Basics of C Programming

C vs. Java

- C is procedural, not object oriented
  - C has no objects, interfaces or packages
  - A program only consists of functions and data
- C is compiled, not interpreted
  - Translated directly to assembly language
  - Faster, less portable and very hard to debug.
- C has no array bounds, null pointer or cast checks
  - You have to detect and handle all problems yourself
- C has no garbage collector
  - You have to do all of the memory management yourself

C vs. Java (cont.)

- C has pointers
  - Similar to Java references but...
  - ...they can be used in calculations (pointer arithmetic)
  - Allows you to use the location of data in computations (not just the value)
  - Useful, powerful and a debugging nightmare!
- Compared to Java, C is a low-level language
  - You can and must do everything yourself
  - Suitable for low-level software like device-drivers, communication libraries, operating systems, etc.

- You can implement anything in C!
  - No limits!

C can be quite complex

- This program computes and prints the first 800 decimals of Pi:

```c
#include <stdio.h>
long a=10000,b,c=2800,d,e,f[2801],g;
int main(){
  for(;b-c;)
    f[++b]=a/5;
  for(;d=0,g=c*2;c-=14,printf("%.4d",e+d/a),e=d%a)
    for(b=c;d+=f[b]*a,f[b]=d%--g,d/=g--;b--;d*=b);
}
```
Basic C Program

```c
main()
{
}
```

---

**Basic C Program: Print to stdout**

```c
#include <stdio.h>

main()
{
    printf("Hello, CSC4304 Class\n");
}
```

---

**Header Files**

- The C compiler works in 3 phases:
  - Pre-process source files
  - Compile source files into object files
  - Link object files into an executable.
- `#include <stdio.h>` means “Include the contents of standard file `stdio.h` here”
- Standard files are usually located in directory `/usr/include`
- `/usr/include` directory may contain `#include` statements itself.
- You can use `#include` to include your own files into each other:
  - `#include "myfile.h"` means “Include file `myfile.h` (from the current directory) here”
  - Included files usually have extension "h" (header)

---

**Read argument and print**

```c
#include <stdio.h>

#include <stdio.h>

int main(int argc, char* argv[])
{
    printf("Hello, %s\n", argv[1]);
}
```

---

**Read argument and print**

```c
#include <stdio.h>

int main(int argc, char* argv[])
{
    if (argc < 2){
        printf("Usage: %s <your name>\n", argv[0]);
    }
    else{
        printf("Hello, %s!\n", argv[1]);
    }
}
```
# Read from stdin and print

```c
#include <stdio.h>

main()
{
    char name[64];
    printf("What's your name?");
    scanf("%s", name);
    printf("Hello, %s!
", name);
}
```

---

# Basic Data Types

- **Basic Types**
  - char : character - 1 byte
  - short: short integer - 2 bytes
  - int: integer - 4 bytes
  - long: long integer - 4 bytes
  - float: floating point - 4 bytes
  - double - double precision floating point - 8 bytes

- **Formatting Template**
  - %d: integers
  - %f: floating point
  - %c: characters
  - %s: string
  - %x: hexadecimal
  - %u: unsigned int

---

# Test Size of Data Types

```c
#include <stdio.h>

main()
{
    printf("sizeof(char): %d\n", sizeof(char));
    printf("sizeof(short): %d\n", sizeof(short));
    printf("sizeof(int): %d\n", sizeof(int));
    printf("sizeof(long): %d\n", sizeof(long));
    printf("sizeof(float): %d\n", sizeof(float));
    printf("sizeof(double): %d\n", sizeof(double));
}
```

---

# Formatting

```c
#include <stdio.h>

main()
{
    char var1;
    float f;
    printf("Enter a character: ");
    scanf("%c", &var1);
    printf("You have entered character: %c \ ASCII value=%d \n Address=%x\n", var1, var1, &var1);
    printf(" And its float value would be: %.2f\n", (float)var1);
}
```

---

# Formatting (cont.)

```c
#include <stdio.h>

int main(void) {
    int val = 8;
    char str[] = "world";
    printf("Hello world\n");
    printf("Hello %d World\n", val);
    printf("Hello %c World\n", str[0], val);
    printf("Hello %s World\n", str);
    printf("Hello %s, etc\n", str);
    return 0;
}
```

---

# Arrays

- **Defining an array is easy:**
  ```c
  int a[3]; // a is an array of 3 integers */
  ```

- **Array indexes go from 0 to n-1:**

- **Beware:** in this example a[3] does not exist, but your compiler will not complain if you use it!
  ```c
  int x = a[4]; // x is the value of a[0] */
  ```

- You can create multidimensional arrays:

  ```c
  int matrix[M][N];
  matrix[0][0] = 42;
  ```
Strings

- A string is an array of characters:

```c
char hello[15] = "Hello, world!";
```

- Unlike in Java, you must decide in advance how many characters can be stored in a string.
  - You cannot change the size of the array afterwards
  - Beware: strings are always terminated by a NULL character '\0'
  - For example, "Hello" is string of 6 characters:
    ```c
    | R | e | l | l | o | \0 |
    ```

Manipulating Arrays

- You cannot copy an array into another directly
  - You must copy each element one at a time

```c
int a[3] = {12, 24, 36};
int b[3];
b = a; // This will NOT work! */
b[0]=a[0];
b[1]=a[1];
b[2]=a[2]; // This will work */
```

Manipulating Strings

- There are standard function to manipulate strings:
  - `strcpy(destination, source)` will copy string source into string destination:

```c
char s[10] = "Hello, world!\n";
char b[10];
strcpy(b, s);
```

```
/* Attention. strcpy does not check that destination is large enough to
 accommodate source.*/
char c[10];
strcpy(c, a); /* This will get you in BIG trouble */
```

Manipulating Strings (cont.)

- Instead of strcpy it is always better to use `strncpy`:
  - `strncpy` takes one more parameter to indicate the maximum number of characters to copy:

```c
char s[10] = "Hello, world!";
char c[10];
strncpy(c, a, 9); /* Why 9 instead of 10? */
```

Comparison Operators

- The following operators are defined for basic data types:

```c
if (x == y) {...}
if (x != y) {...}
if (x < y) {...}
if (x <= y) {...}
if (x > y) {...}
if (x >= y) {...}
if (x >= y) && (x==y) {...} /* logical AND */
if (x==y) || (x==y) {...} /* logical OR */
```

- There is no boolean type in C. We use integers instead:
  - 0 means `FALSE`
  - Any other value means `TRUE`

```c
int x;
if (x) {...} /* Equivalent to: if (x!=0) {...} */
if (!x) {...} /* Equivalent to: if (x==0) {...} */
```

Example

```c
#include <stdio.h>

main()
{
    int x = 5;
    int y = 3;

    if (x==y)
        printf("x is equal to y, x=%d, y=%d\n", x, y);
    else
        printf("x is not equal to y, x-axes=%d, y=%d\n", x, y);
}
```
Classical Bugs

- Do not confuse ' & ' and '&&'!

```c
if (expr1) (...) /* This is incorrect but it means something different */
if (expr2) (/* always executed */)  
if (expr3) (/* never executed */)  
```

- Do not confuse ' | ' and '||'!

```c
if (expr1) (...) /* This is incorrect but it means something different */
if (expr2) (...)  
```

Exercise:
- (7 & 8) vs. (7 & & 8)
- (7 & 8) vs. (7 || 8)

Loops

```c
while (x>0){
...
}
do{
...
} while (x>0);
for (x=0; x<3; x++) {...}
```

Functions

- In C, functions can be defined in two ways:

```c
int foo() {        /* function foo returns an int */
   ...
   return (n);
}
void bar(int pl, double p2) { /* function bar returns nothing */
   ...
}
```

- Calling a function is easy:

```c
int n = foo(); /* call function foo() */
bar(2, -4.123); /* call function bar() */
```

Exercises

1. Write a program which defines an integer, a float, a character and a string, then displays their values and their sizes on screen. /*use the sizeof() function*/

2. Write a program which computes and displays fib(n), where n is a parameter taken from command line:

   fib(0) = 0, fib(1) = 1
   If n > 1 then fib(n) = fib(n - 1) + fib(n - 2)

Summary

- C Basics
  - C vs Java
  - Writing to stdout
  - Taking arguments
  - Reading from stdin
  - Basic data types
  - Formatting
  - Arrays and Strings
  - Comparison Operators
  - Loops
  - Functions

Hmm.

Acknowledgments

- Advanced Programming in the Unix Environment by R. Stevens
- The C Programming Language by B. Kernighan and D. Ritchie
- Understanding Unix/Linux Programming by B. Molay
- Lecture notes from B. Molay (Harvard), T. Kuo (UT-Austin), G. Pierre (Vrije), M. Matthews (SC), and B. Knicki (WPI).