Address of v	Address 140735223946540
p	It is the address of v that points to v	Address 140735223946540
*p	It's where p points. So it's the int value v	5
&p		Address 140735223946540
*(&v)	v (integer value)	5
*(&p)	p	Address 140735223946540
&(*p)	p	Address 140735223946540