Peeter Joot
peeterjoot@protonmail.com

Integer square root

In [1] is a rather mysterious looking constant expression formula for an integer square root. This is a function that returns the smallest integer for which the square is less than the value to take the root of. Check out the black magic he used

```
// Stroustrup 10.4:  constexpr capable integer square root function
constexpr int isqrt_helper( int sq, int d, int n )
{
  return sq <= n ? isqrt_helper( sq + d, d + 2, n ) : d ;
}

constexpr int isqrt( int n )
{
  return isqrt_helper( 1, 3, n )/2   1 ;
}
```

The point of this construction was really to illustrate that it allows complex expressions to be used as compile time constants. I wonder at what point various compilers will give up trying to evaluate such expressions?

1.1 Let's take this apart a bit.

Consider the first few values of $n > 0$.

- $n = 0$. Here we have a call to isqrt_helper$(1, 3, 0)$ so the $1 \le 0$ predicate is false, and the return value is just 3.

  For that value we have (using integer arithmetic):

  $$\frac{3}{2} - 1 = 0, \tag{1.1}$$

  as desired.

- $n = 1$. Here we have a call to isqrt_helper$(1, 3, 1)$ so the $1 \le 1$ predicate is true, resulting in a second call isqrt_helper$(4, 5, 1)$. For that call the $4 \le 1$ predicate is false, resulting in a return value of 5.

This time we have a final result of

$$\frac{5}{2} - 1 = 1, \tag{1.2}$$

as desired again. The result will be the same for any value $n \in [1, 3]$.

- $n = 4$. We will end up with a call to isqrt_helper$(4, 5, 4)$ for which the $4 \le 4$ predicate is true, resulting in a followup call of isqrt_helper$(9, 7, 4)$. For that call the $9 \le 4$ predicate is false, resulting in a return value of 7.

  This time we have a final result of

$$\frac{7}{2} - 1 = 2, \tag{1.3}$$

  as expected. We get the same result for any value $n \in [4, 8]$.

1.2 Recurrence relations

The rough pattern of the magic involved can be seen. We have a sequence of calls

- isqrt_helper$(1, 3, n)$,
- isqrt_helper$(4, 5, n)$,
- isqrt_helper$(9, 7, n)$,
- isqrt_helper$(16, 9, n)$,

which terminates at the point where the first (square) parameter exceeds that value that we are taking the root of. Let the parameters of the sequence of calls be s_k, and d_k, so that with $s_0 = 1, d_0 = 3$ the $k \in [0, ...]$ call to the helper function is $q_k = \text{isqrt\_helper}(s_k, d_k, n)$.

The sequence for the second parameter, the eventual return value, can be summarized compactly as $d_k = 3 + 2k$. It is not entirely obvious how we end up with a square for the values $s_k = s_{k-1} + d_{k-1}$, but this follows by summation. For $k > 1$ that is

$$
\begin{aligned}
s_k &= s_{k-1} + d_{k-1} \\
&= s_0 + d_0 + d_1 + d_{k-1} \\
&= s_0 + \sum_{m=0}^{k-1} d_m \\
&= s_0 + \sum_{m=0}^{k-1} (3 + 2m) \\
&= s_0 + \sum_{m=1}^{k} (3 + 2(m-1)) \\
&= s_0 + \sum_{m=1}^{k} (1 + 2m) \\
&= 1 + k + 2 \sum_{m=1}^{k} m \\
&= 1 + k + 2 \frac{k(k+1)}{2} \\
&= k^2 + 2k + 1 \\
&= (k+1)^2.
\end{aligned} \tag{1.4}
$$

This clearly holds for the boundary cases $k = 0, 1$ as well. This allows the helper function action to be summarized more compactly

$$
\mathrm{isqrt\_helper}(1, 3, n) = 3 + 2k, \tag{1.5}
$$

where k is the smallest integer such that $(k+1)^2 > n$. After integer scaling the final result is

$$
(3 + 2k)/2 - 1 = k. \tag{1.6}
$$

This little beastie makes sense after deconstruction, but it was very Jackson like to toss this into the book without comment or explanation.

As pointed out by Pramod Gupta, there's a spooky appearance of collaboration between Stroustrup and Jackson's publishers, not entirely limited to the book covers.

Bibliography

[1] Bjarne Stroustrup. *The C++ Programming Language, 4th Edition*. Addison-Wesley, 2014. 1