CSE216
Foundations of Computer Science

Instructor: Zhoulai Fu

State University of New York, Korea

C crash course (cont.)

Errata:
Indeed, C99 has a bool type

- It is called _Bool
- Aliased to bool in stdbool.h
- But no bool before C99. E.g. No bool in C89 (ANSI C).

In the following

- **C Pointers**
- **Lab questions**

FANG Interview Question 1: const char* vs. char* const

- What does **const char* s = “hello";** do ?

- **const char*** is a pointer to a "constant character”. This means you're not allowed to change the character that the pointer is referring to

FANG Interview Question 1: const char* vs. char* const

- What does **char* const str = “hello";** do ?

- **char* const** is a “constant pointer” to a character. This means you're not allowed to change the pointer

FANG Interview Question 2: What is void*?

- generic pointer type
- raw address in memory
- can store the address of any object
- can be type-casted to any type of pointer
- Cannot be directly dereferenced. To access the object, you must first cast void * to a correct pointer type.

```c
#include <stdio.h>

void printNumber(void *ptr, char type) {
    if (type == 'i') { // If the type is integer
        int *int_ptr = (int *)ptr;
        printf("The number is %d\n", *int_ptr);
    } else if (type == 'f') { // If the type is float
        float *float_ptr = (float *)ptr;
        printf("The number is %f\n", *float_ptr);
    }
}

int main() {
    int i = 5;
    float f = 9.5;

    // Passing integer pointer
    printNumber(&i, 'i');

    // Passing float pointer
    printNumber(&f, 'f');

    return 0;
}
```

Address of (&)

- To get the address of a variable, we use the ampersand (&) operator,

```c
#include <stdio.h>

int main() {
    int var = 5;
    printf("Address of var: %p\n", (void*)&var);
    return 0;
}
```

Pointers (*)

- A pointer is a variable whose value is an address
- To define a pointer, use **type *var-name;**

```
int    *ip;    /* pointer to an integer */
double *dp;    /* pointer to a double */
float  *fp;    /* pointer to a float */
char   *ch     /* pointer to a character */
```

Question

- If the 1st printf gives 0x7fff5c7ea38c, what will be the results of the 2nd and 3rd printf?

```c
#include <stdio.h>
int main () {
   int var = 20;
   int *ip = &var;

   printf("Address of var variable: %p\n", (void*) &var  );

  printf("Address stored in ip variable: %p\n", (void *) ip );

  printf("Value of *ip variable: %d\n", *ip );
  return 0;
}
```

Question

- Any difference between int *ptr and int* ptr?
- The default usage is int *ptr; although many prefer the other
- What is the type of “b” in **int* a, b**?

NULL Pointer

- The NULL pointer always points to no objects
- int *ptr = NULL;
- cannot be dereferenced

Issue from an online tutorial

The **NULL** pointer is a constant with a value of zero defined in several standard libraries. Consider the following program:

```
#include <stdio.h>

int main ()
{
   int  *ptr = NULL;

   printf("The value of ptr is : %x\n", &ptr  );

   return 0;
}
```

When the above code is compiled and executed, it produces the following result:

```
The value of ptr is 0
```

Not always right (compiler dependent)

Question: What will happen if we run this

```c
#include <stdio.h>
int main () {

  int *ptr = NULL;

  printf("The value of ptr is : %p\n", (void *) &ptr  );

  printf("The value of ptr is : %p\n", (void *) ptr  );

  printf("The value of ptr is : %x\n", *ptr  );

return 0;
}
```

Try: jdoodle.com/ia/IN3

Check if a pointer is Null

```
if(ptr)     /* succeeds if p is not null */
if(!ptr)    /* succeeds if p is null */
```

FANG Interview Question 3: Pointer arithmetics

- Assume ptr is an 32-bit integer pointer which points to the address 0x1009.

- ptr++ = ?

- Assume ptr is a char pointer which points to the address 1009.

- ptr++ = ?

FANG Interview Question 4: Which one will crash, and why?

```c
#include <stdio.h>

const int MAX = 3;
int main () {

    int var[] = {10, 100, 200};

    double b[5] = {1000.0, 2.0, 3.4, 17.0, 50.0};
    double *ptr=b;

    for (int i = 0; i < 5; i++) {
        printf("Value of balance[%d] = %f\n", i, *ptr );
        /* move to the next location */
        ptr++;
    }
    return 0;
}
```

```c
#include <stdio.h>

const int MAX = 3;
int main () {

    int var[] = {10, 100, 200};

    double b[5] = {1000.0, 2.0, 3.4, 17.0, 50.0};

    for (int i = 0; i < 5; i++) {
        printf("Value of b[%d] = %f\n", i, *b );
        /* move to the next location */
        b++;
    }
    return 0;
}
```

Arrays

- Arrays are constant pointers
- int x[5];
- x++ would not work
- Use int * ptr = x; ptr++;

C Exercises

1

- What will be the output of the following C code?

```
int main() {
    int var = 20;
    int *ptr;
    ptr = &var;
    printf("%d", *ptr);
    return 0;
}
```

2

- What will be the output of the following C code?

```
int x = 10;

int *p = &x;

*p = 20;

printf("%d", x);
```

3

- What will be the output of the following C code?

```
int main() {

    int arr[] = {10, 20, 30, 40, 50, 60};

    int *ptr = arr;

    printf("%d ", *(ptr++));

    printf("%d", *ptr);

    return 0;

}
```