CSEN403: Concepts of Programming Languages
Imperative Programming: C

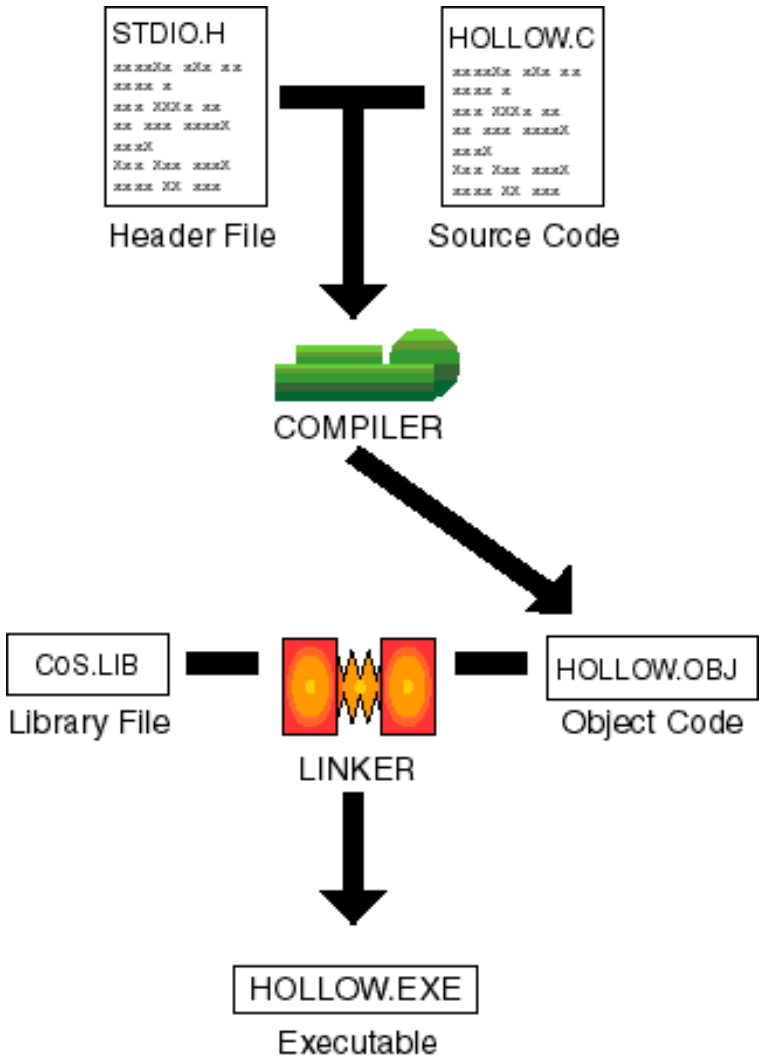
Prof. Dr. Slim Abdennadher
Dr. Nada Sharaf

Spring Semester 2021

Imperative Programming: State

- A variable can be assigned and re-assigned values
- The state is an association between memory locations and values.
- On executing a program, a sequence of states is produced

Executing C


STDIO.H
HOLLOW.C
Header File
Source Code
COMPILER
C0S.LIB
Library File
HOLLOW.OBJ
Object Code
LINKER
HOLLOW.EXE
Executable

Hello World

```
#include <stdio.h>
#include <conio.h>
int main(){
    printf("Hello, World!");
    getch();
    return 0;
}
```

- stdio is a header file for printing
- conio (console in/out) enables getting data from and sending data to the console
- getch() holds the execution

Available types

- int
- char
- float

Other Types

- long
- double

Strings

A string is an array of characters

```
char myName[10] = "Slim";
char* myName = "Slim";
```

Printing

- `printf("%i", counter);`
- `printf("%c", Response);`
- `printf("%f", _averageMileage);`
- `printf("%s", line);`
- `printf("%i %c %f", counter, Response, _averageMileage);`

Incrementing Values I

```
a = 2 ;
b = a++;
```

```
a=3,b=2
```

Incrementing Values I

```
a = 2;
b = ++a;
```

a=3,b=3

Functions in C

- has a *return type*
- has an *identifier*
- has a *parameters list*

Parameter Actual Values

- Call-by-value is used by default
- To use call-by-reference, pointers should be used

Pointers I

```
#include <stdio.h>
int main(){
 int x = 10;
 printf(''The address of %d is %x \n", x, &x);
 return 0;
}
```

Pointers II

```
#include <stdio.h>
int main(){
 int x = 10;
 printf("The address of %d is %x \n", x, &x);
 return 0;
}
```

Pointers I

```
#include <stdio.h>
int main(){
 int x = 10;
 int* ptr;
 ptr = &x;
 printf(âĂIJThe address of %d is %x \nâĂİ, x, &x);
 return 0;
}
```

Pointers

Sending a copy vs. sending the address.

```
void swap(int x, int y){
int tmp;
tmp = x;
x = y;
y = tmp;
}
int main(){
int a = 2, b = 3;
swap(a, b);
return 0;
}
```

```
void swap(int *x, int *y){
int tmp;
tmp = *x;
*x = *y;
*y = tmp;
}
int main(){
int a = 2, b = 3;
swap(&a, &b);
return 0;
}
```

Arrays

- `int arr[] = {4,6,7,2,1,9};`
- How can we access array elements:
  - `arr[0], ..., arr[5]`
  - Access via a pointer

```
 int *ptr;
ptr = &arr[0];

for(i=0; i<6, i++) {
    printf("arr[%d] = %d ", i, arr[i]);
    printf("ptr + %d = %d ", i, *(ptr+i));
}
```

Example

```
#include<stdio.h>
int main()
{
int a = 10, b = 10;
int c,d;
int *ptra = &a;
int *ptrb = &b;
++*ptra;
(*ptrb)++;
printf("\n A = %d , B = %d", a , b);
c = ++*ptra;
d = (*ptrb)++;
printf("\n A = %d , B = %d", a , b);
printf("\n C = %d , D = %d", c , d);
return 0;
}
```

Thank you