Programming Paradigms

CT331 Week 5 Lecture 1

Finlay Smith

Finlay.smith@universityofgalway.ie

Higher Order Functions

Higher Order Functions

A higher order function is a function that:

- Takes one or more functions as arguments (i.e., procedural parameters),
- Returns a function as its result.

Higher Order Functions with function arguments

Pseudocode:

```
add(int x, int y):
    return x + y

subtract (int x, int y) :
    return x - y
```

‘add’ and ‘subtract’ are *first order functions.*

```
add(1,2)
add(5,6)
subtract(7,4)
subtract(10,5)      etc. etc. etc.
```

```
doSomething(func f, int x, int y):
    return f(x,y)


doSomething(add, 1, 2)

doSomething(subtract, 10, 5)
```

doSomething is a higher order function.

Function Pointers

Function Pointers

What is:

```
1. int i;
2. i = 3;
3. &i;
4. int *p;
5. p = &i;
6. printf("%d %d", i, *p);
```

Function Pointers

What is:

```
1. int i;                    Variable declaration
2. i = 3;
3. &i;
4. int *p;
5. p = &i;
6. printf("%d %d", i, *p);
```

Function Pointers

What is:

```
1. int i;                    Variable declaration
2. i = 3;                    Variable assignment
3. &i;
4. int *p;
5. p = &i;
6. printf("%d %d", i, *p);
```

Function Pointers

What is:

```
1. int i;                    Variable declaration
2. i = 3;                    Variable assignment
3. &i;                       Address of i
4. int *p;
5. p = &i;
6. printf("%d %d", i, *p);
```

Function Pointers

What is:

```
1. int i;                    Variable declaration
2. i = 3;                    Variable assignment
3. &i;                       Address of i
4. int *p;                   Pointer variable declaration
5. p = &i;
6. printf("%d %d", i, *p);
```

Function Pointers

What is:

```
1. int i;                    Variable declaration
2. i = 3;                    Variable assignment
3. &i;                       Address of i
4. int *p;                   Pointer variable declaration
5. p = &i;                   Pointer assignment
6. printf("%d %d", i, *p);
```

Function Pointers

What is:

```
1. int i;                    Variable declaration
2. i = 3;                    Variable assignment
3. &i;                       Address of i
4. int *p;                   Pointer variable declaration
5. p = &i;                   Pointer assignment
6. printf("%d %d", i, *p); > 3 3
```

Function Pointers

What is:

```
1. int getNumber();
2. int getNumber(){ return 3;}
3. &getNumber;
4. int (*fp)();
5. fp = &getNumber;
6. printf("%d %d", getNumber(), (*fp)());
```

Function Pointers

What is:

```
1. int getNumber();                    Function declaration
2. int getNumber(){ return 3;}
3. &getNumber;
4. int (*fp)();
5. fp = &getNumber;
6. printf("%d %d", getNumber(), (*fp)());
```

Function Pointers

What is:

```
1. int getNumber();                 Function declaration
2. int getNumber(){ return 3;}      Function definition
3. &getNumber;
4. int (*fp)();
5. fp = &getNumber;
6. printf("%d %d", getNumber(), (*fp)());
```

Function Pointers

What is:

```
1. int getNumber();                  Function declaration
2. int getNumber(){ return 3;}       Function definition
3. &getNumber;                       Address of getNumber func
4. int (*fp)();
5. fp = &getNumber;
6. printf("%d %d", getNumber(), (*fp)());
```

Function Pointers

What is:

```
1. int getNumber();                  Function declaration
2. int getNumber(){ return 3;}       Function definition
3. &getNumber;                       Address of getNumber func
4. int (*fp)();                      Pointer declaration
5. fp = &getNumber;
6. printf("%d %d", getNumber(), (*fp)());
```

Function Pointers

What is:

```
1. int getNumber();                  Function declaration
2. int getNumber(){ return 3;}       Function definition
3. &getNumber;                       Address of getNumber func
4. int (*fp)();                      Pointer declaration
5. fp = &getNumber;                  Pointer assignment
6. printf("%d %d", getNumber(), (*fp)());
```

Function Pointers

What is:

```
1. int getNumber();                  Function declaration
2. int getNumber(){ return 3;}       Function definition
3. &getNumber;                       Address of getNumber func
4. int (*fp)();                      Pointer declaration
5. fp = &getNumber;                  Pointer assignment
6. printf("%d %d", getNumber(), (*fp)()); > 3 3
```