Lecture Notes 7

C vs. C++

- All library headers are .h
  - C++ C
  - <cstdio> <stdio.h>
  - <cstdlib> <stdlib.h>
  - <cstring> <string.h>
  - <cctype> <ctype.h>
  - <cstddef> <stddef.h>
- Working with Strings
  - Copying Strings
    - Example C++
      ```
      std::string AString;
      ...
      AString.length(); // Returns length of string
      ```
    - Example C
      ```
      char *AString; // Will point to a C string
      ...
      strlen(AString); // Returns length of string
      ```
  - Copying Strings
    - Example C++
      ```
      std::string Original = "Some string";
      std::string Copy = Original;
      ```
    - Example C
      ```
      char Original[] = "Some string";
      char *Copy;
      // Need to allocate space equal to string length
      // also need to account for null terminator
      Copy = malloc(strlen(Original) +  1);
      // strcpy copies a null terminated string into another
      strcpy(Copy,Original);
      ```
  - Comparing Strings
    - Example C++
      ```
      std::string A, B;
      ...
      if(A == B){
          // do something if A == B
      }
      else if(A < B){
          // do something if A < B
      }
      ```

- Example C
```
char *A, *B;
...
if(strcmp(A,B) == 0){
    // strcmp returns 0 if A == B
}
else if(strcmp(A,B) < 0){
    // strcmp returns < 0 if A < B
}
```
- Concatenating Strings
  - Example C++
```
std::string A, B, C;
...
C = A + B;
```
  - Example C
```
char *A, *B, *C;
...
C = malloc(strlen(A) + strlen(B) + 1);
strcpy(C, A);
strcat(C, B);
```
- I/O
  - Standard Output
    - Example C++
```
int I = 3;
double D = 2.2;
std::string S = "Hello";

std::cout<<"I = "<<I<<std::endl;
std::cout<<"D = "<<D<<std::endl;
std::cout<<"S = "<<S<<std::endl;
```
    - Example C
```
int I = 3;
double D = 2.2;
char S[] = "Hello";

printf("I = %d\n",I);
printf("D = %lf\n",D);
printf("S = %s\n",S);
```
  - Standard Input
    - Example C++
```
int I;
double D;
std::string S;
```

```
std::cin>>I>>D>>S;
```

- Example C

```
int I;
double D;
char S[128]; // Can only accept strings of 127 chars

scanf("%d%lf",&I,&D);
fgets(S,sizeof(S),stdin); // Safer than scanf
```

- File Output
  - Example C++

```
std::ofstream OutFile("out.txt");

OutFile<<"This is a file!"<<std::endl;
```

  - Example C

```
FILE *OutFile = fopen("out.txt","w");

fprintf(OutFile,"This is a file!\n");
fclose(OutFile);
```

- File Input
  - Example C++

```
std::ifstream InFile("in.txt");
int I;
double D;

InFile>>I>>D;
```

  - Example C

```
FILE *InFile = fopen("in.txt","r");
int I;
double D;

fscanf(InFile,"%d%lf",&I,&D);
```

- File Seeking
  - Example C++

```
std::ifstream InFile("in.txt");
InFile.seekg(0, InFile.end);
int FileLength = InFile.tellg();
InFile.seekg(0, InFile.beg);
```

  - Example C

```
FILE *InFile = fopen("in.txt","r");
int FileLength;
fseek(InFile, 0, SEEK_END);
FileLength = ftell(InFile);
```

```
fseek(InFile, 0, SEEK_SET);
```

-

C vs. C++ Functions Pointers

- Function Pointers (Pointer to a function)
  - C++
    ```
    using TFuncPointer = int (*)(int);
    int Foo(int param){
    ...
    }
    ...
    TFuncPointer Ptr = Foo;
    int ReturnVal = Ptr(3);
    ```
  - C
    ```
    typedef int (*TFuncPointer)(int);
    int Foo(int param){
    ...
    }
    ...
    TFuncPointer Ptr = Foo;
    int ReturnVal = Ptr(3);
    ```
- Callback Function – Function that is called when an event occurs (passed as a function pointer)
- Call Data – Parameter that will be passed into Callback Function that will specify the context
- Callback Example – Call the doctor about results and ask for them to call you back
  - C++
    ```
    using TResultsCallback = void (*)(int,void*);
    void CallTheDoctor(TResultsCallback phone, void *who);

    void MyPhoneNumber(int result, void *who){
        Person *Patient = static_cast<Person *>(who);
        Patient->ProcessResults(result);
    }
    ...
    Person Me;
    ...
    CallTheDoctor(MyPhoneNumber,&Me);
    // Give doctor your number and name
    ```
  - C
    ```
    typedef void (*TResultsCallback)(int,void*);
    void CallTheDoctor(TResultsCallback phone, void *who);
    ```

```
void MyPhoneNumber(int result, void *who){
    struct Person *Patient = (Person *)who;
    PatientProcessResults(Patient,result);
}
...
struct Person Me;
...
CallTheDoctor(MyPhoneNumber,&Me);
// Give doctor your number and name
```