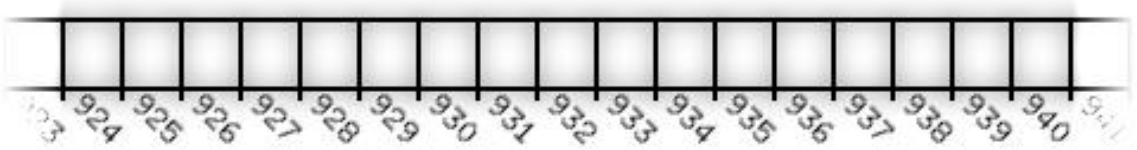
C Programming Introduction

week 11: Pointers

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;
}
```

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.

Solution

```
#include <stdio.h>

int main(){
   int x, y, z;
   int* ptr;
   printf("Enter three integers: ");
   scanf("%d %d %d", &x, &y, &z);
   printf("\nThe three integers are:\n");
   ptr = &x;
   printf("x = %d\n", *ptr);
   ptr = &y;
   printf("y = %d\n", *ptr);
   ptr = &z;
   printf("z = %d\n", *ptr);
   return 0;
}
```

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} ;
```

Solution

```
#include <stdio.h>
int main(){
  int a[7]= {13, -355, 235, 47, 67, 943,
  1222};
  int i;
  printf("address of first five elements in
  memory.\n";
  for (i=0; i<5;i++)printf("\ta[%d]",i);
  printf("\n");
  for (i=0; i<5;i++)printf("\t%p",&a[i]);
  return 0;
}
```

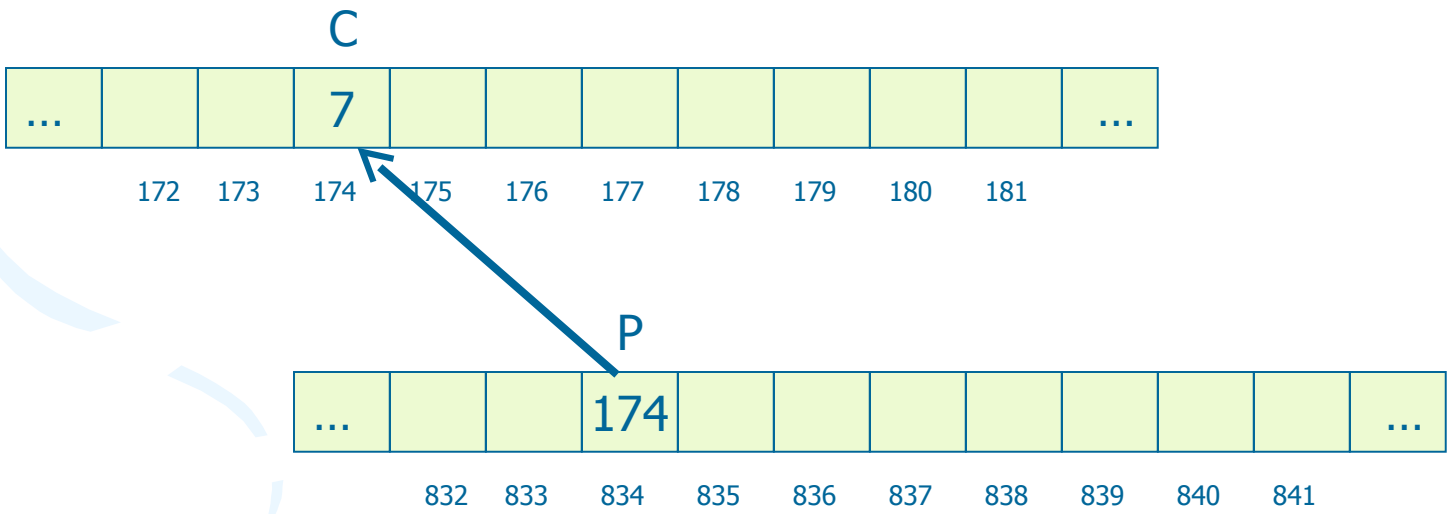
Declaring a pointer variable

```
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

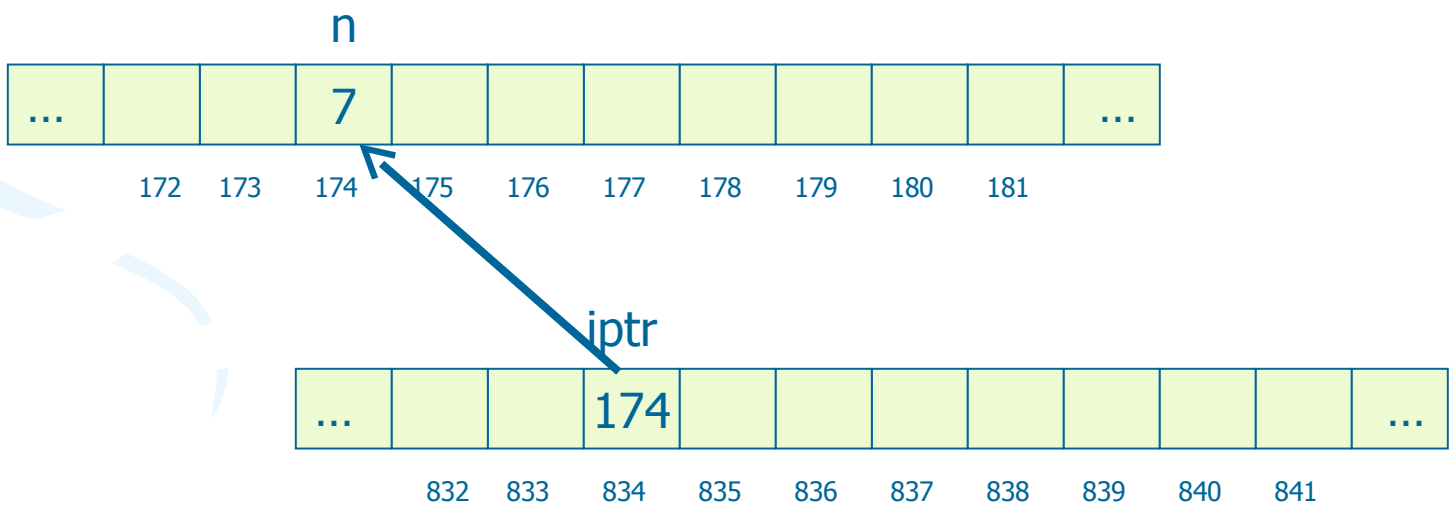


Referencing

- 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 12.3

- 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.

Solution

```
#include <stdio.h>
void main(void)
{
   int x = 25, y = 50, z = 75;
   int *ptr;
   printf("Here are the values of x, y, and z:\n");
   printf("%d %d %d\n", x, y, z);
   ptr = &x;  // Store the address of x in ptr
   *ptr += 100; // Add 100 to the value in x
   ptr = &y;  // Store the address of y in ptr
   *ptr += 100; // Add 100 to the value in yx
   ptr = &z;  // Store the address of z in ptr
   *ptr += 100; // Add 100 to the value in z
    printf("Once again, here are the values of x, y,
   and z:\n");
    printf("%d %d %d\n", x, y, z);
}
```

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.

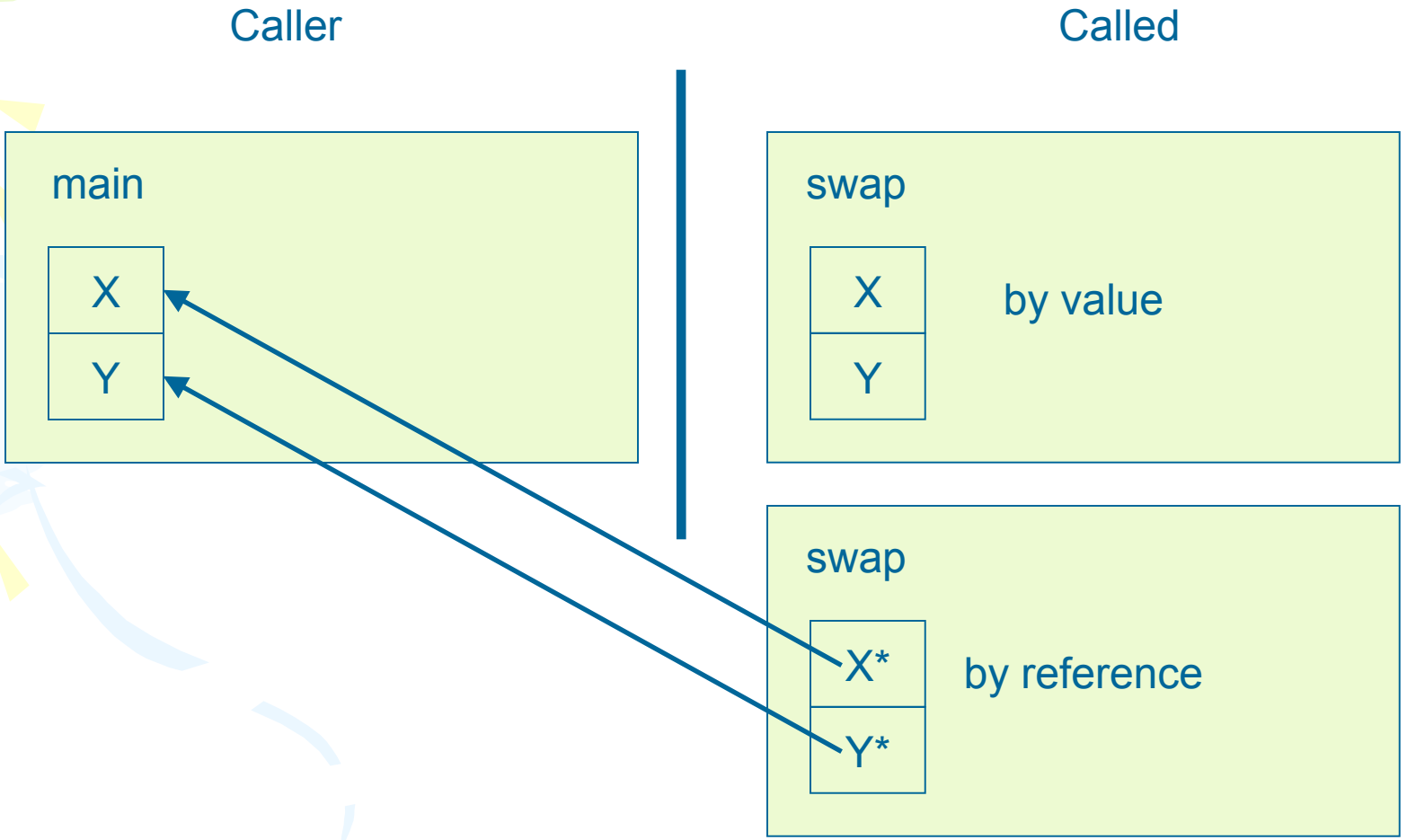
```
void swap(int x,  int y)
{
        int tmp = x;
    x = y;
    y = tmp;
}
```

How can we fix it?

- We can define swap so it gets pointers to 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


Caller
Called
main
X
Y
swap
X
Y
by value
swap
X*
Y*
by reference

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

Solution

```
#include <stdio.h>
void swap3(int *p, int *q, int *r){
   int tmp;
   tmp= *p; *p=*q; *q=*r; *r=tmp;
}
void main(void)
{
   int a, b, c;
   printf("Enter a, b, c:");
   scanf("%d%d%d", &a, &b, &c);
   printf("Value before swap. a=%d, b=%d, c=%d\n", a,
   b, c);
   swap3(&a,&b,&c);
   printf("Value after swap. a=%d, b=%d, c=%d\n", a,
   b, c);

}
```

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**.

Exercises 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.

Solution

```
#include <stdio.h>
void incomeplus(long *current, int year){
   if (year >3) *current = *current + 300000;
}
void main(void)
{
   long cursal; int year;
   do {
   printf("Enter your current salary:);
   scanf("%ld",&cursal);
   printf("Number of years passed:");
   scanf("%d", &year);
   incomeplus(&cursal,year);
   printf("Your salary now: %ld", cursal);
   }while(year!=-1);

}
```