Lecture 11

Recursion

INFORMATION TECHNOLOGY UNIVERSITY

QUIZ

سورة طه پارہ: 16

قَالَ رَبِّ اشْرَحْ لِيْ صَدْرِيْ ۙ ﴿٢٥﴾

[قَالَ: اس نے کہا] [رَبِّ: اے رب !] [اشْرَحْ: کھول دے] [لِيْ: میرے لیے] [صَدْرِيْ : میرا سینہ]

وَيَسِّرْ لِيْٓ اَمْرِيْ ۙ ﴿٢٦﴾

[وَيَسِّرْ: اور آسان کر دے] [لِيْ: میرے لیے] [اَمْرِيْ : میرا کام]

وَاحْلُلْ عُقْدَةً مِّنْ لِّسَانِيْ ۙ ﴿٢٧﴾

[وَاحْلُلْ: اور کھول دے] [عُقْدَةً: گرہ] [مِّنْ: سے] [لِّسَانِيْ: میری زبان]

يَفْقَهُوْا قَوْلِيْ ۖ ﴿٢٨﴾

[يَفْقَهُوْا: وہ سمجھ سکیں] [قَوْلِيْ : میری بات]

QUESTIONS / FEEDBACK / CONCERNS

UBS

ITU INFORMATION TECHNOLOGY UNIVERSITY

SE SEC A
SLIDE OF FAME

NO ONE
WEEK-1

Muhammad Daniyal Hammad (BSSE23046)
WEEK-2

Syed Hashim Abbas (BSSE23084)
WEEK-3

Umar Ahmad (BSSE23032)
WEEK-4

YOUR NAME
WEEK-5

YOUR NAME
WEEK-6

YOUR NAME
WEEK-7

YOUR NAME
WEEK-8

YOUR NAME
WEEK-9

YOUR NAME
WEEK-10

YOUR NAME
MIDTERM

YOUR NAME
WEEK-11

YOUR NAME
WEEK-12

YOUR NAME
WEEK-13

YOUR NAME
WEEK-14

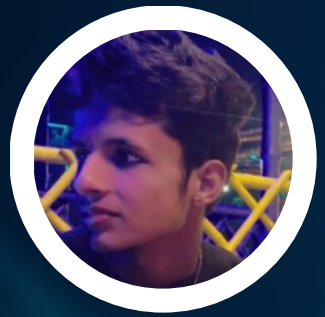
YOUR NAME
WEEK-15

UBS

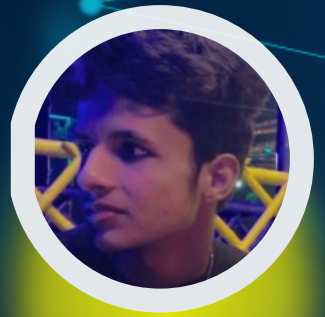
ITU INFORMATION TECHNOLOGY UNIVERSITY

SE SEC B
SLIDE OF FAME

Muhammad Mukarram
BSSE23029
WEEK-1



Muhammad Abdullah
(BSSE23087)
WEEK-2



Muhammad Abdullah
(BSSE23087)
WEEK-3

Fasiha Rohail
(BSSE23041)
WEEK-4

YOUR NAME
WEEK-5

YOUR NAME
WEEK-6

YOUR NAME
WEEK-7

YOUR NAME
WEEK-8

YOUR NAME
WEEK-9

YOUR NAME
WEEK-10

YOUR NAME
MIDTERM

YOUR NAME
WEEK-11

YOUR NAME
WEEK-12

YOUR NAME
WEEK-13

YOUR NAME
WEEK-14

YOUR NAME
WEEK-15

UBS

ITU INFORMATION TECHNOLOGY UNIVERSITY

RECAP

GitHub

Tools (Cygwin, IDE, GitHub)

Approach towards a word problem

Flowcharts

Flowcharts Advantages & Disadvantages

Algorithms

Pseudocode

Numbers Systems (Decimal, Binary, Octal & Hexadecimal)

Ten’s Complement

Twos Complement

main function

Stream in and stream out operators

if else

Functions

Data Types

Arithmetic Operators

Relational Operators

Loops (While, for , do while)

Nested Loops

Switch cases

RECAP

Function Overloading

Scope of variables

Function Prototype and Definition

Default Value in parameters of functions

Parameters by value vs Parameters by Reference

GROUP DISCUSSIONS

Nested Loops

Nested For loop

CODE SYNTAX

```
for ( initialization; condition; increment ) {
   // statement(s) of outer loop

   for ( initialization; condition; increment ) {

      // statement(s) of inside loop
   }

   // statement(s) of outer loop
}
```

https://www.geeksforgeeks.org/nested-loops-in-c-with-examples-2/

Nested While loop

CODE SYNTAX

```
while(condition) {
  // statement(s) of outer loop

  while(condition) {

    // statement(s) of inside loop
  }

  // statement(s) of outer loop
}
```

https://www.geeksforgeeks.org/nested-loops-in-c-with-examples-2/

Nested Do-While loop

CODE SYNTAX

```
do{
  // statement(s) of outer loop
  do{

    // statement(s) of inside loop
  }while(condition);

  // statement(s) of outer loop
}while(condition);
```

https://www.geeksforgeeks.org/nested-loops-in-c-with-examples-2/

There is no rule that a loop must be nested inside its own type. In fact, there can be any type of loop nested inside any type and to any level.

https://www.geeksforgeeks.org/nested-loops-in-c-with-examples-2/

Mix Nested loop

CODE SYNTAX

```
do{
  while(condition) {
        for ( initialization; condition; increment ) {
              // statement(s) of inside for loop
        }
        // statement(s) of inside while loop
  }
  // statement(s) of outer do-while loop
}while(condition);
```

https://www.geeksforgeeks.org/nested-loops-in-c-with-examples-2/

Problem

Print a calendar such that we take number of weeks from user

Week: 1
Day:1
Day:2
Day:3
Day:4
Day:5
Day:6
Day:7
Week: 2
Day:1
Day:2
Day:3
Day:4
Day:5
Day:6
Day:7
Week: 3
...

Problem

Print a calendar such that we take number of weeks from user

Week: 1
Day:2
Day:4
Day:6
Week: 2
Day:2
Day:4
Day:6
Week: 3
...

Problem

Print a calendar such that we take number of weeks from user

Week: 1
Day:2
Day:4
Day:6
Week: 2
Week: 3
Day:2
Day:4
Day:6
Week: 4
Week: 5
Day:2
Day:4
Day:6
...

Prime Factors

"Prime Factorization" is finding which prime numbers multiply together to make the original number.

$$12 = 2 \times 2 \times 3$$

https://www.mathsisfun.com/prime-factorization.html

Prime Factors

Can we divide 147 exactly by 2?

$$147 \div 2 = 73½$$

No it can't. The answer should be a whole number, and 73½ is not.

Let's try the next prime number, 3:

$$147 \div 3 = 49$$

That worked, now we try factoring 49.

The next prime, 5, does not work. But 7 does, so we get:

$$49 \div 7 = 7$$

And that is as far as we need to go, because all the factors are prime numbers.

$$\mathbf{147 = 3 \times 7 \times 7}$$

https://www.mathsisfun.com/prime-factorization.html

Square

```
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

Triangle

```
*
**
***
****
*****
******
```

Output

```
....1
...22
..333
.4444
55555
```

Recursive Functions

$$f(n) = 1 + 2 + 3 + \ldots\ldots + n$$

$$f(n) = 1 \qquad n=1$$
$$f(n) = n + f(n-1) \qquad n>1$$

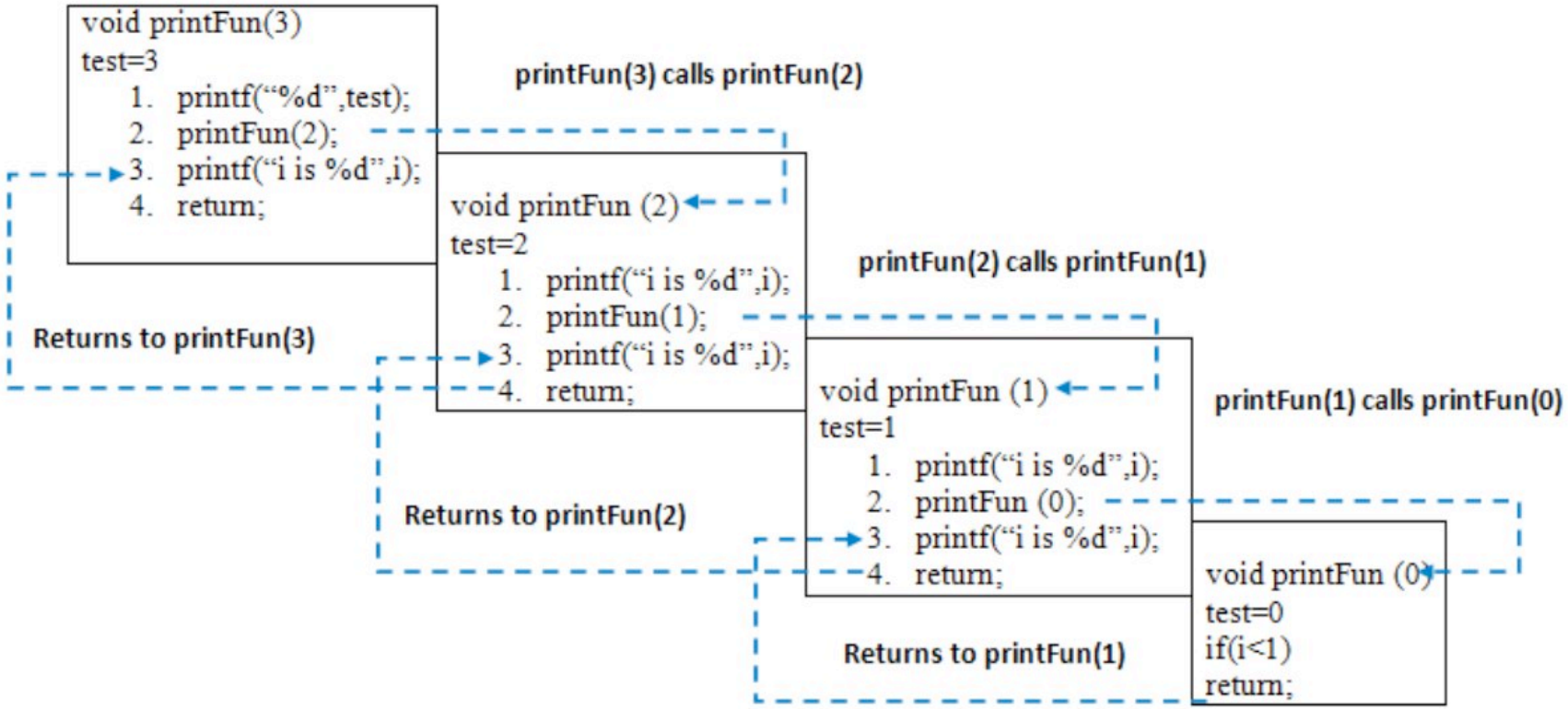
```
int f(int n)
{
    if (n == 1) // base case
        return 1;
    else
        return n+f(n-1);
}
```

```
int fact(int n)
{
    if (n <= 1) // base case
        return 1;
    else
        return n*fact(n-1);
}
```

```
#include <bits/stdc++.h>
using namespace std;

void printFun(int test)
{
   if (test < 1)
      return;
   else {
      cout << test << " ";
      printFun(test - 1); // statement 2
      cout << test << " ";
      return;
   }
}
```

3 2 1 1 2 3



```
// recursive.cpp
// Demonstrates the principle of recursion by a
// function, which reads a line from the keyboard
// and outputs it in reverse order.
// ------------------------------------------------
#include <iostream>
using namespace std;

void getput(void);

int main()
{
   cout << "Please enter a line of text:\n";
   getput();
   cout << "\nBye bye!" << endl;
   return 0;
}

void getput()
{
   char c;
   if( cin.get(c)  &&  c != '\n')
      getput();
   cout.put(c);
}
```

Recursive Functions

Recursive Functions

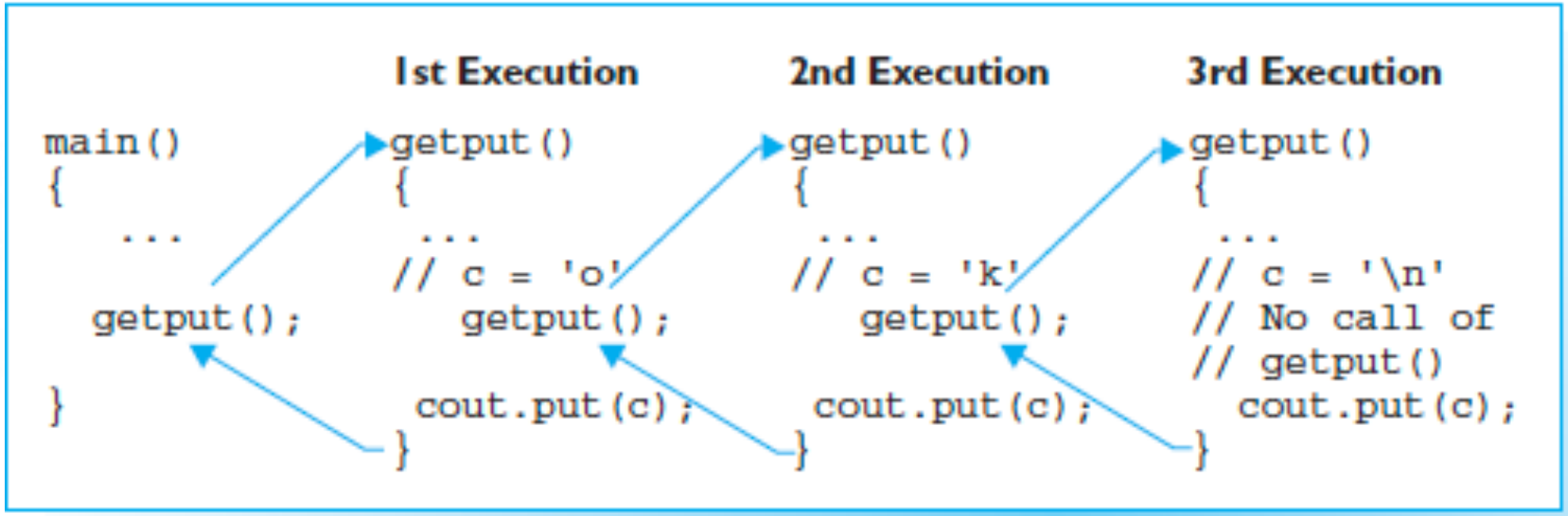


Image Reference : A Complete Guide to Programming in C++

- Series

```
1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144
```

Recursion

- Functions can call themselves.
- fib(n) = fib(n-1) + fib(n-2) can be easily expressed via a recursive implementation

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

Recursion

- Functions can call themselves.
- fib(n) = fib(n-1) + fib(n-2) can be easily expressed via a recursive implementation


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

Recursion

- Functions can call themselves.
- fib(n) = fib(n-1) + fib(n-2) can be easily expressed via a recursive implementation

recursive step

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