From Mathematics to Generic Programming

Generated by Doxygen 1.8.11

Contents

1 File Index

1.1 File List

Here is a list of all documented files with brief descriptions:

2 File Documentation

2.1 ch02.hpp File Reference

Egyptian Multiplication.

Functions

- int multiply0 (int n, int a)
- bool odd (int n)
- int half (int n)
- int multiply1 (int n, int a)

2.1.1 Detailed Description

Egyptian Multiplication.

Author

Eric Bailey

Date

2016-12-11

Definition in file ch02.hpp.

2.1.2 Function Documentation

2.1.2.1 int half (int *n*)

Return half of a given number.

$$\text{half}(n) = \frac{n}{2}$$

Definition at line 44 of file ch02.hpp.

```
00045 {
00046   return n >> 1;
00047 }
```

2.1.2.2 int multiply0 (int *n*, int *a*)

Add instances of `a` together `n` times.

Efficiency: $\mathcal{O}(n)$

Parameters

n	the number of instances of `a` to add.
a	the number to add `n` times.

Returns

$n \times a$

$$1a = a \tag{2.1}$$

$$(n+1)a = na + a \tag{2.2}$$

Definition at line 16 of file ch02.hpp.

```
00017 {
00019   if (n == 1) return a;
00021   return multiply0(n - 1, a) + a;
00022 }
```

2.1.2.3 int multiply1 (int *n*, int *a*)

"Egyptian multiplication" aka the "Russian Peasant Algorithm"

$$\begin{aligned} 4a &= ((a+a)+a)+a \\ &= (a+a)+(a+a) \end{aligned}$$

The law of associativity of addition:

$$a + (b + c) = (a + b) + c$$

Power of 2	1-bits	Doublings
1	✓	59
2		118
4		236
8	✓	472
16		944
32	✓	1888

$$41 \times 59 = (1 \times 59) + (8 \times 59) + (32 \times 59)$$

Efficiency: $\mathcal{O}(\log_2 n)$

$$\begin{aligned} \#_+(n) &= \lfloor \log_2 n \rfloor + (v(n) - 1) \\ v(n) &= \text{the number of 1s in the binary representation of } n\text{, i.e. the } \textit{population count} \end{aligned}$$

Definition at line 82 of file ch02.hpp.

```
00083 {
00084   if (n == 1) return a;
00085   int result = multiply1(half(n), a + a);
00086   if (odd(n)) result = result + a;
00087   return result;
00088 }
```

2.1.2.4 bool odd (int *n*)

Determine whether a number is odd.

$$n = \frac{n-1}{2} + \frac{n-1}{2} + 1 \Longrightarrow \mathsf{odd}(n)$$

$$\mathsf{odd}(n) \Longrightarrow \mathsf{half}(n) = \mathsf{half}(n-1)$$

Definition at line 34 of file ch02.hpp.

```
00035 { return n & 0x1;
00036 }
```

2.2 ch02.hpp

```
00001
00016 int multiply0(int n, int a)
00017 {
00019   if (n == 1) return a;
00021   return multiply0(n - 1, a) + a;
00022 }
00023
00034 bool odd(int n)
00035 { return n & 0x1;
00036 }
00037
00044 int half(int n)
00045 {
00046   return n >> 1;
00047 }
00048
00082 int multiply1(int n, int a)
00083 {
00084   if (n == 1) return a;
00085   int result = multiply1(half(n), a + a);
00086   if (odd(n)) result = result + a;
00087   return result;
00088 }
```

Index