CS F111: Computer Programming

(Second Semester 2020-21)

Lect 21: Recursion and Intro to Structures

Dr. Nikumani Choudhury
Asst. Prof., Dept. of Computer Sc. & Information Systems
nikumani@hyderabad.bits-pilani.ac.in

BITS Pilani

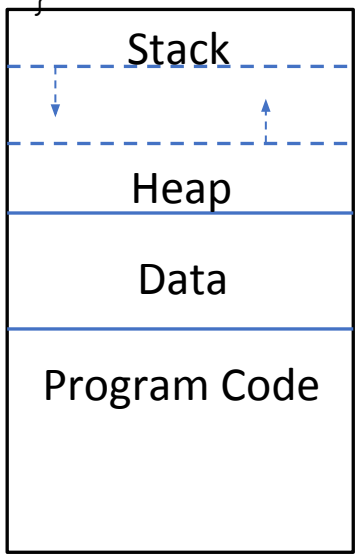
Hyderabad Campus

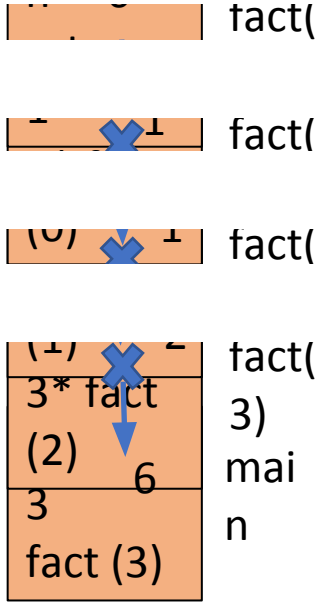
Recursive Functions: Fact

```
#include <stdio.h>
int fact (int n)  {
  if (n == 0)
     return 1;
  else
     return n * fact (n-1);
}

int main()  {
  int num;
  printf ("Enter the
number: ");

  scanf("%d", &num);
  printf("Factorial is %d",

fact(num));
return 0;
}
```

```
Enter the number: 3
Factorial is 6
```

```
#include <stdio.h>
int main(void) {
  static long n;
  printf ("Call number %ld\n",
n++);
  main();
return 0;
}
```






```
...
Call number 523825
Call number 523826
Call number 523827
Call number 523828
Call number 523829
Call number 523830
Call number 523831
Call number 523832
Call number 523833
Call number 523834
Call number 523835
Call number 523836
Call number 523837
Call number 523838
Call number 523839
Call number 523840
Call number 523841
Call number 523842
Call number 523843
Segmentation fault
```

STACK OVERFLOW

Recursive Functions: Sum, Fib

```
#include <stdio.h>
int sum(int n);
int main() {
 int n, result;
 printf("Enter a positive int: ");
 scanf("%d", &n);
 result = sum (n);
 printf("sum = %d", result);
return 0;
}
int sum (int p) {
 if (p != 0)
   return p + sum(p-1);
else
   return p;
}
```

```
Enter any positive integer: 4
sum = 10
Check the call stack in onlinegdb.
```

```
#include<stdio.h>
int fib (int);
int main(void){
 int terms, n;
 printf("Enter terms: ");
 scanf("%d", &terms);
 for (n = 0; n < terms; n++)
 printf("%d ", fib(n));
return 0;
}

int fib(int num){
 if(num == 0 || num == 1)
   return num;
 else
   return fib(num-1) + fib(num-2);
}
```

```
Enter terms: 5
0 1 1 2 3
```

Recursive Functions: Palindrome

```
#include <stdio.h>
#include <string.h>

void checkPalindrome(char [ ], int);

int main(){

 char wordPal[25];
 printf("Input a word: ");
 scanf("%s", wordPal);
 checkPalindrome(wordPal, 0);

return 0;
}
```

- Recursion makes program stylish.
- Used heavily in tree traversals in data structures and divide & conquer models.
- However, if performance is vital, use loops instead, as recursion is usually much slower and takes more space.

```
void checkPalindrome(char wordPal[],
int index){

int len=strlen(wordPal)-(index + 1);

if (wordPal[index] == wordPal[len])
{
 if(index+1 == len || index == len)
  {
    printf(" It is a palindrome\n");
    return;
  }
  checkPalindrome(wordPal,index+1);
}
else {
  printf(" It is not a palindrome");
 }
}
```

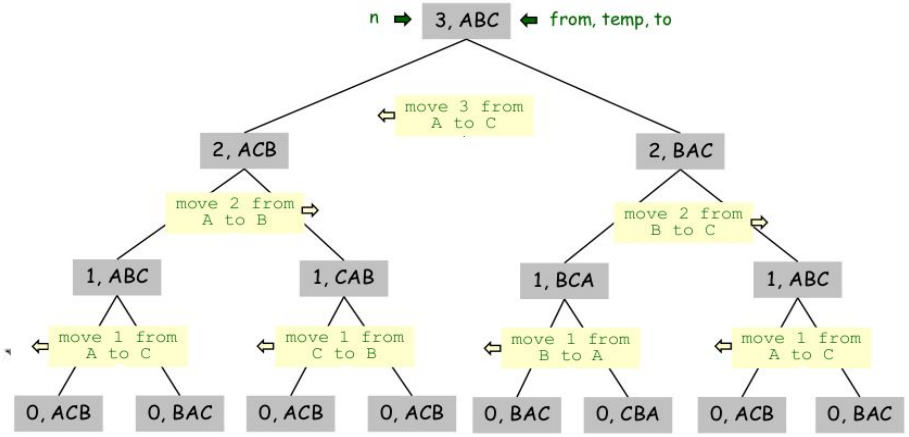
mom, madam, level, noon, radar, …

Recursive Functions: Towers of Hanoi

```
#include <stdio.h>
void t_hanoi(int n, char from, char
                    to, char aux) {
 if (n == 1) {
  printf("\n Move disk 1 from tower
          %c to tower %c", from, to);
  return;
 }
 t_hanoi(n-1, from, aux, to);
 printf("\n Move disk %d from tower %c
           to tower %c", n, from, to);
 t_hanoi(n-1, aux, to, from);
}
int main() {
 int n = 3;
 t_hanoi(n, 'A', 'C','B');
return 0;
}
```

Recursion is better over Iteration…

```
Move disk 1 from tower A to tower C
Move disk 2 from tower A to tower B
Move disk 1 from tower C to tower B
Move disk 3 from tower A to tower C
Move disk 1 from tower B to tower A
Move disk 2 from tower B to tower C
Move disk 1 from tower A to tower C
```



Advantages of recursion

1. The code may be easier to write.
2. To solve such problems which are naturally recursive such as tower of Hanoi.
3. Reduce unnecessary calling of function.
4. Extremely useful when applying the same solution.
5. Recursion reduce the length of code.
6. It is very useful in solving the data structure problem.
7. Stacks evolutions and infix, prefix, postfix evaluations etc.

Disadvantages of recursion

1. Recursive functions are generally slower than non-recursive function.
2. It may require a lot of memory space to hold intermediate results on the system stacks.
3. Hard to analyze or understand the code.
4. It is not more efficient in terms of space and time complexity.
5. The computer may run out of memory if the recursive calls are not properly checked.

User Defined Data Types: Structures and Unions

- A structure is a mechanism for grouping together logically related data items of different types using a single name.

- Exs.-Time: seconds, minutes, hour; Date: day, month, year; Book: author, title, price, year; City: name, country, population; Employee: name, position, PAN no, department, salary; …

  - Recall that an array groups items of a single type (int, float, char etc).
  - Both arrays and structures are structured data types.

Structure Definitions

General format

```
struct [tag_name]
{
   data_type  member1;
   data_type  member2;
   ...
   data_type  membern;
} [var1, var2, …];
```

Example1

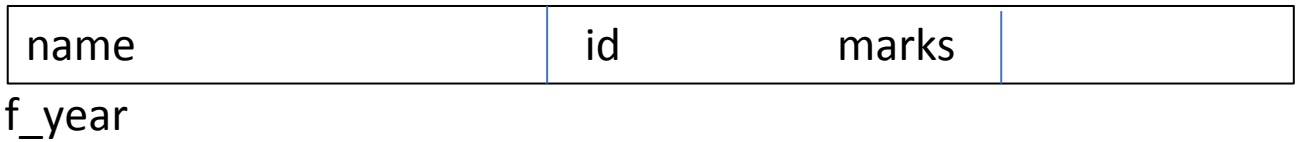
```
struct f_year
{
   char name[30];
   char id[15];
   float marks;
};
```

```
struct f_year stud1;
```

Example2

```
typedef struct
{
   char name[30];
   char id[15];
   float marks;
} f_year;
```

```
f_year stud1;
```



variable declarations

Multiple Variable Declarations

```
struct f_year
{
   char name[30];
   char id[15];
   float marks;
};
```

```
struct f_year
stud1, stud2,
stud3;
```

```
struct
{
   char name[30];
   char id[15];
   float marks;
} stud1, stud2, stud3;
```

(Definition and declaration are combined)

Note: Members do not occupy any memory until they are associated with structure variables.