# 601.220 Intermediate Programming

Spring 2023, Day 5 (Feb 1st)

## Today's agenda

- Exercise 4 review
- Arrays and strings
- Exercise 5

## Reminders

- HW0 due Friday (Feb 3rd)

## Exercise 4 review

Renaming your personal repository (to make it easier to access for future exercises, homework, etc.):

```
mv 2023-spring-student-JHEDID my220repo
```

Now you can refer to your personal repository as `~/my220repo`.

## Exercise 4 review

Starting an exercise (this general procedure will work for future exercises):

```
cd ~/cs220-sp23-public
git pull
cd ~/my220repo
mkdir -p exercises/ex04
cd exercises/ex04
cp ~/cs220-sp23-public/exercises/ex04/* .
```

## Exercise 4 review

Writing a loop to read grade followed by number of credits:

```c
char grade;
float num_credits;

while (scanf(" %c %f", &grade, &num_credits) == 2) {
  // tally grade and number of credits
  // ...
}
```

Note that

- The space before `%c` makes `scanf` skip whitespace characters before reading the grade character
  - Otherwise it could read a whitespace character instead of the letter grade
- If the user enters Control-D, that ends the input

## Exercise 4 review

Using a switch statement to match the grade:

```
switch (grade) {
case 'A': case 'a':
  quality_points += (num_credits * 4.0);
  break;

case 'B': case 'b':
  quality_points += (num_credits * 3.0);
  break;

// ...etc...

default:
  // invalid grade
}
```

## Day 5 review questions

1. When we declare an array in C, what are the initial values?
2. What is the ASCII (Unicode) table?
3. What is a null terminator? What is its ASCII value?
4. Consider c-string `"ab\0cd\0"` - what is the reported string length?
5. How do we check if two C-strings are the same? In addition, are these two strings the same: `"ab\0cd\0"` and `"ab\0"`?

## 1. When we declare an array in C, what are the initial values?

Elements of an array are uninitialized by default. For example:

```c
int a[3];
printf("%d\n", a[0]); // undefined behavior
```

## 2. What is the ASCII (Unicode) table?

Text characters are represented as integer “character codes”.

ASCII codes range from 0 to 127. Examples:

- “`!`” has the code 33
- “`0`” has the code 48
- “`A`” has the code 65
- “`a`” has the code 97

In C, a *character literal* (in single quotes) yields the ASCII code for that chracter. E.g., `'A'` is the integer value 65.

Unicode: encoding scheme for (essentially) all characters in all human languages, plus symbols, emojis, etc.

## 3. What is a null terminator? What is its ASCII value?

In C, a character string is

- stored in an array of `char` elements, and
- is terminated by a "NUL" character, which is the character whose integer character code is 0

The NUL character can be written as `'\0'` or just `0`.

E.g.:

```c
char s[4] = "foo";
assert(s[0] == 'f');
assert(s[1] == 'o');
assert(s[2] == 'o');
assert(s[3] == 0);
```

## 4. Consider c-string `"ab\0cd\0"` - what is the reported string length?

Note that `\0` in a string literal means a literal NUL character.

The `strlen` function determines the length of a string, which is the number of characters preceding the NUL terminator marking the end of the string.

So:

```
assert(strlen("ab\0cd\0") == 2);
```

## 5. How do we check if two C-strings are the same? In addition, are these two strings the same: `"ab\0cd\0"` and `"ab\0"`?

The `strcmp` function returns 0 if the two strings passed as arguments consist of the same sequence of characters.

So:

```
assert(strcmp("ab", "ab\0cd") == 0);
assert(strcmp("ab", "abc") != 0);
```

## Exercise 5

- Enhanced `.bashrc` and `.bash_profile` startup scripts
- Working with character arrays and strings

Let us know if you have a question!

## Notes

## Notes

## Notes

## Notes

## Notes