TA: Xifan Zheng
Email: zhengxifan0403@gmail.com

Welcome to CPSC 441!

Today's Tutorial

- Introduction to C
- Library
- Array
- Structure
- String
- Standard I/O
- File I/O

Tools before you start

- **C/C++ Compiler**
  - Windows: Code::Blocks
  - Linux: gcc
  - Mac OS X: XCode

- **IDE (Integrated Development Environment)**
  - NetBeans/Eclipse

Simple C Example

```
// C
#include <stdio.h>

int main(int argc, char *argv[]) {
  printf("Hello world!\n");
  return 0;
}
```


Compiling C in Linux

- gcc invokes C compiler
- gcc translates C program into executable for some target
- default file name a.out

- Example: compile and run hello.c

```
$ gcc hello.c
$ a.out
Hello, World!

$ gcc hello.c –o hello
$ ./hello
Hello, World!
```


More options

- Some useful command line options:
- [-o file]: specifies the output file for object or executable

- [-Wall]: show all warnings (highly recommended)

- [-l libnam]: Links the library libname, e.g., -lxnet
- If you get errors saying the library cannot be found, make sure the path is correctly set, and you do have the libraries you need.

Libraries

- C provides a set of standard libraries for

numerical math functions	<math.h>	-lm
character strings	<string.h>	
character types	<ctype.h>	
I/O	<stdio.h>	

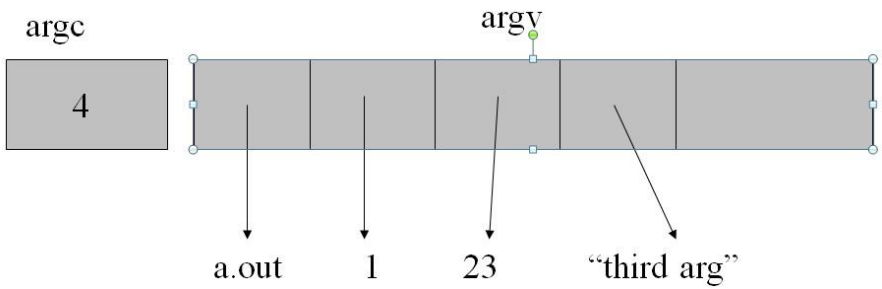
- #include <math.h>
  careful: sqrt(5) without header file may give wrong result!

- gcc –o a main.c –lm

Main arguments

int main(int argc, char *argv[])

- argc: number of arguments passed to the program
- argv: pointers that list all of the arguments
  - Name of executable + space-separated arguments
  - Name of executable is stored in argv[0]
- $ a.out 1 23 'third arg'



- The return value is int
  - convention: 0 means success, > 0 some error

Array

- Array declaration: int a[100];

- C/C++ arrays have no length parameter!

Note: when passing an array to a function, typically you have to pass the array size as a separate argument as well.

- You have to take care of array bounds yourself
- int input[10];
- input[10] = 20; // out of bound!
- input[-1] = 5; // out of bound!
- This code could compile and run, but most likely, you'll see unexpected behavior or crash your program.

- Array's name is a pointer to its first element

Structure

- C struct is a way to logically group related types
  - Is very similar to (but not same as) C++/java classes
  - store many different values in variables of potentially different types under the same name.
- A struct component can be of any type (including other struct types)

- Example:

```
struct database {
 int id_number;
 int age;
 float salary; };
```

```
int main() {
 struct database employee;
employee.age = 22;
employee.id_number = 1;
employee.salary = 12000.21;
 }
```

String - 1

- In C, string is an array of **`char`** terminated with "\0" (a null terminator: '\0')
  - **`"hello" = hello\0`**

- Declaring and initialize a string

```
char str[]="hello"; //initialized string

char *strp="hello";  // a char pointer
```

String - 2

- What if the size of string is not fixed? (e.g. the string is input by user every time it runs)

- Make use of malloc(size)

```
char *ptr;
unsigned int size;
ptr=malloc(size);
//The pointer returned value NULL if there's no memory
  left. Should be checked every time!
int returncode;
returncode=free(ptr);
//The return code is 0 if the release is successful
```


String - 3

- Some useful string handling functions: (remember header: <string.h>)

```
char *str1, *str2;
```

- **strlen() – give the length of string (not include "\0")**

```
int len;
len=strlen(str1);
```

- **strcpy() – copy a string from one place to another**

```
strcpy(str1,str2);
```

- **strstr() – test whether a substring is present in a larger string**

```
int location;
location=strstr(str1,str2);
//if str2 is present in str1, return the first location; otherwise, return null
```

- **strcmp() – compare two strings**

```
int value;
value=strstr(str1,str2);
//if two strings are identical, return 0; otherwise indicating the ASCII order
```

String - 4

- Example

Example 22 from
http://www.iu.hio.no/~mark/CTutorial/CTutorial.html#Strings

Standard I/O - 1

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

- Formatted I/O

```
int scanf(const char *format, ...)
```

– read from standard input and store according to format (ignore space)

```
int printf(const char *format, ...)
```

– write to standard output according to format

Example:

```
int this_is_a_number;
printf( "Please enter a number: " );
scanf( "%d", &this_is_a_number );
printf( "You entered %d", this_is_a_number );
```

Standard I/O - 2

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

- Formatted I/O

gets()- fetches a string from standard input file (which is keyboard here) and put into buffer (compulsory)

```
#define SIZE 255;
char *sptr, buffer[size];
sptr=gets(buffer);
//gets() stop reading when it finds a new line character
```

puts()- sends a string to output file until it finds a NULL. A new line character will be written instead of NULL.

```
int returncode;
char *sptr;
returncode=puts(sptr);
```

File I/O - 1

- **`FILE *fopen(const char *path, const char *mode)`**

  - open a file and return a FILE pointer. Can use the FILE pointer perform input and output functions on the file
  - Mode:

    r Open file for reading
    w Open file for writing
    a Open file for appending
    rw Open file for reading and writing (some systems)

e.g. FILE *fp;

```
if ((fp = fopen (“c:\\test.txt","r")) == NULL)
    { printf ("File could not be opened\n");
      error_handler(); }
```

//if the file is not open successfully, it returns NULL. Always check!!

File I/O - 2

- **`fclose()- close the file, return 0 if success`**

e.g.

```
if (fclose(fp) != 0)
{ printf ("File did not exist.\n")
; error_handler(); }
```

//Always check!!

- **`fprintf() – same as printf(), but it writes to a file`**

e.g.

```
printf ("Hello world %d", 1);
fprintf (fp,"Hello world %d", 1);
```

File I/O - 3

- **`fscanf()- same as scanf(), but it reads from a file: fscanf (fp,format,pointers)`**

e.g.

fscanf (fp, "%d %f %c", &i, &x, &ch);

//it's easy to get error when you are using fscanf, so be careful!

- **`Fgets() – same as gets(), but it readsfrom a file: fgets (pointers, n, fp)`**

e.g.

fgets (strbuff,n,fp);

Reference & Reading

C for Java programmers:

http://faculty.ksu.edu.sa/jebari_chaker/papers/C_for_Java_Programmers.pdf

http://www.cs.columbia.edu/~hgs/teaching/ap/slides/CforJavaProgrammers.ppt

C tutorial:

http://www.iu.hio.no/~mark/CTutorial/CTutorial.html

http://www.cprogramming.com/tutorial/c-tutorial.html

Socket programming with C: (for next session)

Beej's Guide to Network Programming Using Internet Sockets

http://beej.us/guide/bgnet/output/html/multipage/index.html

Thanks for attending!