

```

struct TwoTierStruct
{
    POINT[2] pts;
    RECT *   prcOptional;
    long[10] extra;
    long *   pLong;
}

```

```

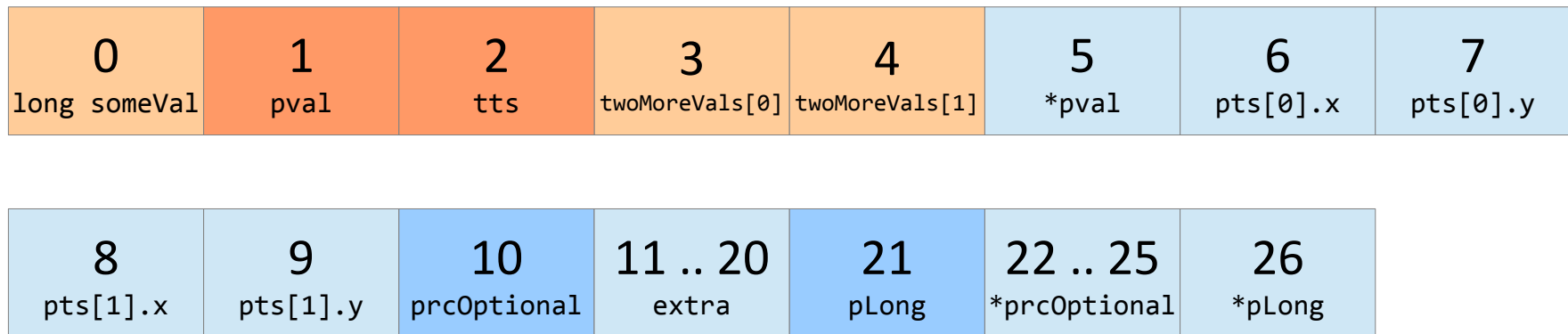
dll long ThreeTierFunc(long someVal, long * pval, TwoTierStruct * tts, long[2] twoMoreVals);

```

```

sizeDirect = 5
ptr = { 1:5, 2:6, 10:22, 21:26 }
data =

```



- copy value straight onto stack
- set pointer to value from ptr and copy onto stack
- ignore
- set pointer to offset from ptr

```

struct TwoTierStructWithString
{
    POINT[2] pts;
    RECT *   prcOptional;
    long[10] extra;
    string   s;
    buffer   b;
}

```

```

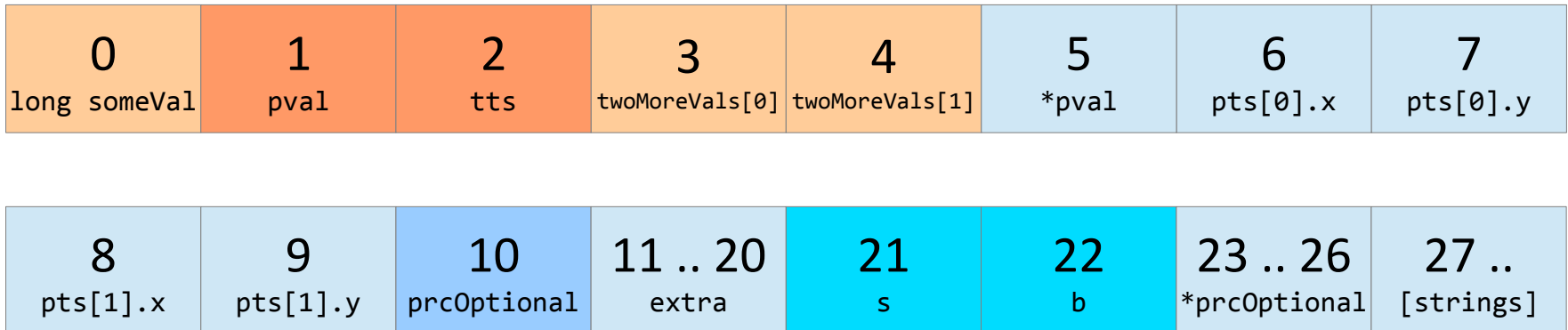
dll long ThreeTierFunc(long someVal, long * pval, TwoTierStruct * tts, long[2] twoMoreVals);






```

```

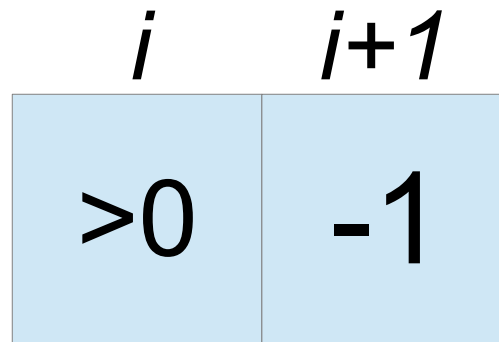
sizeDirect = 5
ptr = { 1:5, 2:6, 10:22, 21:-1, 22:-1 }
data =

```



	Java	C++
	marshall in direct data	copy value straight onto stack
	precomputing: precompute ptr index, marshalling: skip	set pointer to address given by ptr and copy onto stack
	marshall in indirect data	ignore
	precomputing: precompute ptr index, marshalling: skip	set pointer to offset from ptr
	determine appropriate ptr indices based on variable-length string data (can't precompute ptr before marshalling)	set pointer to offset from ptr

Meaning of the value -1 in ptrArray tuple



This value will never be set to a negative number.

Java	C++
<p>The exact index into the data array isn't known at the time the marshall plan is precomputed because the data pointed to is a variable length string.</p> <p>(-1 is replaced with the correct index at marshalling time).</p>	<p>The value placed in the pointer variable identified by <code>ptrArray[i]</code> (ie the pointer should not be set to point to <code>&data[ptrArray[i+1]]</code>).</p> <p>This can happen when a NULL pointer is passed, or when an integer resource is passed.</p>