Recursividad

Recursividad

Prof. Ing. José María Sola.

Introducción a Recursividad Por Ing. José María Sola

1. Factorial

Figura 1. Conceptual

2. Fibonacci

$$F(n) = \begin{cases} 0 & n = 0 \\ 1 & n = 1, \\ F(n-1) + F(n-2) & n > 1 \end{cases}$$

Figura 2. Definición recursiva

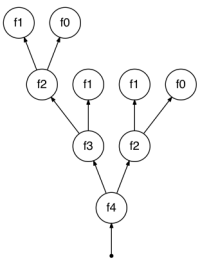

Figura 3. Digrafo de pila de llamadas

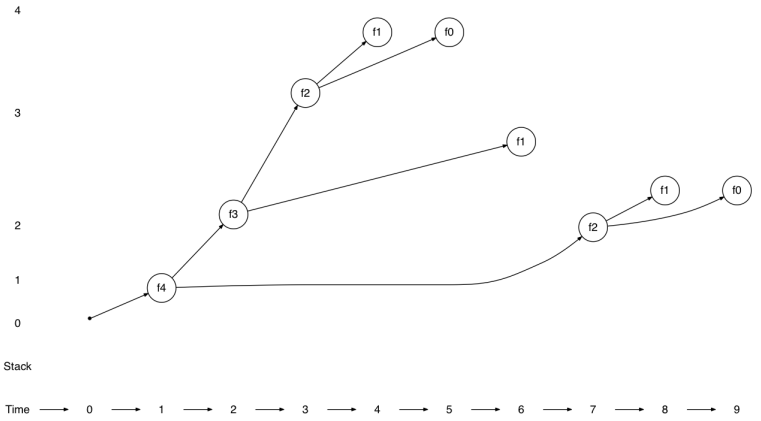

Figura 4. Traza de pila de llamadas

Figura 5. Conceptual

3. Listados Completos

3.1. Factorial

FactorialDriver.cpp

```
/* FactorialDriver
 * 20130822
 * JMS
 */

#include "Factorial.h"
#include <iostream>
#include <iomanip> // setprecision
#include <chrono>

int main(int n, char** argv) {
 std::cout << argv[0] << '\n';

 auto t = std::chrono::steady_clock::now(); // start

 std::cout << "Factorial(17) = " << Factorial(17) << "\n";
```

```
 std::chrono::duration<double,std::micro> d = std::chrono::steady_clock::now() -
 t; // stop
 std::cout << std::setprecision(1) << std::fixed << d.count() << " microseconds\n
\n";
}
```

FactorialIterative.cpp

```
/* FactorialIterative
 * 20130822
 * JMS
 */

#include "Factorial.h"

unsigned Factorial(unsigned n){
 unsigned f=1;

 for(unsigned i=1; i<=n; ++i)
  f *= i;

 return f;
}
```

FactorialRecursiveIfElse.cpp

```
/* FactorialRecursiveIfElse
 * 20130822
 * JMS
 */

#include "Factorial.h"

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

FactorialRecursiveIf.cpp

```
/* FactorialRecursiveIf
 * 20130822
 * JMS
 */
```

```
#include "Factorial.h"

unsigned Factorial(unsigned n){
 if(n == 1)
  return 1;
 return n * Factorial(n-1);
}
```

FactorialRecursiveSwitch.cpp

```
/* FactorialRecursiveSwitch
 * 20130822
 * JMS
 */

#include "Factorial.h"

unsigned Factorial(unsigned n){
 switch(n){
  case 0 : return 1;
  default: return n * Factorial(n-1);
 }
}
```

Fatorial output

```
./FactorialIterative
Factorial(17) = 4006445056
17.2 microseconds

./FactorialRecursiveIfElse
Factorial(17) = 4006445056
15.2 microseconds

./FactorialRecursiveIf
Factorial(17) = 4006445056
11.2 microseconds

./FactorialRecursiveSwitch
Factorial(17) = 4006445056
11.9 microseconds
```

3.2. Fibonacci

FibonacciDriver.cpp

```
/* FibonacciDriver
```

```
 * 20130822
 * JMS
 */

#include "Fibonacci.h"
#include <iostream>
#include <iomanip> // setprecision
#include <chrono>

int main(int n, char** argv) {
 std::cout << argv[0] << '\n';

 auto t = std::chrono::steady_clock::now(); // start

 for(unsigned i=0; i<31; ++i)
  std::cout << "F" << i << " = " << Fibonacci(i) << '\n';

 std::chrono::duration<double,std::micro> d = std::chrono::steady_clock::now() -
 t; // stop
 std::cout << std::setprecision(3) << std::fixed << d.count() << " microseconds.\n
\n";
}
```

FibonacciIterative.cpp

```
/* FibonacciIterative
 * 20130822
 * JMS
 */

#include "Fibonacci.h"

unsigned Fibonacci(unsigned n) {
 if(n<2)
  return n;

 unsigned f;
 for(unsigned p=0, pp=1, i=0; i<n; ++i){
  f = p + pp;
  pp = p;
  p = f;
 }
 return f;
}
```

FibonacciRecursiveIfElse.cpp

```
/* FibonacciRecursiveIf
 * 20130822
 * JMS
 */

#include "Fibonacci.h"

unsigned Fibonacci(unsigned n){
 if(n<2)
  return n;
 else
  return Fibonacci(n-1) + Fibonacci(n-2);
}
```

FibonacciRecursiveIf.cpp

```
/* FibonacciRecursiveIf
 * 20130822
 * JMS
 */

#include "Fibonacci.h"

unsigned Fibonacci(unsigned n){
 if(n<2)
  return n;
 return Fibonacci(n-1) + Fibonacci(n-2);
}
```

FibonacciRecursiveSwitch.cpp

```
/* FibonacciRecursiveSwitch
 * 20130822
 * JMS
 */

#include "Fibonacci.h"

unsigned Fibonacci(unsigned n) {
 switch(n){
  case 0:  return 0;
  case 1:  return 1;
  default: return Fibonacci(n-1) + Fibonacci(n-2);
 }
}
```

FibonacciRecursiveSwitchFallthrough.cpp

```
/* FibonacciRecursiveSwitchFallthrough
 * 20130822
 * JMS
 */

#include "Fibonacci.h"

unsigned Fibonacci(unsigned n){
 switch(n){
  case 0:
  case 1:  return n;
  default: return Fibonacci(n-1) + Fibonacci(n-2);
 }
}
```

FibonacciMemoizationCode.cpp

```
/* FibonacciMemoizationCode
 * 20140825
 * JMS
 */

#include "Fibonacci.h"

unsigned Fibonacci(unsigned n){
 switch(n){
  case  0: return 0;
  case  1: return 1;
  case  2: return 1;
  case  3: return 2;
  case  4: return 3;
  case  5: return 5;
  case  6: return 8;
  case  7: return 13;
  case  8: return 21;
  case  9: return 34;
  case 10: return 55;
  case 11: return 89;
  case 12: return 144;
  case 13: return 233;
  case 14: return 377;
  case 15: return 610;
  case 16: return 987;
  case 17: return 1597;
  case 18: return 2584;
```

```
  case 19: return 4181;
  case 20: return 6765;
  default: return Fibonacci(n-1) + Fibonacci(n-2);
 }
}
```

FibonacciMemoizationStaticData.cpp

```
/* FibonacciMemoizationStaticData
 * 20140825
 * JMS
 */

#include "Fibonacci.h"
#include <array>

unsigned Fibonacci(unsigned n){
 static std::array<unsigned,21> f={{
  0,
  1,
  1,
  2,
  3,
  5,
  8,
  13,
  21,
  34,
  55,
  89,
  144,
  233,
  377,
  610,
  987,
  1597,
  2584,
  4181,
  6765
 }};
 if(n<21)
  return f.at(n);
 return Fibonacci(n-1) + Fibonacci(n-2);
}
```

Fibonacci output

```
./FibonacciIterative
F0 = 0
F1 = 1
F2 = 1
F3 = 2
F4 = 3
F5 = 5
F6 = 8
F7 = 13
F8 = 21
F9 = 34
F10 = 55
F11 = 89
F12 = 144
F13 = 233
F14 = 377
F15 = 610
F16 = 987
F17 = 1597
F18 = 2584
F19 = 4181
F20 = 6765
F21 = 10946
F22 = 17711
F23 = 28657
F24 = 46368
F25 = 75025
F26 = 121393
F27 = 196418
F28 = 317811
F29 = 514229
F30 = 832040
113.563 microseconds.

./FibonacciRecursiveIfElse
F0 = 0
F1 = 1
F2 = 1
F3 = 2
F4 = 3
F5 = 5
F6 = 8
F7 = 13
F8 = 21
F9 = 34
F10 = 55
F11 = 89
```

```
F12 = 144
F13 = 233
F14 = 377
F15 = 610
F16 = 987
F17 = 1597
F18 = 2584
F19 = 4181
F20 = 6765
F21 = 10946
F22 = 17711
F23 = 28657
F24 = 46368
F25 = 75025
F26 = 121393
F27 = 196418
F28 = 317811
F29 = 514229
F30 = 832040
31733.028 microseconds.

./FibonacciRecursiveIf
F0 = 0
F1 = 1
F2 = 1
F3 = 2
F4 = 3
F5 = 5
F6 = 8
F7 = 13
F8 = 21
F9 = 34
F10 = 55
F11 = 89
F12 = 144
F13 = 233
F14 = 377
F15 = 610
F16 = 987
F17 = 1597
F18 = 2584
F19 = 4181
F20 = 6765
F21 = 10946
F22 = 17711
F23 = 28657
F24 = 46368
F25 = 75025
```

```
F26 = 121393
F27 = 196418
F28 = 317811
F29 = 514229
F30 = 832040
27901.888 microseconds.

./FibonacciRecursiveSwitch
F0 = 0
F1 = 1
F2 = 1
F3 = 2
F4 = 3
F5 = 5
F6 = 8
F7 = 13
F8 = 21
F9 = 34
F10 = 55
F11 = 89
F12 = 144
F13 = 233
F14 = 377
F15 = 610
F16 = 987
F17 = 1597
F18 = 2584
F19 = 4181
F20 = 6765
F21 = 10946
F22 = 17711
F23 = 28657
F24 = 46368
F25 = 75025
F26 = 121393
F27 = 196418
F28 = 317811
F29 = 514229
F30 = 832040
36244.022 microseconds.

./FibonacciRecursiveSwitchFallthrough
F0 = 0
F1 = 1
F2 = 1
F3 = 2
F4 = 3
F5 = 5
```

```
F6 = 8
F7 = 13
F8 = 21
F9 = 34
F10 = 55
F11 = 89
F12 = 144
F13 = 233
F14 = 377
F15 = 610
F16 = 987
F17 = 1597
F18 = 2584
F19 = 4181
F20 = 6765
F21 = 10946
F22 = 17711
F23 = 28657
F24 = 46368
F25 = 75025
F26 = 121393
F27 = 196418
F28 = 317811
F29 = 514229
F30 = 832040
35778.742 microseconds.

./FibonacciMemoizationCode
F0 = 0
F1 = 1
F2 = 1
F3 = 2
F4 = 3
F5 = 5
F6 = 8
F7 = 13
F8 = 21
F9 = 34
F10 = 55
F11 = 89
F12 = 144
F13 = 233
F14 = 377
F15 = 610
F16 = 987
F17 = 1597
F18 = 2584
F19 = 4181
```

```
F20 = 6765
F21 = 10946
F22 = 17711
F23 = 28657
F24 = 46368
F25 = 75025
F26 = 121393
F27 = 196418
F28 = 317811
F29 = 514229
F30 = 832040
146.787 microseconds.

./FibonacciMemoizationStaticData
F0 = 0
F1 = 1
F2 = 1
F3 = 2
F4 = 3
F5 = 5
F6 = 8
F7 = 13
F8 = 21
F9 = 34
F10 = 55
F11 = 89
F12 = 144
F13 = 233
F14 = 377
F15 = 610
F16 = 987
F17 = 1597
F18 = 2584
F19 = 4181
F20 = 6765
F21 = 10946
F22 = 17711
F23 = 28657
F24 = 46368
F25 = 75025
F26 = 121393
F27 = 196418
F28 = 317811
F29 = 514229
F30 = 832040
60.976 microseconds.
```