

CIS*1500 Lab 11

Introduction to Programming
TA: Alliyah Mo

Today's Lab

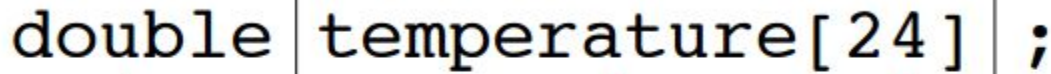
- Overview of arrays
- Review passing arrays to functions
- `ctype` library (character functions)
- using `fgets`
- `string` library (string functions)
- reading from a file using `fgets`

Assignment 2 due on Sunday, November 29th 11:55pm!

Arrays

An array is a consecutive series of variables that share one variable name.

```
double temperature[24] ;
```

A diagram illustrating the syntax of an array declaration. The code 'double temperature[24] ;' is shown. A light gray rectangular box highlights the 'temperature[24]' portion. Three pink arrows point upwards from labels below to specific parts of the code: one from 'type' to 'double', one from '"name"' to 'temperature', and one from 'size' to '[24]'.

type

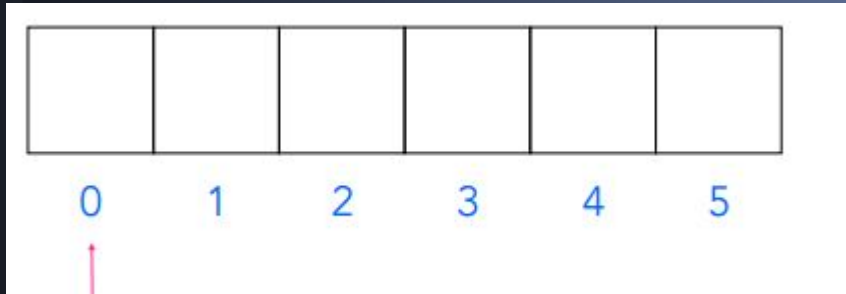
"name"

size

```
type name[size] ;
```

Arrays start at 0

```
int numArray[6] //6 elements
```



Indexed from 0-5

```
int x[4]={1,3,5,7};
```

declares **x** to be a 4-element integer array whose elements have the initial values **x[0]=1**, **x[1]=3**, etc.

Array elements

loop indexes as
array subscripts

```
int i, z[100];
```

```
for (i=0; i<100; i=i+1)  
    z[i] = i;
```

for loop to process
elements in an array

each time the value
of a control variable
is incremented, the
next array element is
selected.

What size to make my array???

Options:

Declare the array statically with a large usable size, ex. 50 - 100.(preferred for CIS*1500) ex. `char array[100];`

Figure out the size of the array then declare it with a variable size (feature of C99 standard)

Dynamic arrays!(CIS*2500)

How many elements?

So far, we've been allocating arrays statically. This means that we cannot increase or reduce the size of our array after it has been declared.

It is possible to create dynamic arrays, but this won't be covered until CIS*2500.

As a rule of thumb, when using static arrays is it better to have too much room as opposed to too little.

Probably not going to want to do `arrayName[99999999]`

Passing arrays to functions

```
#include <stdio.h>

void setGrades(int num, int grades[]);
double classAverage(int num, int grades[]);
void printGrades(int num, int grades[]);

int main()
{
    int numGrades;
    double average;
    int grades[100];

    average = 0.0;
    numGrades = 0;

    printf("How many grades will you like to input\n");
    scanf("%d",&numGrades);

    setGrades(numGrades,grades);
    printGrades(numGrades,grades);

    average = classAverage(numGrades,grades);
    printf("The class average was: %.2lf%%\n", average);

    return 0;
}
```

Quick Warm-up Exercise

Create a program that declares a character array of any size, fill it with your name.

Create a function that prints out your name.

call this function

Submit it to git!

```
git add file.c
```

```
git commit -a -m "my message here"
```

```
git push
```

ctype.h

- a library containing character functions
 - including: (man ctype.h for more)
 - int isalpha(int c)
 - int islower(int c)
 - int isupper(int c)
 - int isspace(int c)
 - int tolower(int c)
 - int toupper(int c)

- **int isalpha(int c)**
 - Returns non-zero if c is an alphabet character, 0 otherwise.
- **int isupper(int c)**
 - Returns non-zero if c is an uppercase letter, 0 otherwise.
- **int islower(int c)**
 - Returns a non-zero if c is a lowercase letter, 0 otherwise
- **int isdigit(int c)**
 - Returns non-zero if c is a digit (0, 1, ..., 9), 0 otherwise.

Warm-up Exercise

Using the ctype library(include it just how we included the math library and the standard library)

Read in a character array of 20 characters, then loop through the array checking if the character is lowerCase, if it is convert it to uppercase.

Then print out the character array

Use and create any functions you feel fit!

Strings!

- strings are just one dimensional arrays of characters with a special ending character
 - the null terminator `'\0'`
- The final character of the array must be a null terminator in order for it to be a string to perform string based functions upon it
- You must size your arrays keeping in mind of this null terminator

```
char hello[] = "hello";
```

Index	0	1	2	3	4	5
Variable	H	e	l	l	o	\0
Address	0x23451	0x23452	0x23453	0x23454	0x23455	0x23456

Null Terminator ('\0')

- The null terminator is a special character that in a character indicates the end of the string
- For a character array to be considered a valid string it must have a null terminator

Format Specifier for Strings

Since these character arrays are strings, we no longer have to print the character array one element at a time

We can use the format specifier %s in our print statements

ex. `printf("Our string %s",hello);`

string.h

- The string library contains dozens of functions that can be used to manipulate and work with strings including:
 - Finding the length of a string
 - Copying one string to another
 - Concatenating strings
 - Comparing 2 strings

Functions in the Library

- This library includes
 - `char *strcat(char *restrict, const char *restrict);`
 - `int strcmp(const char *, const char *);`
 - `char *strcpy(char *restrict, const char *restrict);`
 - `size_t strlen(const char *);`
 - `char *strtok(char *restrict, const char *restrict);`

We'll be using `strlen` and `strcpy` today.

Feel free to experiment using other string functions

strlen

- The function strlen returns the number of characters in the string not including the null terminator.
- The format of strlen is:

size_t strlen(const char *str)

- **str** - The string whose length is to be found
- **size_t** - Is an unsigned integer type meaning it cannot represent negative values.

On success the function returns the length of the string.

strcpy

- The function strcpy will allow you to copy the contents of one string into another.
- The format of strcpy is:

`char * strcpy(char * dest, const char * src)`

- **dest** - The pointer to the destination array where the content is to be copied.
- **src** - The string to be copied.

On success returns a pointer to the destination string **dest**.

strcat

- The function strcat is used to concatenate strings together. That is, it is used to insert the contents of one string to the end of the contents of another string. Also overwriting the null terminator '\0' of the first string.

- The format of strcpy is:

char *strcat(char *dest, const char *src)

- **dest** - The pointer to the destination array, which should contain a C string, and should be large enough to contain the concatenated resulting string
- **src** - The string to be appended. This should not overlap the destination.
- On success, the function returns a pointer to the resulting string dest

More String Functions from stdlib

int atoi(const char *str)

- Converts the string argument str to an integer (type int).

long int strtol(const char *str, char **endptr, int base)

- Converts the initial part of the string in str to a long int value according
- to the given base, which must be between 2 and 36 inclusive, or be the
- special value 0.
- Use base = 10 for regular numbers.

fgets

The answer to life the universe and everything

```
char *fgets(char *s, int size, FILE *stream)
```

The fgets() function reads at most one less than the number of characters specified by size from the given stream and stores them in the string str. Reading stops when a newline character is found, at end-of-file or error. The newline, if any, is retained. If any characters are read and there is no error, a '\0' character is appended to end the string.

It return NULL on failure

Standard Streams include stdin, stdout,stderr

fgets

- The format of fgets is:

```
char * fgets(char * str, max_char, input_stream);
```

- **str** - The pointer to an array of characters where the string read is stored.
- **max_chars** - The maximum number of characters to be read (including the null terminator).
- **input_stream** - The pointer to a FILE pointer that identifies the stream where characters are read from.

On success, **fgets** returns the same **str** parameter. Otherwise, if the End-of-File is encountered and no characters have been read, the contents of **str** remain unchanged and a null pointer is returned

fgets: comes with a newline

```
#include <stdio.h>
#include <string.h>

int main() {
    char str[60];
    int length;

    //Input
    fgets(str, sizeof(str), stdin);

    //Remove the inputted newline
    length = strlen(str);
    if (str[length - 1] == '\n') {
        str[length - 1] = '\0';
    }

    //Output
    printf("%s\n", str);

    return 0;
}
```

String exercise!

Get a string from the user(remember to replace the newline with the null terminator)

Make a copy of this string into another string

Then replace every lowercase letter with a 9, and print out both strings

```
printf(“%s”,string);
```

More Exercises

Under Third Quarter:

Go onto bucky and go through

Lesson: Functions

In Independent Exercises,

Lab 8: Functions

The only way to get better is to practice!!!

Got a question?

- Ask me! :)
- Post on the Forums:
forum.socs.uoguelph.ca
- Email us: cis1500@soecs.uoguelph.ca

Protip: Search the forums and bucky before making a post or sending an email!

Need Extra help?

- Free tutoring offered by TAs! (Book an appointment on bucky) (also see me after Lab)
- Drop-in help hours (right after your Lecture!) (11:30, 2:30 and 5:30) (Tuesday & Thursday)

All meetings will take place in room 001/002 in the basement of Reynolds.

Or are you Incredibly Bored?

And looking for a Challenge? Or something new?



Come see me after lab!

Join GCC!

See bucky for SideQuests!(Also email cis*1500)

Sites To Check To Stay up to Date

Course Website: bucky.socs.uoguelph.ca

SOCS Forums: forum.socs.uoguelph.ca

Textbook: zybooks.com

Reminders!

- **Complete assigned textbook readings before 9 am Tomorrow(Tuesday)!**
- **Protip: Finish assigned readings before your Lab!** (The lab will feel like a light breeze if you do that)
- **Get started on your assignment!**