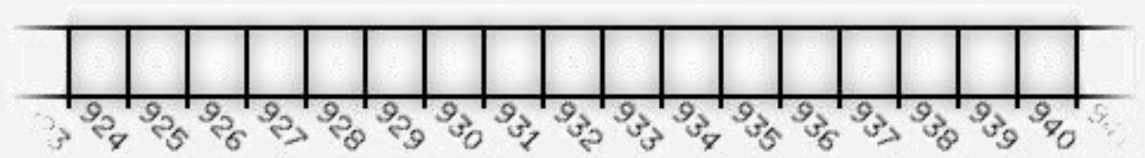
Lesson 11
Pointers

Trịnh Thành Trung

trungtt@soict.hust.edu.vn

Memory address

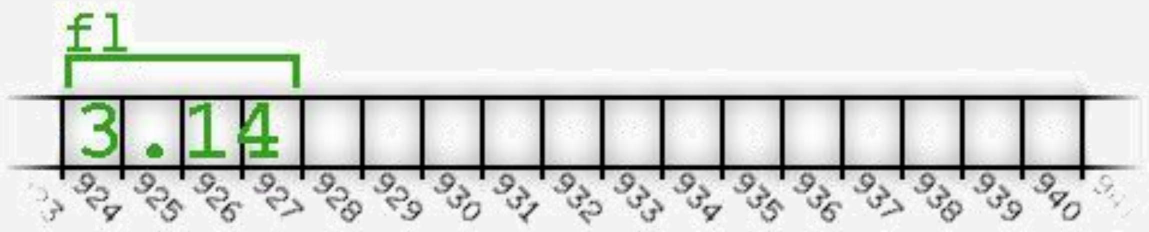
- Computer's memory is made up of bytes. Each byte has a number, an address, associated with it.
- In the picture below, addresses 924 through 940 are shown.



Memory address

- The unary operator **&** gives the address of a variable

```
#include <stdio.h>
int main()
{
    float fl=3.14;
    printf("fl's address=%u\n", (unsigned int) &fl);
    return 0;
}
```



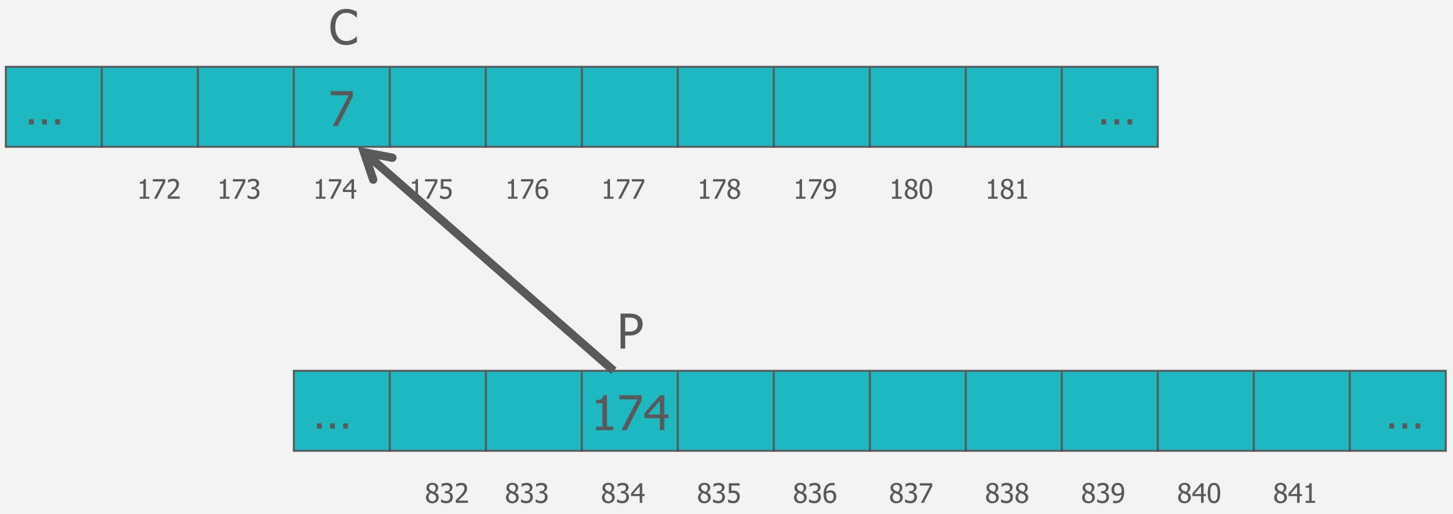
Declaring a pointer variable

```
type *variable_name;
```

- A pointer is declared by adding a * before the variable name.
- Pointer is a variable that contains an address in memory.
- The address should be the address of a variable or an array that we defined.

Pointers

- Here ptr is said to *point* to the address of variable c



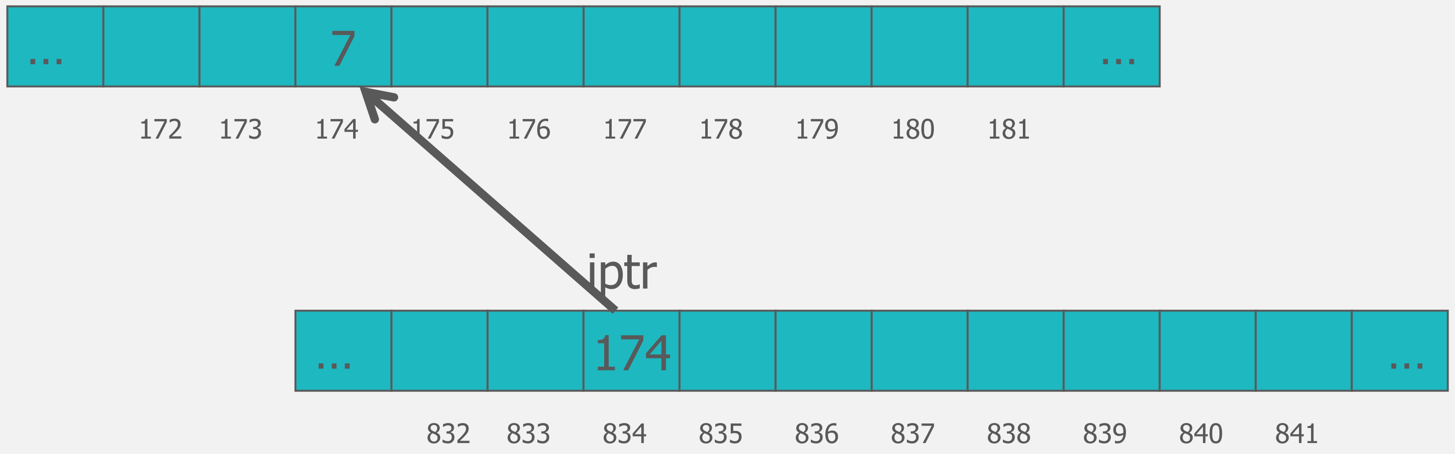
Referencing

- The unary operator **&** gives the address of a variable
- The statement: **ptr = &c;**

  assigns the address of **c** to the pointer variable **ptr**, and now **ptr** points to c
- To print a pointer, use **%p** format.

Referencing

```
int n;
int *iptr; /* Declare P as a pointer to int */
n = 7;
iptr = &n;
```



Dereferencing

- The unary operator ***** is the dereferencing operator
- Applied on pointers
- Access the object the pointer points to
- The statement: ***iptr = 5;**

  puts in **n** (the variable pointed to by **iptr**) the value 5

Exercise 12.1

- Write a C program to input three integers. Set up a single pointer to point to each of these integers in turn. Display the value dereferencing the pointer.

Exercise 12.2

- Write a program that print out the address (in hexadecimal format) of first 5 elements of the array predefined as below:

```
int a[7]= {13, -355, 235, 47, 67, 943, 1222} ;
```

Exercise 12.3

- Write a program asking the value from user for 3 float variable a, b, c. Then add 100 to the content of them by using just a pointer.

Pass arguments by value

- The functions we saw until now received their arguments “by value”
- They could manipulate the passed values
- They couldn’t change values in the calling function

Wrong Swap

A swap that gets integers as variables does not change the value in the original variables.

```
void swap(int x,  int y)
{
    int tmp = x;
    x = y;
    y = tmp;
}
```

How can we fix it?

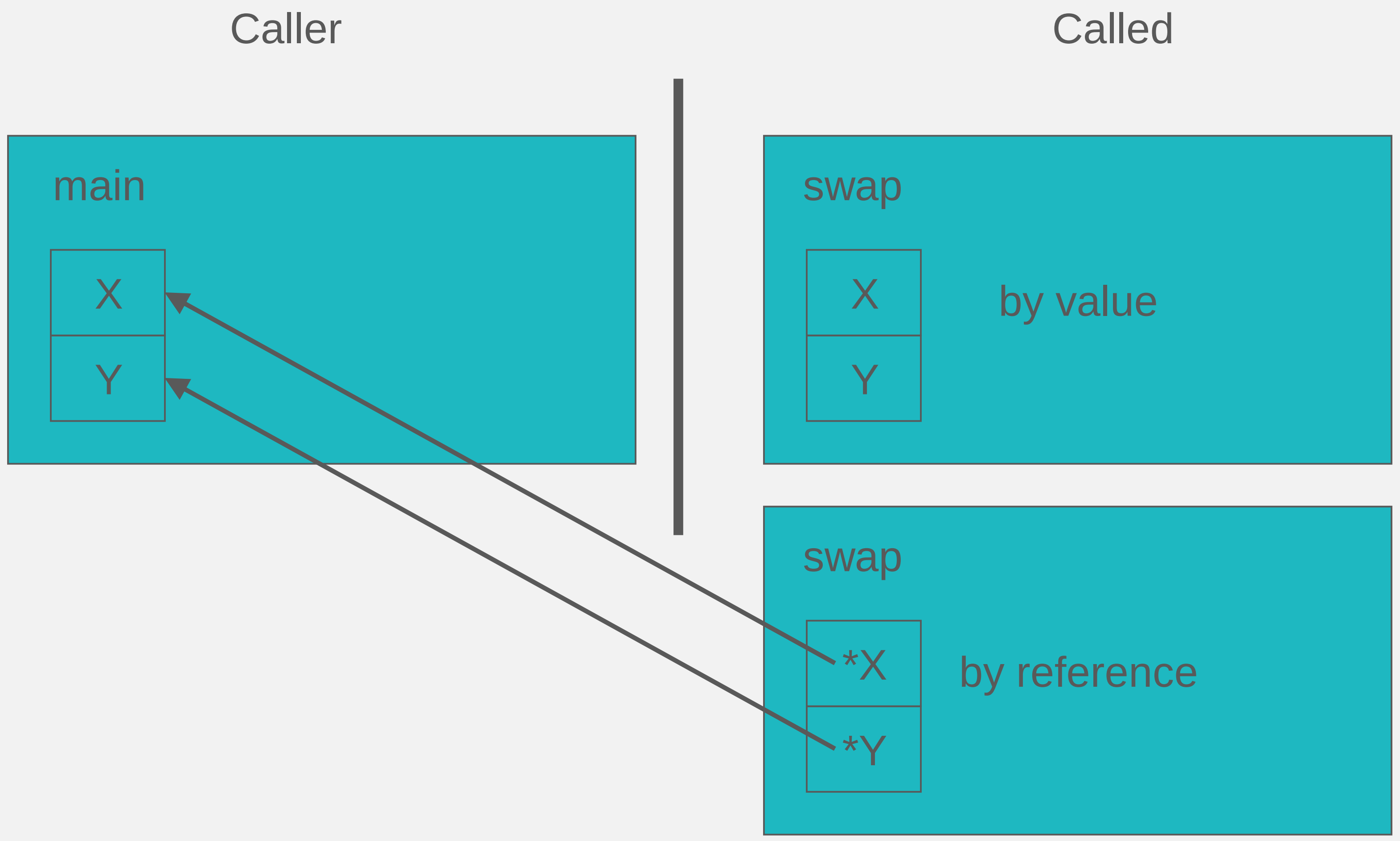
- We can define swap so it gets <u>pointers to</u> integers instead of integers

```
void swap(int *x,  int *y)
{
    int temp = *x;
    *x = *y;
    *y = temp;

}
```

- We then call swap by `swap(&x, &y);`
- This is pass by reference

Swap function representation



Exercise 12.4

- Write a function that takes three variable (a, b, c) in as separate parameters and rotates the values stored so that value a goes to be, b, to c and c to a. Test this function in a program

Exercise 12.5

- Introduce **int** variables **x**, **y**, **z** and **int*** pointer variables **p**, **q**, **r**. Set **x**, **y**, **z** to three distinct values. Set **p**, **q**, **r** to the addresses of **x**, **y**, **z** respectively.

1) Print with labels the values of **x**, **y**, **z**, **p**, **q**, **r**, ***p**, ***q**, ***r**.

2) Swapping values of x, y, z. Print with labels the values of **x**, **y**, **z**, **p**, **q**, **r**, ***p**, ***q**, ***r**.

3) Swapping values of p, q, r. Print with labels the values of **x**, **y**, **z**, **p**, **q**, **r**, ***p**, ***q**, ***r**.

Exercise 12.6

- To increase salary for an employee, write a function *incomeplus* that is based on the current salary and the number of years passed from the beginning years (must > 3) of current salary.
- Test it in a program.