CS 24000 - Programming In C

Week Three: Arrays, Strings, Pointers

Zhiyuan Li
Department of Computer Science
Purdue University, USA

PURDUE
UNIVERSITY

Lab requirement on the main() function

- Announce: In order to get the autograder to work properly, all labs and projects require the main function to
  - Declare the return type to be int
  - To explicitly return 0 when the program finishes

```
Int main() {

 if (….) return 0;
….
 return 0;
}
```

PURDUE
UNIVERSITY

Arrays in C

- When defining an array, its size must be given
- *But when just declaring an array w/o defining it, the size is omitted.*
- Let's run this example:

```
#include <stdio.h>
main() {
        int i, ndigit[10];
        for (i = 0; i < 10; ++i) {
                ndigit[i] = i;
                printf("i = \t %d\n", i);
        }
        for (i = 0; i < 10; ++i)
                printf("ndigit[%d] = \t %d\n", i, ndigit[i]);
}
```

- Next, we separate the program into two files (*main.c*, *f.c*) and try again

```
int ndigit[10];
extern void f();
Main() {
        /* int i, ndigit[10]; */
        int i;
        for (i = 0; i < 10; ++i) {
                ndigit[i] = i;
        }
        f();
}
```

```
#include <stdio.h>
extern int ndigit[];

void f() {
        int i;

        for (i = 0; i < 10; ++i)
                printf("ndigit[%d] = \t %d\n", i, ndigit[i]);
}
```

PURDUE UNIVERSITY

- This time, we pass the array as an argument in the called function

```
extern void g(int []);
Main() {
        int i;
        for (i = 0; i < 10; ++i) {
                ndigit[i] = i;
        }
        g(ndigit);
}
```

```
#include <stdio.h>

void g(int mydigit[]) {
        int i;

        for (i = 0; i < 10; ++i)
                printf("mydigit[%d] = \t %d\n", i, mydigit[i]);
}
```

We will discuss more about function parameters later in the lecture

PURDUE UNIVERSITY

Initializing an Integer Array In Its Definition

```
#include <stdio.h>
int a[ ]={0,1,2,3,4};
/*int a[5]={0,1,2,3,4}; */
/*int a[2]={0,1,2,3,4}; */
/*int a[8]={0,1,2,3,4}; */
/* int a[ ]; */
main() {
        int i;
        int a[ ] = {0,1,2,3,4};
        for (i = 0; i < 5; ++i)
                printf("a[%d] = \t
%d\n", i, a[i]);
}
```

Let's try different ways to define the array

PURDUE UNIVERSITY

What are characters?

```
#include <stdio.h>

main() {
        int i;
        int a[ ] = {'a','b','c','d','e'};
        for (i = 0; i < 5; ++i)
                printf("a[%d] = \t %d\n", i, a[i]);
}
```

PURDUE
UNIVERSITY

What if we really want to see the characters, not the integer code?

8

```
#include <stdio.h>

main() {
        int i;
        int a[ ] = {'a','b','c','d','e'};
        for (i = 0; i < 5; ++i)
                printf("a[%d] = \t %s\n", i, (char *) &a[i]);
}
```

Notice how we recast &a[i] to pointer to character,
Because this is what the “%s” format must match

PURDUE UNIVERSITY

Conversion Specifications in printf()

- Since printf will be a commonly used way for the programmer to trace program execution and data changes, we look at its *conversion specifications* in more details
- The syntax of a conversion specification:
- % [some flag] *conversion character*

PURDUE
UNIVERSITY

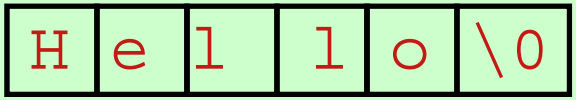
Table B.1 Conversion Characters

Character Argument type;	Printed As
`d,i`	`int`; signed decimal notation.
`c`	`int`; single character, after conversion to `unsigned char`
`s`	`char *`; characters from the string are printed until a `'\0'` is reached or until the number of characters indicated by the precision have been printed.

It is important to terminate a string with the null character '\0'

PURDUE UNIVERSITY

What is a String Anyway?




```
"Hello"
```

- A string literal is a sequence of characters delimited by double quotes
- It has type **`array of char`** and is initialized with the given characters
- The compiler places a null byte (`\0`) at the end of each string literal
- A double-quote (`"`) in a string literal must be preceded by a backslash (`\`)
- Creating an array of character:
  - `char c[6] = "Hello";`

PURDUE UNIVERSITY

Experiment with a few printing statements

```
#include <stdio.h>

main() {
        char a[ ] = "abcde";
        /* char a[10] = "abcde"; */
        /* char a[2] = "abcde"; */
                printf("a[] = \t %s\n",  a);
                printf("a[2] = \t %s\n", &a[2]);
}
```

PURDUE
UNIVERSITY

- Let us also examine the integer coding of ‘\0’ and look at %c in the next experiments
  - char8.c below

```
#include <stdio.h>

main() {
        int i;
        char a[ ] = {'a','b','c','d','e'};
        for (i = 0; i < 5; ++i)
                printf("a[%d] = \t %c\n", i, a[i]);
}
```

PURDUE UNIVERSITY

- Now, look at the ‘\0’ character

```
#include <stdio.h>

main() {

        printf("Null character has integer value \t %d\n", '\0');
        printf("Null character can be printed as \t %c\n", '\0');
        printf("Null character has hexadecimal value  \t %x\n", ‘\0');
}
```

PURDUE UNIVERSITY

- By now we see that the same bit pattern can be presented in different ways by the printf function under different conversion specification
- We can use unsigned hexadecimal format to examine the exact bit pattern

`o`	`int`; unsigned octal notation (without a leading zero).
`x,X`	`unsigned int`; unsigned hexadecimal notation (without a leading `0x` or `0X`), using `abcdef` for `0x` or `ABCDEF` for `0X`.
`u`	`int`; unsigned decimal notation.

```
#include <stdio.h>

main() {
        int n = -1;
        printf("n  decimal \t %i\n",  n);
        printf("n  hexadecimal \t %X\n", n);
        printf("n  octal \t %o\n", n);
        printf("n  unsigned decimal \t %u\n", n);
        printf("n's  address \t %lX\n", (unsigned long) &n);
}
```

PURDUE UNIVERSITY

A New Quiz 0

- Warm up your clickers...

Quiz 0.1

- Is the following true?
- `10 == 010`
- (a) true
- (b) false

PURDUE UNIVERSITY

Answer

- False. Octal 10 is decimal 8

PURDUE UNIVERSITY

Quiz 0.2

- Is the following true?
- `0x11 == 011`
- (a) true
- (b) false

PURDUE UNIVERSITY

Answer

- False. Octal 11 is decimal 9 and hexadecimal 11 is decimal 17

PURDUE UNIVERSITY