EXAM 2 REVIEW & STRUCTS

SINCE SOME QUESTIONS ARE STRAIGHT-FORWARD, SOME MAY OR MAY NOT BE EXACTLY ON THE EXAM.

FALL 2023

WRITE MY NAME IN THE TA NAME FIELD

- **If it does not have Kyle or Kyle Parker, you will lose 5 points!**

BASIC QUESTIONS

- How many types can an array hold?
  - One
- Is an array a data structure?
  - Yes, **a data structure is a way of organizing memory.**
- What is an array?
  - A contiguous section of memory corresponding to a specific type.
  - Must explicitly give a size.
- What is a struct?
  - A data structure that has a name for a group of related fields (i.e., person)
- What is a C string?
  - An array with a terminating null character (‘\0’).
- Is an array a pointer?
  - No, the name holds the starting memory address (address of array[0]).

POINTERS

- Declare
  - `int * myIntPtr;`
- Obtain
  - `&myTargetVar;`
- Pass (in argument)
  - `foo(myIntPtr); // foo(int*)`
  - `bar(&myInt); // bar(int*)`
- Dereference
  - `*myVarPtr;`
- Increment/Decrement
  - `myVarPtr++;`
  - `myVarPtr--;`

POINTER-ARITHMETIC NOTATION (SPECIFIC TO GENERAL)

- myArr[100] is the same as *(myArr + 100)
- *(ARR_NAME + OFFSET)
- Dereference (MEMORY ADDRESS + OFFSET)
- Dereference (MEMORY ADDRESS)
- OFFSET can be a hard-coded value or a variable

SENTINEL VS COUNTING LOOP

- **Sentinel**

```
while(More To Read) {
      // Read and process
input
}
```

- **Counting**

```
for (int a = 0; a < 100;
++a) {
      // Process value of a
}
```

STRINGS

```
char navDev[20] = "Garmin";
char navApp[9] = "Gaia GPS";
```

- How do I copy two (2) characters from navApp to navDev?

```
strncat(navDev, navApp, 2);
```

- [Advanced] How do I copy " GPS” from navApp to navDev?

```
strncat(navDev, navApp + 4, 11);
```

```
Alt: strcat(navDev, navApp + 4);
```

- How do I copy navDev to navApp overriding the (i.e., navDev should equal navApp)

```
strncpy(navApp, navDev, 9); or strcpy(navApp, navDev);
navApp[strlen(navDev)] = '\0'; // Unsafe, but we know
```

STRINGS (CONT.)

- What function do we use for copying data?

```
strncpy(DEST, SRC, NUM TO COPY);
strcpy(DEST, SRC);
```

- What function do we use for concatenation?

```
strncat(DEST, SRC, NUM TO CONCAT);
strcat(DEST, SRC);
```

- What function do we use for comparing string values?

```
strncmp(FIRST, SECOND, NUM TO CHECK);
strcmp(FIRST, SECOND);
```

Assume all strings are initially set to all null characters

STRINGS (CONT.)

```
char navDev[20] = "Garmin";
char navApp[9]  = "Gaia GPS";
char myChar     = '|';
```

- What will `strcmp(navDev, navApp)` return? (positive, negative, or zero)

  `POSITIVE`

- What is the value of navApp after `strcat(navApp, navDev)`?

  `(Program crashes) [Buffer overflow]`

- What is the value of navDev after `strncat(navDev, &myChar, 10)`?

  `Garmin|`

- What is the value of navDev[20] after the previous call?

  `(Program crashes) [Out of bounds read]`

- What is the value of navDev[18] after the previous call?

  `'\0'`

DON'T MAKE THE SAME MISTAKES I DID; MANY MADE THIS MISTAKE ON QUIZ 7

```
char myString[100] = "Something Here";
myString[2] = 'm'; // Single char, not array
myString[12] = 'r'; // Single char, not array
myString[14] = '\0'; // Single char, not array
```

FOR LOOP TO WHILE LOOP

- Problem

```
int sum = 0;
for (int i = 0; i < 100; ++i)
{
    sum += pow(i, 2);
}
sum = sqrt(sum);
```

- Solution

```
int sum = 0;
int i = 1;
while (i < 100) {
    sum += pow(i, 2);
}
sum = sqrt(sum);
```

I have some exercises in GitHub on converting for loops to while.

WHILE LOOP TO FOR LOOP

- Problem

```
int sum = 0, current, count, avg;
while (
fscanf(myFile, "%d", &current)
!= EOF) {
    sum += current;
    ++count;
}
avg = sum / count;
```

- Solution

```
int sum = 0, current, count, avg;
for (;
    fscanf(myFile, "%d", &current)
!= EOF;
    ++count) {
    sum += current; // This can
technically go in the incrementer
}
avg = sum / count;
```

I have some exercises in GitHub on converting while loops to for.

QUESTIONS FOR EXAM 2?

STRUCTS

LAB 10

FALL 2023

HOW TO DECLARE STRUCTS

```
typedef struct _person {
    char * name;
    int age;
    char gender;
} Person;
// Usage:
//    Person p = {.name = malloc(), .age = 10, .gender = ‘d’};
```

```
//    Person * p;
struct _person {
    char * name;
    int age;
    char gender;
};
// Usage:
//    struct _person p;
//    struct _person * p;
```

STRUCTS (NON-POINTER DEFINITION)

```
typedef struct _person {
    char * name;
    int age;
    char hair_color;
} Person;
```

- Suppose we have a person instance p.

```
Person p;
p.age = 100;
p.hair_color = ‘c’;
strcpy(p.name, “My Name”);
```

STRUCTS (POINTER DEFINITION)

```
typedef struct _person {
    char * name;
    int age;
    char hair_color;
} Person;
```

- Suppose we use malloc to create p.

```
Person * p =
malloc(sizeof(Person));
// verify p is given memory.
p->age = 100;
p->hair_color = ‘c’;
strcpy(p->name, “My Name”);
```

STRUCTS (OPERATORS)

- `. (Access)`
  - Access a member of a struct
  - `Struct.myInt = 100;`
- `-> (Access)`
  - Access a member using a pointer to a struct
  - `myPtrToMyStruct->myInt = 100;`
- LHS = RHS (Copy)
  - Copy struct on RHS to LHS
  - This happens internally, this is simply done by copying field-by-field from RHS to LHS.
  - `MyStruct newStruct = copyThisStructOver;`

WHAT TYPES CAN STRUCTS CONTAIN?

- Any type. In an indirect manner, we can have “functions” as a member.
- Stick to types we have used so far.