AAB University

Faculty of Computer Sciences

Programimi i Orientuar në Objekte

Java 6:

Funksionet dhe Rekurzionet

Doc. Dr. **Mentor Hamiti**

mentor.hamiti@universitetiaab.com

- **Fuksionet e Librarisë standarde**
  - Funksionet e Librarisë “math”
- **Funksionet e Definuara nga Shfrytëzuesi**
  - Funksionet Standarde
  - Funksionet Inline
  - Macro Funksionet
- **Disa veti më specifike të Funksioneve**
  - Referenca dhe Parametrat Referent
  - Argumentet *Defaults*
  - Operatorët e ashtuquajtur *Unary Scope Resolution*
  - Funksionet e mbingarkuara (*Overloading*)
- **Rekursionet**

Hyrje

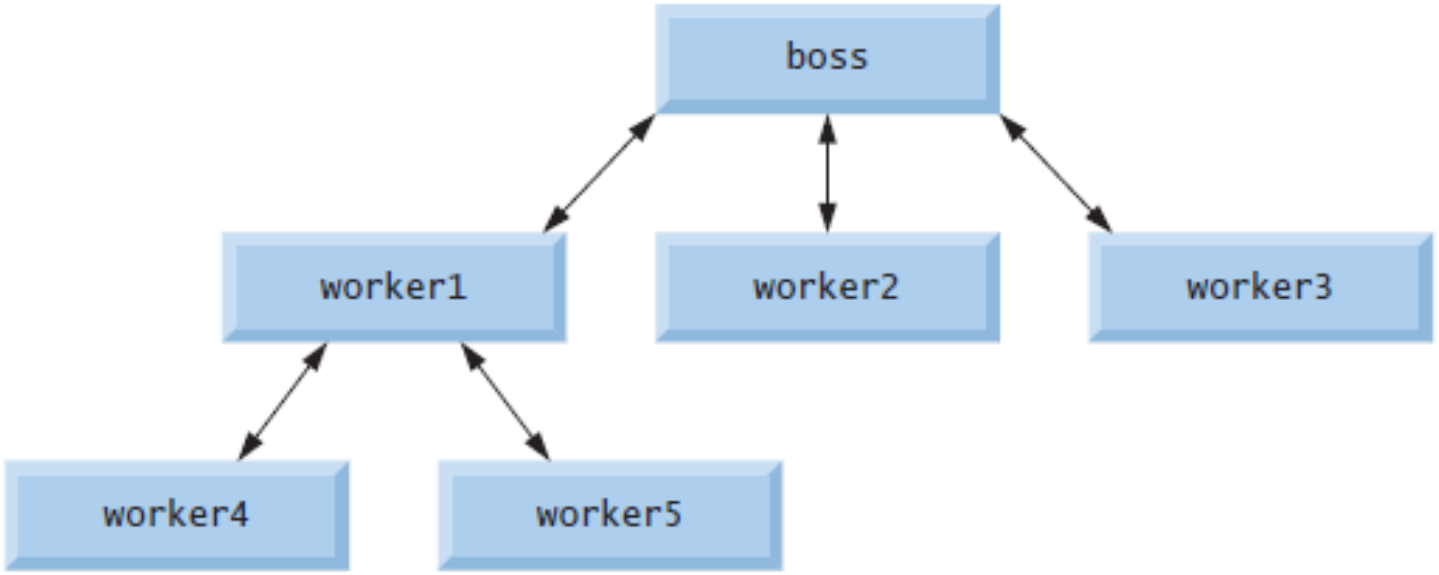
- Eksperienca ka treguar se mënyra më e mirë për të zhvilluar programe komplekse është përmes kompozimit të tyre nga pjesë më të vogla, më të thjeshta dhe ose nga komponente përbërëse të pavaruara
  - Teknika e njohur ndaj dhe bashko (*divide and conquer*)
- C++ programet zakonisht kompozohen përmes funksioneve dhe klasave që ndodhen të paketuara në Libraritë Standarde të C++

Komponentet Programore të C++

- Libraritë Standare në C++ofrojnë koleksione të pasura me funksione të gatshme
- Funksionet lejojnë konceptin **modular** të programit të përbëra nga njësi të pavarura
- Funksionet që shkruajmë ne njihen si **funksione të definuara nga shfrytëzuesi**
- Blloku i urdhërave në kuadër të një funksioni shkruhet vetëm një herë, kurse lejon mundësinë **e rishfrytëzimit** nga disa lokacione të ndryshme dhe njëkohësisht është i fshehur nga qasja prej funksioneve të tjera

Komponentet Programore të C++

- Funksioni që thiret për ekzekutim, zakonisht kthen vlerë ose vetëm ia rikthen kontrollin rrjedhës programore
- Analogjia e njejtë e thirjes së funksioneve shfrytëzohet edhe te hierarkia e menaxhimit ☺



Funksionet e Librarisë ***math***

- Nganjëherë funksionet nuk janë anëtarë të klasave (*class*)
  - Të tillët njihen si **funksione globale**
  - Ata ruhen në header-fajlla, dhe me ri-inkuadrim të fajllave mund të shfrytëzohen në programe të ndryshme
- Funksionet **<cmath>** ofrojnë kolekcion të gjerë të **kalkulimeve matematikore**
  - Të gjitha funksionet në **<cmath>** header-fajllin janë funksione globale

Funksionet e Librarisë ***math***

Function	Description	Example
`ceil( x )`	rounds x to the smallest integer not less than x	`ceil( 9.2 )` is `10.0` `ceil( -9.8 )` is `-9.0`
`cos( x )`	trigonometric cosine of x (x in radians)	`cos( 0.0 )` is `1.0`
`exp( x )`	exponential function e^x	`exp( 1.0 )` is `2.718282` `exp( 2.0 )` is `7.389056`

`fabs( x )`	absolute value of x	`fabs( 5.1 )` is `5.1`
		`fabs( 0.0 )` is `0.0`
		`fabs( -8.76 )` is `8.76`

`floor( x )`	rounds x to the largest integer not greater than x	`floor( 9.2 )` is `9.0` `floor( -9.8 )` is `-10.0`
`fmod( x, y )`	remainder of x/y as a floating-point number	`fmod( 2.6, 1.2 )` is `0.2`

Funksionet e Librarisë ***math***

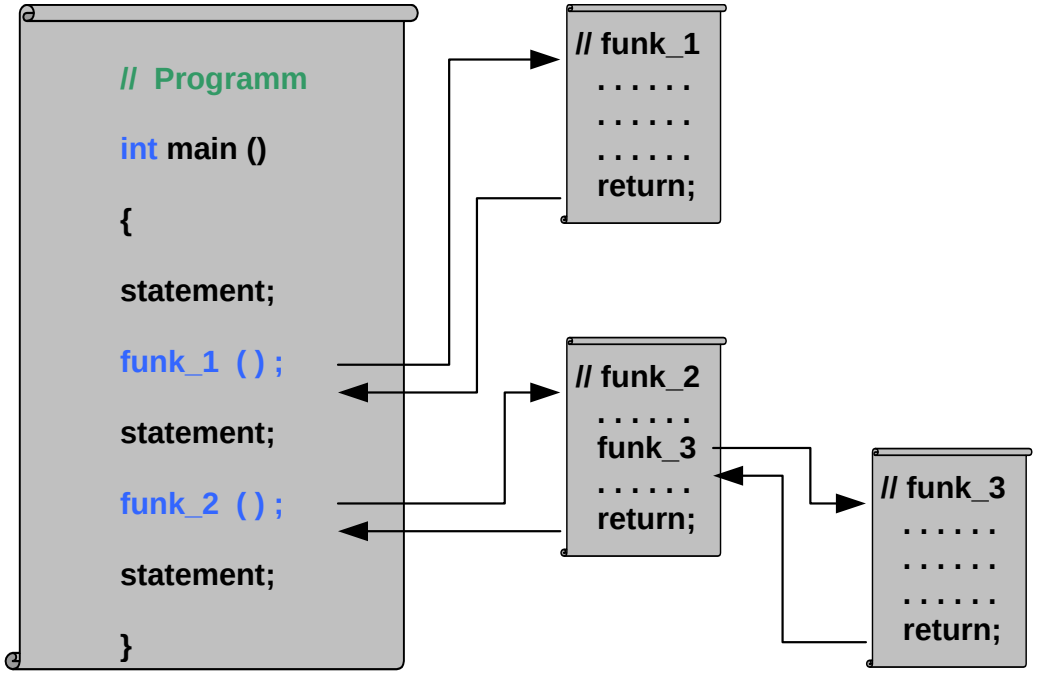
Function	Description	Example
`log( x )`	natural logarithm of x (base e)	`log( 2.718282 ) is 1.0` `log( 7.389056 ) is 2.0`
`log10( x )`	logarithm of x (base 10)	`log10( 10.0 ) is 1.0` `log10( 100.0 ) is 2.0`
`pow( x, y )`	x raised to power y (x^y)	`pow( 2, 7 ) is 128` `pow( 9, .5 ) is 3`
`sin( x )`	trigonometric sine of x (x in radians)	`sin( 0.0 ) is 0`
`sqrt( x )`	square root of x (where x is a nonnegative value)	`sqrt( 9.0 ) is 3.0`
`tan( x )`	trigonometric tangent of x (x in radians)	`tan( 0.0 ) is 0`

Prototipi dhe Argumenti i Funksioneve

- Prototipi i funksionit (deklarimi) përmban informatat vijuese për kompajlerin:
  - Emrin e funksionit
  - Tipin dhe të dhënat që kthen funksioni
  - Numrin e parametrave hyrës të funksionit
  - Tipin e parametrave të tillë
  - Rënditjen e parametrave hyrës

Prototipi dhe Argumenti i Funksioneve

- Funksionet shpesh mund të kenë nevojë edhe për thirje të nënfunksioneve përkatëse
- Gjithashtu egzistojnë edhe funksione pa parametra



Prototipi dhe Argumenti i Funksioneve

- Egzistojnë disa mënyra për rikthim të rrjedhës programore në pozicionin fillestar prej ku është kërkuar thirja e funksionit

  - Nëse funksioni është i tipit "return" pas ekzekutimit të urdhërave që ndodhen në trupin e programit, vlera e llogaritur i kthehet rrjedhës programore në pozicionin e njejtë prej ku është kërkuar thirja

  - Nëse funksioni është i tipit që nuk kthen vlerë (**void**), kontrolli, pas ekzekutimit të urdhërit të fundit në kuadër të trupit të programit, i kthehet rrjedhës programore në pozicionin e njejtë prej ku është kërkuar thirja

Funksionet pa parametra

- Shembulli 1:

Funksion pa parametra

```
//Star Line
# include <iostream>
# include <iomanip>
using namespace std;

void starLine ()
{
    cout<<"\n"<<setw(79)<<setfill('*')<<'*'<<endl;
};

int main()
{
    starLine();

    cout<<"\n\tUniversiteti AAB"<<endl;

    starLine();

    cin.get();
    return 0;
}
```

```
*******************************************************************************
        Universiteti AAB
*******************************************************************************
```

Funksionet me parametra

- Shembulli 2:

Funksion me vlerë kthyese

```
//Maximum Number x, y, z
# include <iostream>
using namespace std;

int maximumNumber (int x, int y, int z)
{
    int maximumValue = x;
    if (y > maximumValue)
        maximumValue = y;
    if (z > maximumValue)
        maximumValue = z;
    return maximumValue;
};

int main()
{
    int Number;
    int a, b, c;
    cout<<"Enter Three Integer Numbers:"<<endl;
    cin>>a>>b>>c;
    Number = maximumNumber (a, b, c);
    cout<<"\nMaximum Number is:"<<Number<<endl;
    cin.get(); cin.get();
    return 0;
}
```

```
Enter Three Integer Numbers:
1
7
6

Maximum Number is:7
```

Declaring function parameters of the same type as `double x, y` instead of `double x, double y` is a syntax error—a type is required for each parameter in the parameter list.

Funksionet me parametra

- Shembulli 3:

Funksion që nuk kthen vlerë

```
//Maximum Number x, y, z
# include <iostream>
using namespace std;

void maximumNumber (int x, int y, int z)
{
     int maximumValue = x;
     if (y > maximumValue)
          maximumValue = y;
     if (z > maximumValue)
          maximumValue = z;
     cout<<maximumValue;
};

int main()
{
     int a, b, c;
     cout<<"Enter Three Integer Numbers:"<<endl;
     cin>>a>>b>>c;
     cout<<"\nMaximum Number is:";
     maximumNumber (a, b, c);
     cin.get(); cin.get();
     return 0;
}
```

```
Enter Three Integer Numbers:
1
7
6

Maximum Number is:7
```

*Funksionet **Inline***

- C++ ofron funksione që reduktojnë numrin e thirjeve të funksioneve

- Me bashkangjitjen e nocionit **in-line** përa tipit të vlerës rikthyese të funksionit, informohet kompajleri që çdo herë kur të ketë thirje të funksionit të tillë të gjenerohet nga një kopje e tillë e funksionit në kuadër të rrjedhës programore, me çka reduktohet thirja e funksioneve

  - Kjo kuptohet, se rrit kodin e programit

Funksionet Inline

- Shembulli 4:

```
// Using an inline function to calculate the volume of a cube.
#include <iostream>
using namespace std;

inline double cube(double side )
{
return side * side * side;      // calculate cube
}

int main()
{
    double sideValue; // stores value entered by user

    cout << "Enter the side length of your cube: ";
    cin >> sideValue; // read value from user

    cout << "Volume of cube with side "<< sideValue << " is " <<cube( sideValue )<< endl;

    cin.get(); return 0;
}
```

Inline Function

```
Enter the side length of your cube: 3
Volume of cube with side 3 is 27
```

Macro Funksionet

- Janë funksione që shkruhen në një rresht të vetëm
- Shembull: Cilën zgjidhje do ta preferoni?!

```
#define MAX (a,b)  (a > b)  ?  a : b
```

OSE

```
int MAX (int a, int b)
{
    if (a > b)
        return a;
    else
        return b;
}
```

Macro Funksionet

- Shembulli 5:

```
//Macro Function
# include <iostream>
using namespace std;

# define cube(a) (a*a*a)

int main()
{

    int x;
    cout<<"Side =";
    cin>>x;

    //V = cube(a);

    cout << "Volume of cube is " << cube(x) << endl;

    cin.get(); return 0;
}
```

Macro Funksion

```
Side = 5
Volume of cube is 125
```

Referencat dhe Parametrat Referent

- Në shumicën e gjuhëve programor, funksionet j'u qasen argumenteve në dy mënyra: përmes vlerave reale ose përmes vlerave referente

  - Dallimi qëndron në faktin se përmes qasjes në vlera, vlera origjinale e lokacionit nuk ndryshon, kurse përmes referencës ndryshon!

  - Deklarimi i vlerave referente zakonisht realizohet përmes simbolit "**&**"

One disadvantage of pass-by-value is that, if a large data item is being passed, copying that data can take a considerable amount of execution time and memory space.

Referencat dhe Parametrat Referent

- Shembulli 6:

```
// Comparing pass-by-value and pass-by-reference with references.
# include <iostream>
using namespace std;

int squareByValue( int number)      // function prototype (value pass)
    { return number *= number; }    // caller's argument not modified

void squareByReference( int &numberRef)     // function prototype (reference pass)
{ numberRef *= numberRef; }         // caller's argument modified

int main()
{
    int x = 2; // value to square using squareByValue
    int z = 4; // value to square using squareByReference

    // demonstrate squareByValue
    cout << "x = " << x << " before squareByValue\n";
    cout << "Value returned by squareByValue: " << squareByValue( x ) << endl;
    cout << "x = " << x << " after squareByValue\n" << endl;
    // demonstrate squareByReference
    cout << "z = " << z << " before squareByReference" << endl;
    squareByReference ( z );
    cout << "z = " << z << " after squareByReference" << endl;
    cin.get(); return 0;
}
```

Reference Parameters

```
x = 2 before squareByValue
Value returned by squareByValue: 4
x = 2 after squareByValue

z = 4 before squareByReference
z = 16 after squareByReference
```

*Argumentet **Default***

- Në disa raste që të evitohen thirjet e vlerave të njejta nga funksionet, ata deklarohen si të tipit *default*
  - Gjithmonë kur thiret funksioni, nëse nuk definohen ndryshe, këta vlera të njejta i ofrohen funksionit si argumente
- **Argumentet Default** zakonisht janë të shfrytëzueshëm nga ana e djathtë e listës së argumenteve!

Argumentet Default

- Shembulli 7:

Default Argumente

```
// Using default arguments.
# include <iostream>
using namespace std;

// function boxVolume calculates the volume of a box
int boxVolume( int length=1, int width=1, int height=1 )
{
return length * width * height;
}

int main()
{

    // no arguments--use default values for all dimensions
    cout << "The default box volume is: " << boxVolume();

    // specify length; default width and height
    cout << "\n\nThe volume of a box with length 10,\n"
        << "width 1 and height 1 is: " << boxVolume(10);

    // specify length and width; default height
    cout << "\n\nThe volume of a box with length 10,\n"
        << "width 5 and height 1 is: " << boxVolume(10, 5);

    cin.get();  return 0;
}
```

```
The default box volume is: 1

The volume of a box with length 10,
width 1 and height 1 is: 10

The volume of a box with length 10,
width 5 and height 1 is: 50
```

Operatori me Rezolucion Unary Scope

- Ekziston mundësia edhe e deklarimit të variablave lokale dhe globale (**local** and **global**) me të njejtin emër

- C++ ofron operatorin *scope resolution operator* "**::**" për qasje në variabla të tilla globale, kur e njeta egziston edhe me emrin lokal

Avoid using variables of the same name for different purposes in a program. Although this is allowed in various circumstances, it can lead to errors.

Operatori me Rezolucion Unary Scope

- Shembulli 8:

```
// Using the unary scope resolution operator.
#include <iostream>
using namespace std;

int number = 7; // global variable named number

int main()
{
    double number = 10.5; // local variable named number

    // display values of local and global variables

    cout << "Local double value of number = " <<number
        << "\nGlobal int value of number = " <<::number<< endl;

    cin.get(); return 0;
}
```

Global ose Local Variabla

```
Local double value of number = 10.5
Global int value of number = 7
```

Functionet e mbingarkuara (Overloaded)

- C++ lejon definimin e më shumë funksioneve me të **njetin emër**, përderisa ata kanë ardumente të tipeve të ndryshme
  - Funksionet e këtilla janë të njohura si funksione të mbingarkuara (function overloading)
- Funksionet e këtilla zakonisht kryejnë veprime të ngjashme ose të njejta, por mbi tipe të ndryshme të të dhënave

Overloading functions that perform closely related tasks can make programs more readable and understandable.

Functionet e mbingarkuara (Overloaded)

- Shembulli 9:

Funksion i mbingarkuar

```
// Overloaded functions
#include <iostream>
using namespace std;

// function square for int values
int square( int x )
{
cout << "square of integer " << x << " is ";
return x * x;
} // end function square with int argument

// function square for double values
double square( double y )
{
cout << "square of double " << y << " is ";
return y * y;
} // end function square with double argument

int main()
{
cout << square( 7 ); // calls int version
cout << endl;
cout << square( 7.5 ); // calls double version
cout << endl;

cin.get(); return 0;
}
```

```
square of integer 7 is 49
square of double 7.5 is 56.25
```

Rekursionet

- **Funksion rekurziv** është funksioni që thëret vetëveten qoftë në mënyrë direkte ose indirekte përmes ndonjë funksioni tjetër

- **Faktorieli**
  - Faktorieli i numrave jonegativ **n**, simboloikisht **n!**, (1! = 1, dhe 0! = 1) është produkti:

$$n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 1$$

- Faktorieli mund të llogaritet edhe pa rekursion, pra përmes iteracioneve (***iteratively***) me shfrytëzim të unazave (*loop*):

```
factorial = 1;
for ( int counter = number; counter >= 1; --counter )
   factorial *= counter;
```

Rekursionet

- Definimi **rekurziv** i faktorielit: :

$$n! = n \cdot (n-1)!$$

- Shembull:

5!

5 * 4!

5 * 4 * 3!

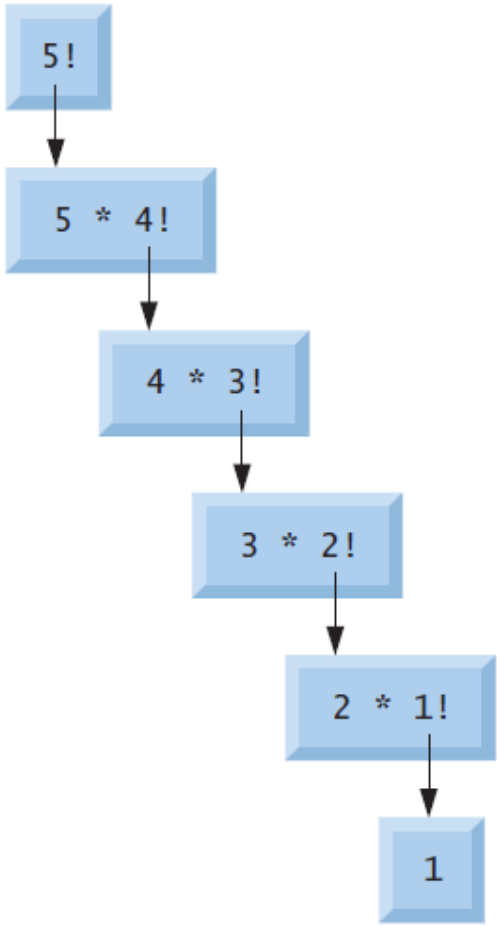
5 * 4 * 3 * 2!

5 * 4 * 3 * 2 * 1!

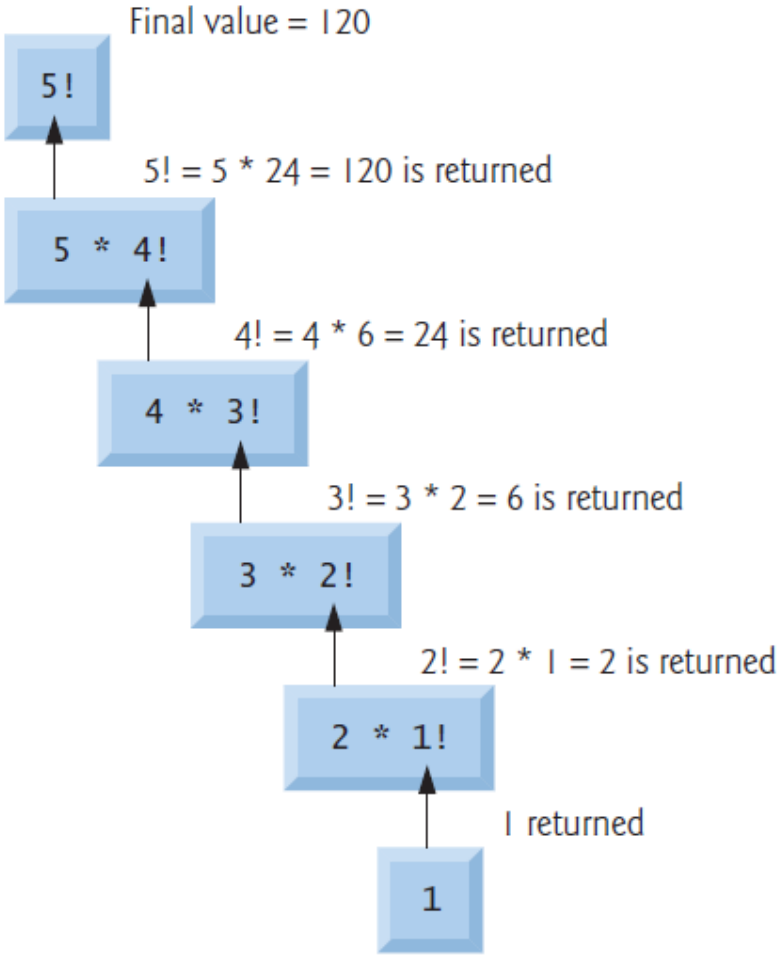
5 * 4 * 3 * 2 * 1 = **120**

Rekursionet

- Evaluimi i faktorielit rekurziv: **5!**:



Procession of recursive calls



Values returned from each recursive call

Rekursionet

- Shembulli 10:

Faktorieli Rekurziv

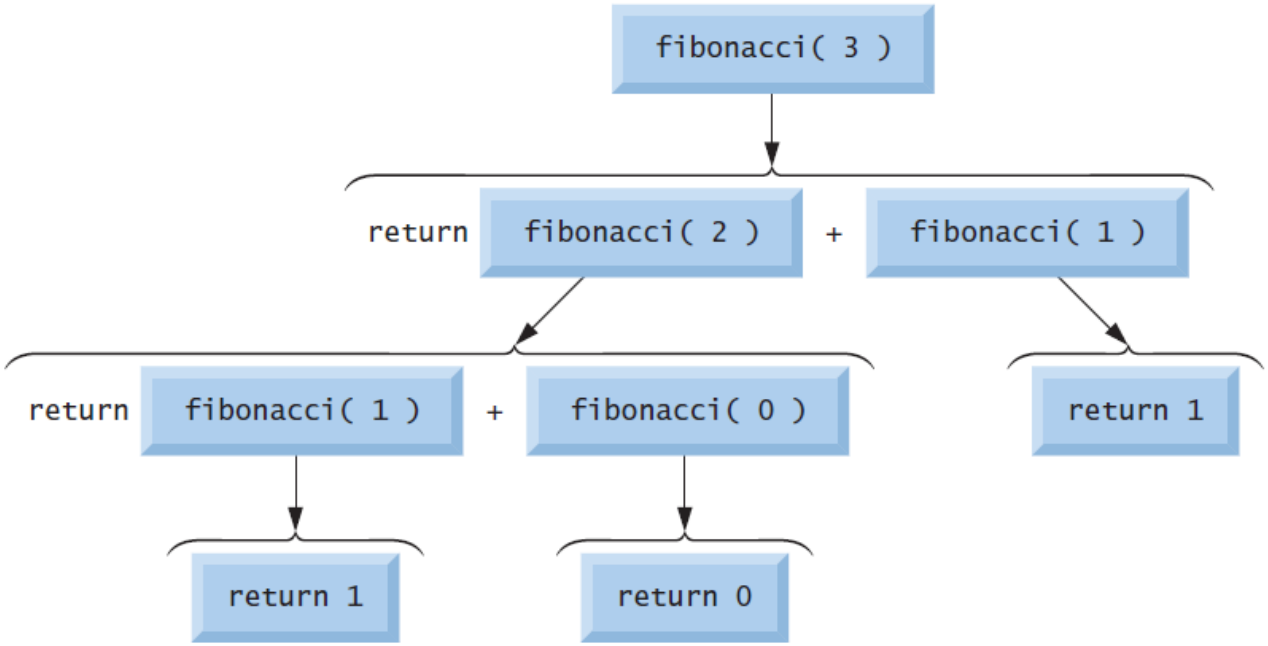
```
n = 5

        Factorial ( 5 ) = 120
```

```
#include <iostream>
using namespace std;

int Factorial (int n);

int main()
{

int n, Result;
cout<<"\n";
cout<<" n = ";  cin>>n;
cout<<"\n\n";

Result=Factorial(n);
cout<<"\n\tFactorial ( "<<n<<" ) = "<<Result;

    cin.get(); cin.get();
    return 0;
}

                    int Factorial (int n)
                        {
                        if (n<=1)
                            return (1);
                        else
                            return (n*Factorial(n-1));
                        }
```

- Vargu **Fibonacci**:

0, 1, 1, 2, 3, 5, 8, 13, 21, ...

Çdo numër vijues i vargut Fibonacci përfaqëson shumë të dy numrave paraprak Fibonacci



Rekursionet

- Shembulli 11:

Fibonacci Rekursiv

```
Fibonacci:
        1, 1, 2, 3, 5, 8, 13, 21, 34, 55, . . .
-----------------------------------------------
n = 10


        fib ( 10 ) = 55
```

```
#include <iostream>
using namespace std;

int fib (int n);

int main()
{

cout<<"\nFibonacci:\n"<<endl;
cout<<"\t1, 1, 2, 3, 5, 8, 13, 21, 34, 55, . . .\n"<<endl;
cout<<"...........................................................\n"<<endl;

int n, Result;
cout<<"n = ";   cin>>n;
cout<<"\n\n";

Result=fib(n);
cout<<"\n\tfib ( "<<n<<" ) = "<<static_cast<float>(Result);
//convert from int to float

    cin.get(); cin.get();
    return 0;
}

                    int fib (int n)
                        {
                        if (n<3)
                            return (1);
                        else
                            return (fib(n-2)+fib(n-1));
                        }
```

- Pyetje eventuale?!

mentor.hamiti@universitetiaab.com