Programming Paradigms

CT331 Week 5 Lecture 2

Finlay Smith

Finlay.smith@universityofgalway.ie

Higher Order Functions

Function Definition

```
type name(arg1, arg2, … argN);
```

Examples:

```
int addNumbers(int a, int b);
```

```
void printSomething(char* charArrayToPrint);
```

```
double getHeatTransferCoefficient(double wattPerSqMeter, double temperatureDifferenceInCelsius);
```

Function Pointer Definition

```
type (*name)(arg1, arg2, … argN);
```

Examples:

```
int (*addNumbers)(int a, int b);
```

```
void (*printSomething)(char* charArrayToPrint);
```

```
double (*getHeatTransferCoefficient)(double wattPerSqMeter, double temperatureDifferenceInCelsius);
```

Why Function Pointers?

Motivating example:

```
void sortAsc(int* array){
        ...
        if(array[i] < array[j]){
                swap(array[i],
array[j]
        }
        ...
}
```

```
void sortDesc(int* array){
        ...
        if(array[i] > array[j]){
                swap(array[i],
array[j]
        }
        ...
}
```

Why Function Pointers?

//use comparison function!

```
void sort(int* array, <compare function pointer>){
        ...
        if(compare(array[i],array[j])){
                swap(array[i], array[j]
        }
        ...
}
```

Why Function Pointers?

//use comparison function!

```
void sort(int* array, int (*compare)(int, int)){
        ...
        if(compare(array[i],array[j])){
                swap(array[i], array[j]
        }
        ...
}
```

Why Function Pointers?

```
int compLT(int a, int b){
        Return a < b;
}


int compGT(int a, int b){
        Return a > b;
}
```

Why Function Pointers?

//use comparison function!

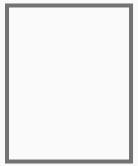
```
int i[] = {1,2,3, … 99};          // pseudocode...

sort(i, &compLT);                 //sorts ascending (1,2,3…)

sort(i, &compGT);                 // sorts descending (99, 98, 97…)
```

Typedef and Function Pointers?

```
void sort(int* array, int (*compare)(int, int)){
        ...
        if(compare(array[i],array[j])){
                swap(array[i], array[j]
        }
        ...
}
```



Typedef and Function Pointers?

```
typedef int (*compare)(int, int);

void sort(int* array, compare compFunc){
        ...
        if(compFunc(array[i],array[j])){
                swap(array[i], array[j]
        }
        ...
}
```

Typedef and Function Pointers?

Syntax [edit]

The syntax for a creating a typedef is: **typedef** *typedeclaration*;[2]

Some examples:

```
typedef int Length;
```

creates Length as a synonym for int.

```
typedef int (*PFI)(char *, char *);
```

creates PFI as a synonym for a pointer to a function of two char * arguments that returns an int.

The linked list currently only accepts char* data. Create the files genericLinkedList.h and genericLinkedList.c files, extend the linked list to accept any data type (Hint: Use void* to create a pointer to any data type.)

[10 marks total]

- Ensure all functions from Question 2 work accordingly.
- Traverse() will not be able to format the data (%d, %s, %ld etc...) so…
- Update the struct to store a function pointer. The function should print out a specific data type (like printStr() will print a string, and printInt() will print an integer) Eg:

```
void printChar(void* data){
        printf("%c\n", *(char*)data);
}
```

When traverse is called, use the function pointer with the element's data to print it out.

Eg:

```
element->printFunction(element->data);
```