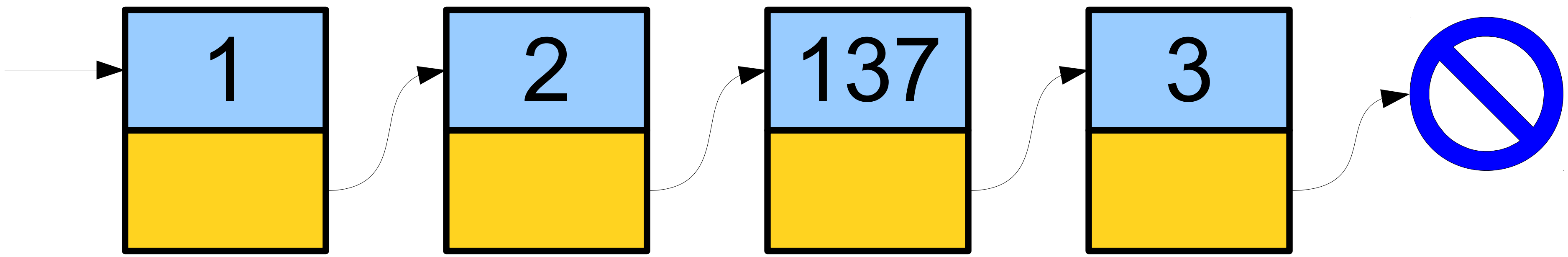
Linked Lists

Part Two

Recap from Last Time

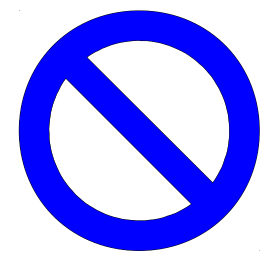
Linked Lists

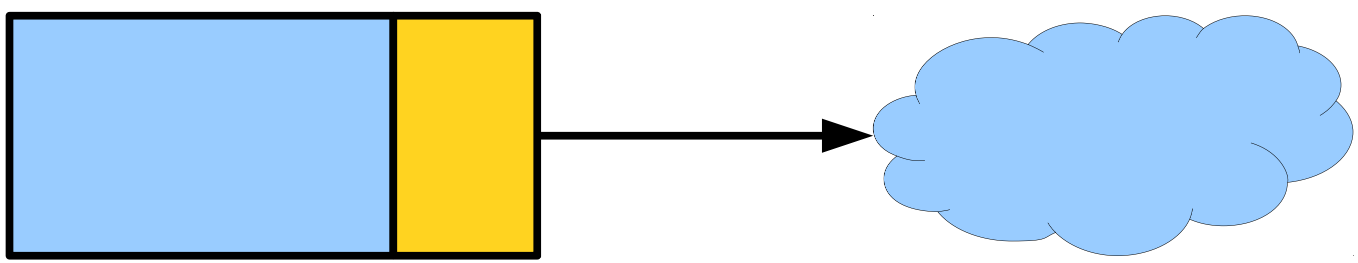
- A ***linked list*** is a data structure for storing a sequence of elements.
- Each element is stored separately from the rest.
- The elements are then chained together into a sequence.
- The end of the list is marked with some special indicator.



A Linked List is Either…

…an empty list, represented by `nullptr`, or…





a single linked list cell that points…

… at another linked list.

New Stuff!

A Problem

Stack Overflows

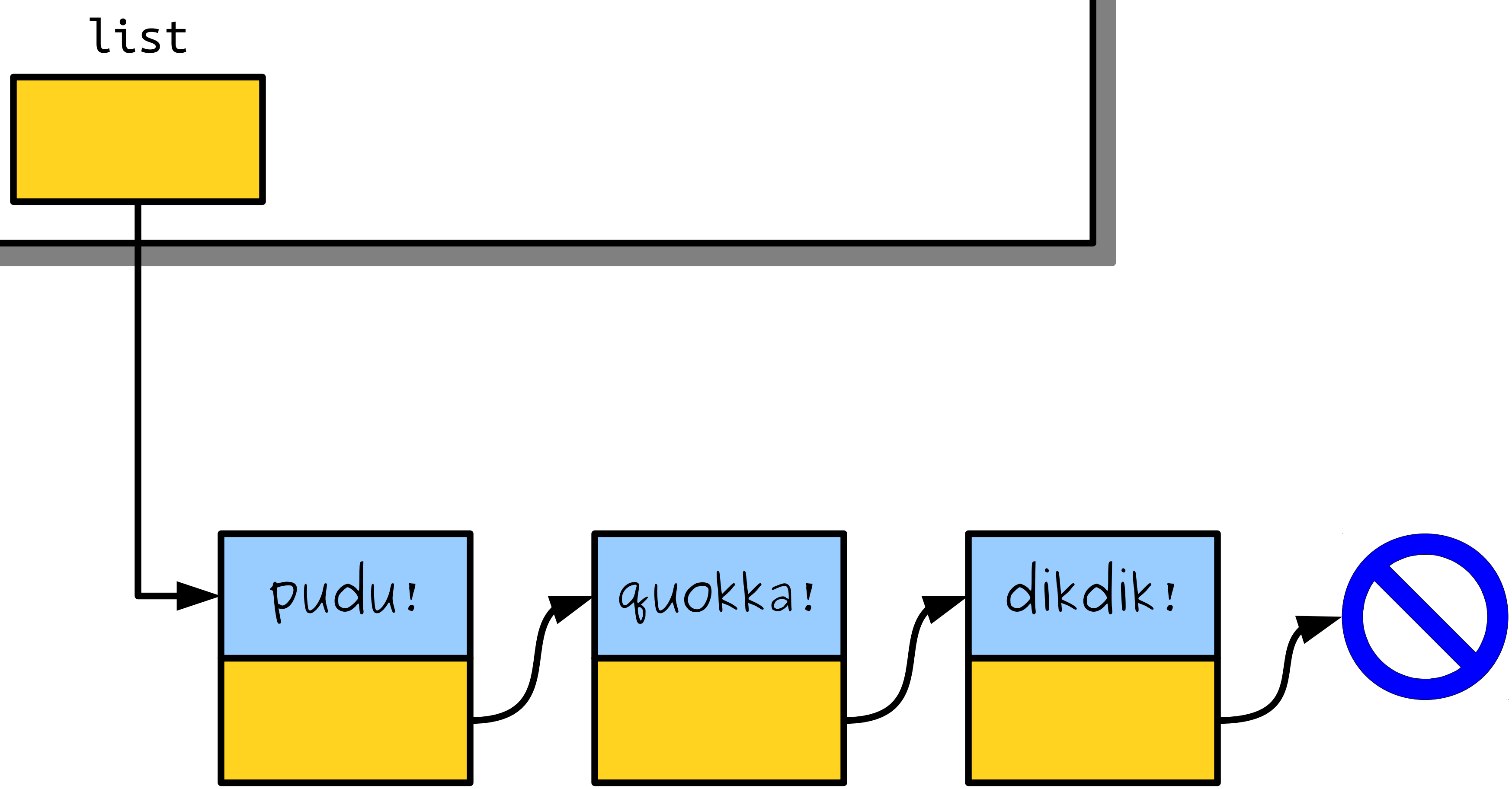
- Recursive code can result in stack overflows in cases where the recursion requires too many stack frames to finish a calculation.
- This means that recursion might not be the best strategy for manipulating linked lists, especially if those lists get really long.
- What should we do instead?

Processing Lists Iteratively

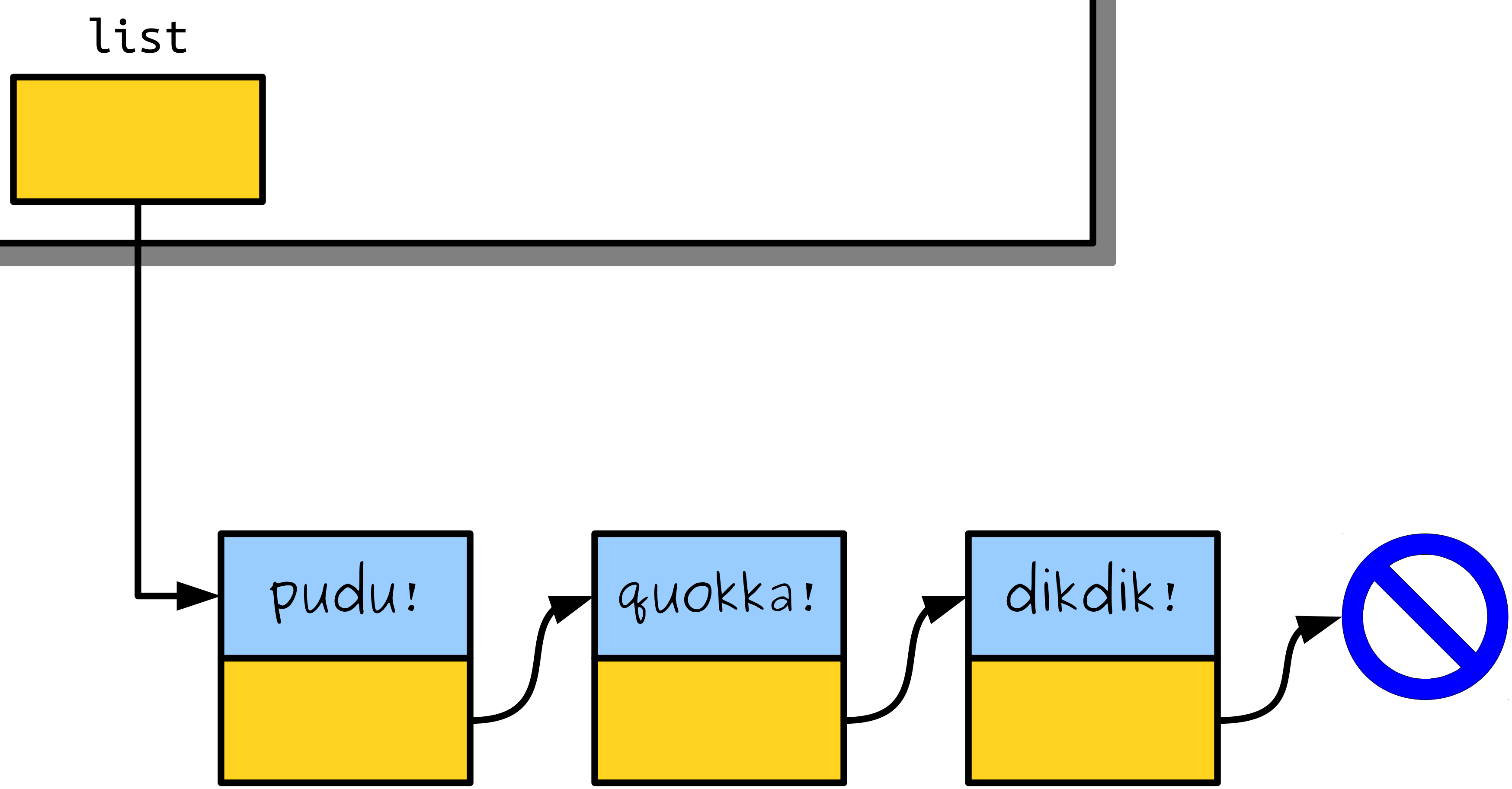
```
int main() {
   Cell* list = readList();
   cout << lengthOf(list) << endl;

   /* … other listy things. … */
}
```

```
int main() {
    Cell* list = readList();
    cout << lengthOf(list) << endl;

    /* … other listy things. … */
}
```

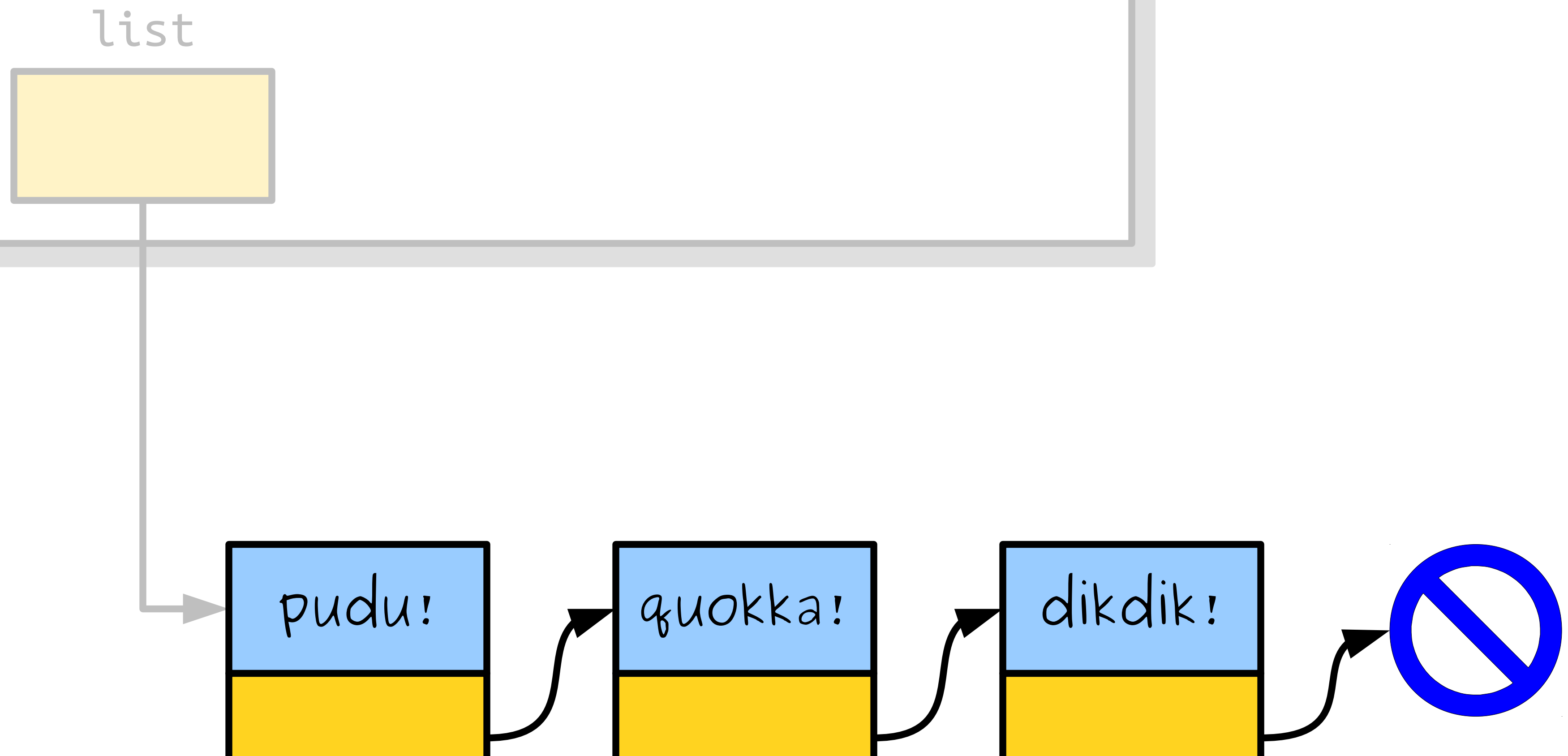
```
int main() {
    Cell* list = readList();
    cout << lengthOf(list) << endl;

    /* … other listy things. … */
}
```



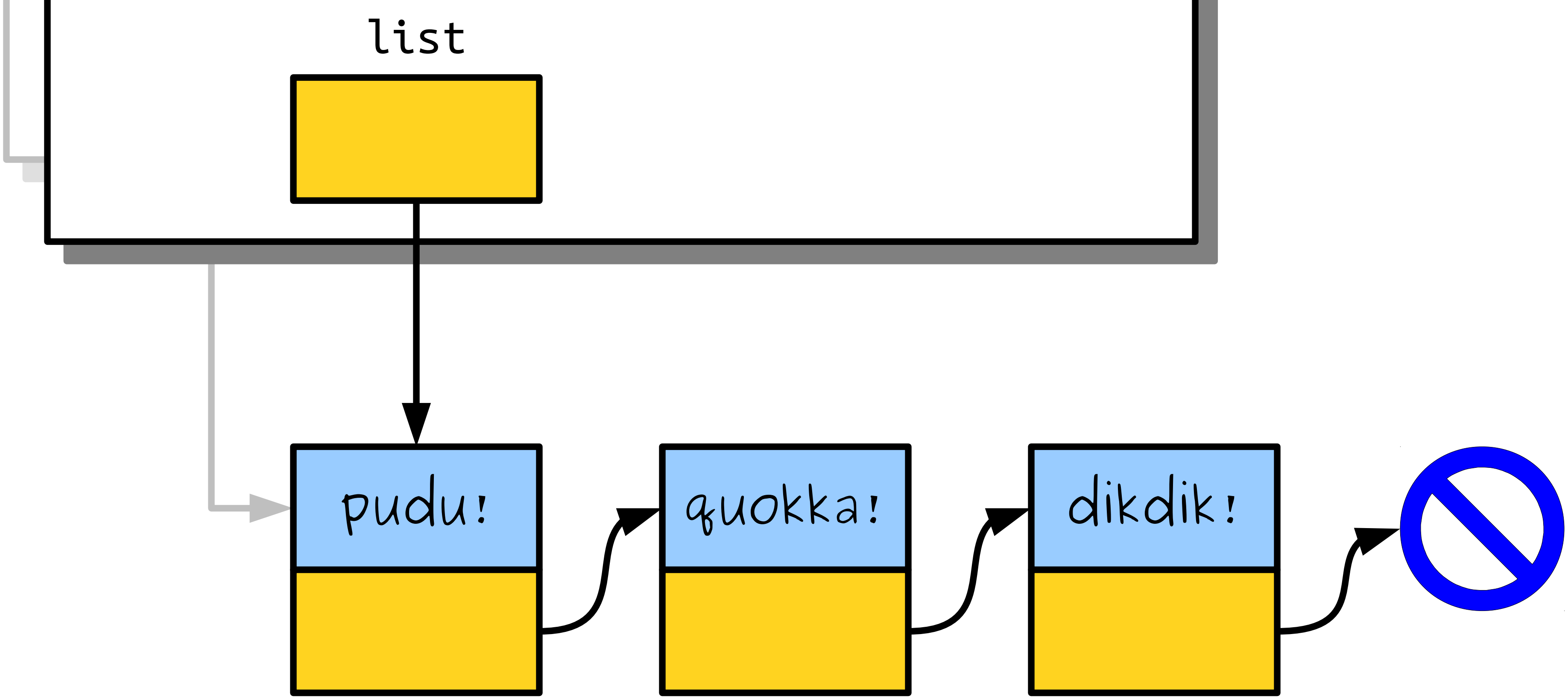
```
int main() {
    Cell* list = readList();
    cout << lengthOf(list) << endl;

    /* … other listy things. … */
}
```



```
int main() {
    Cell* list = readList();
    cout << lengthOf(list) << endl;

    /* … other listy things. … */
}
```

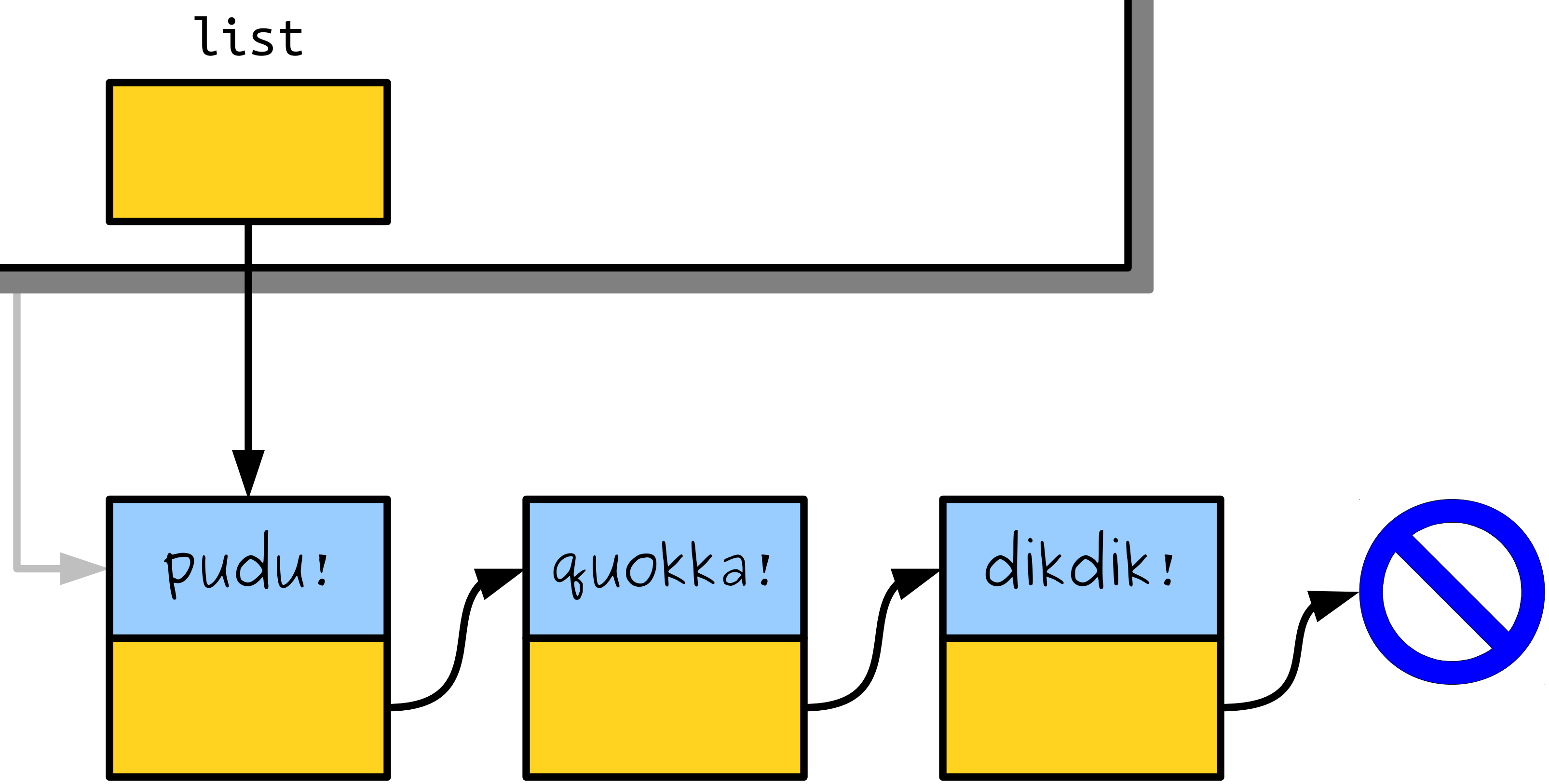


```
int lengthOf(Cell* list) {
    int result = 0;
    while (list != nullptr) {
        result++;
        list = list->next;
    }
    return result;
}
```



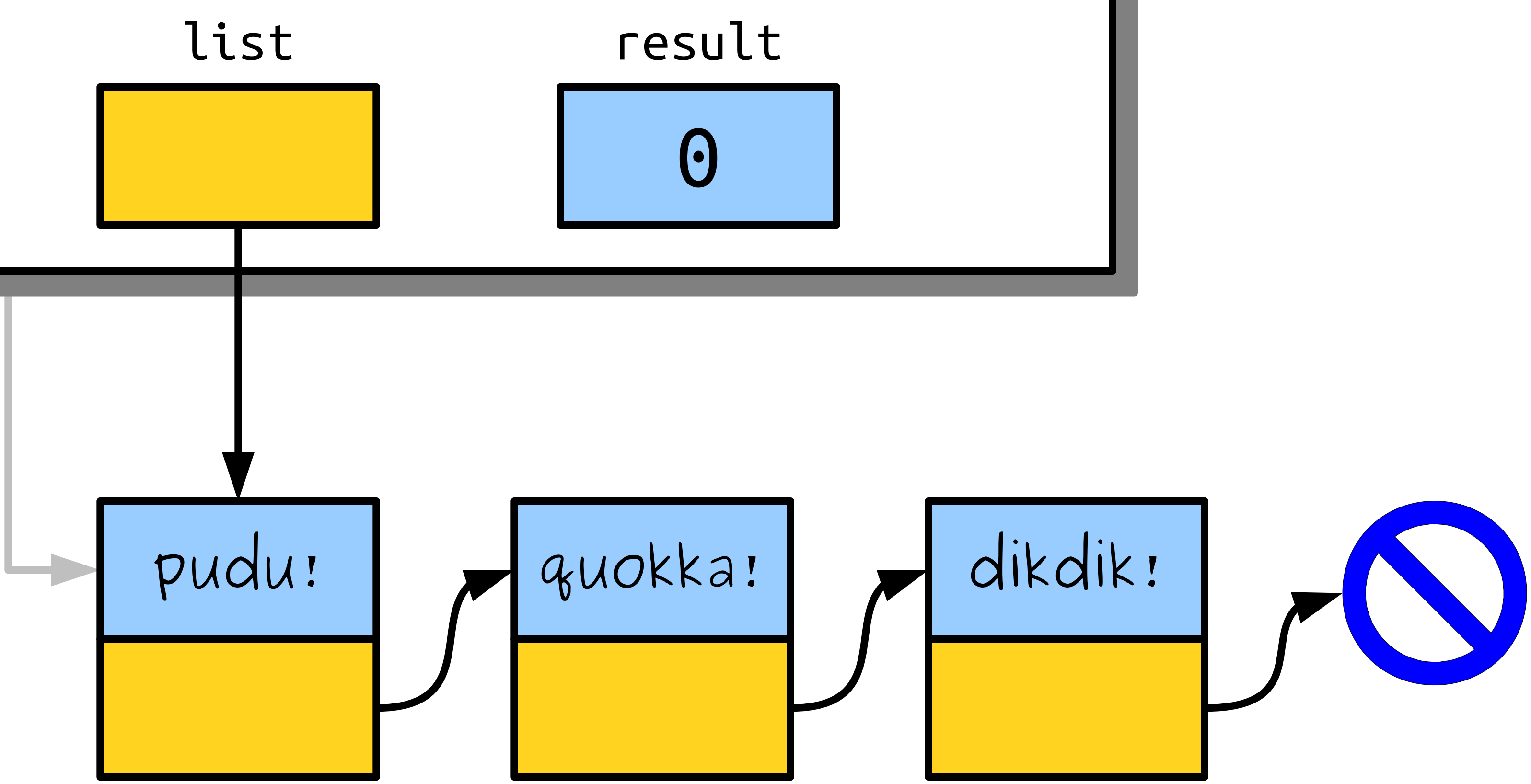
```
int main() {
```

```
int lengthOf(Cell* list) {
    int result = 0;
    while (list != nullptr) {
        result++;
        list = list->next;
    }
    return result;
}
```



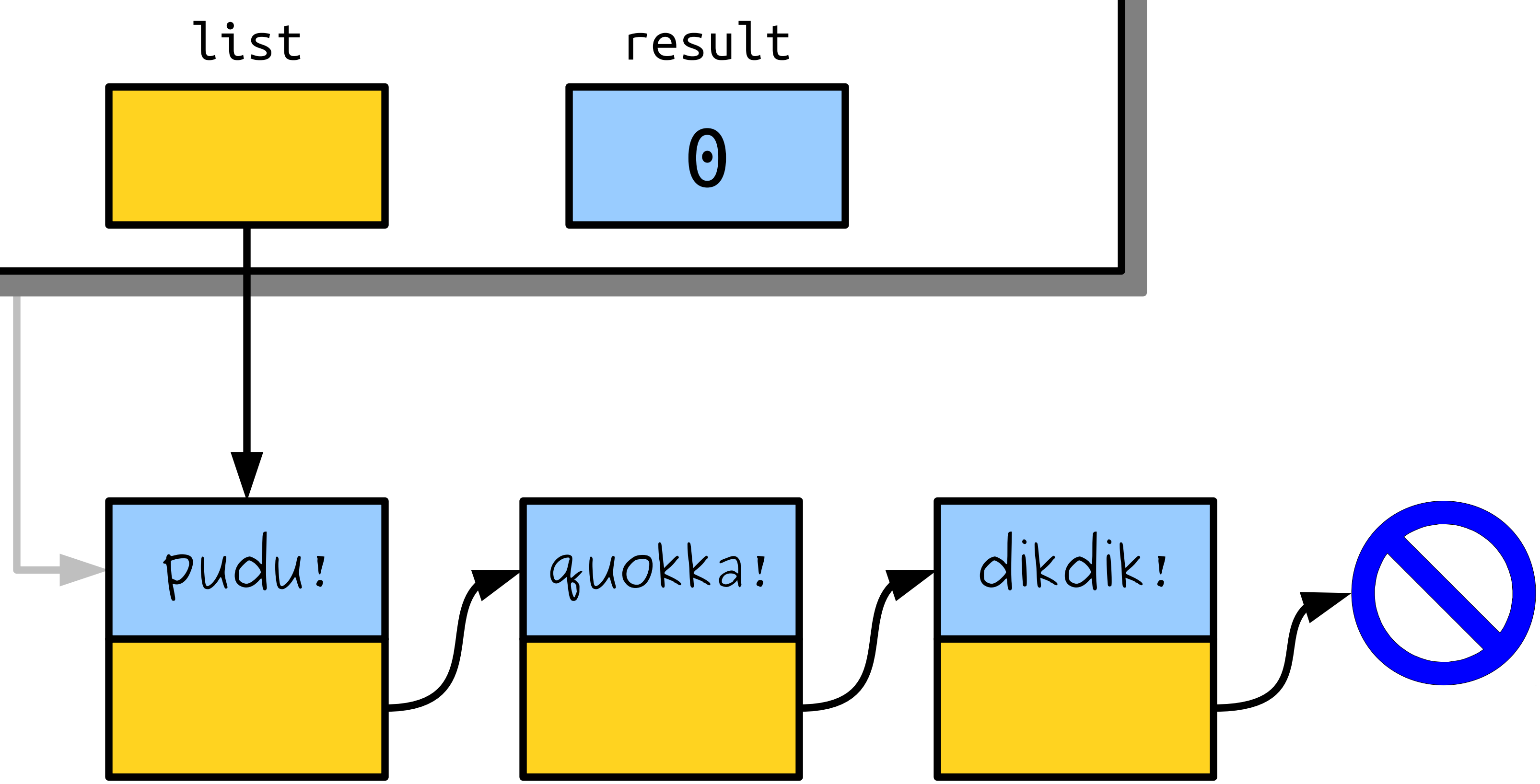
```
int main() {
```

```
int lengthOf(Cell* list) {
   int result = 0;
   while (list != nullptr) {
      result++;
      list = list->next;
   }
   return result;
}
```

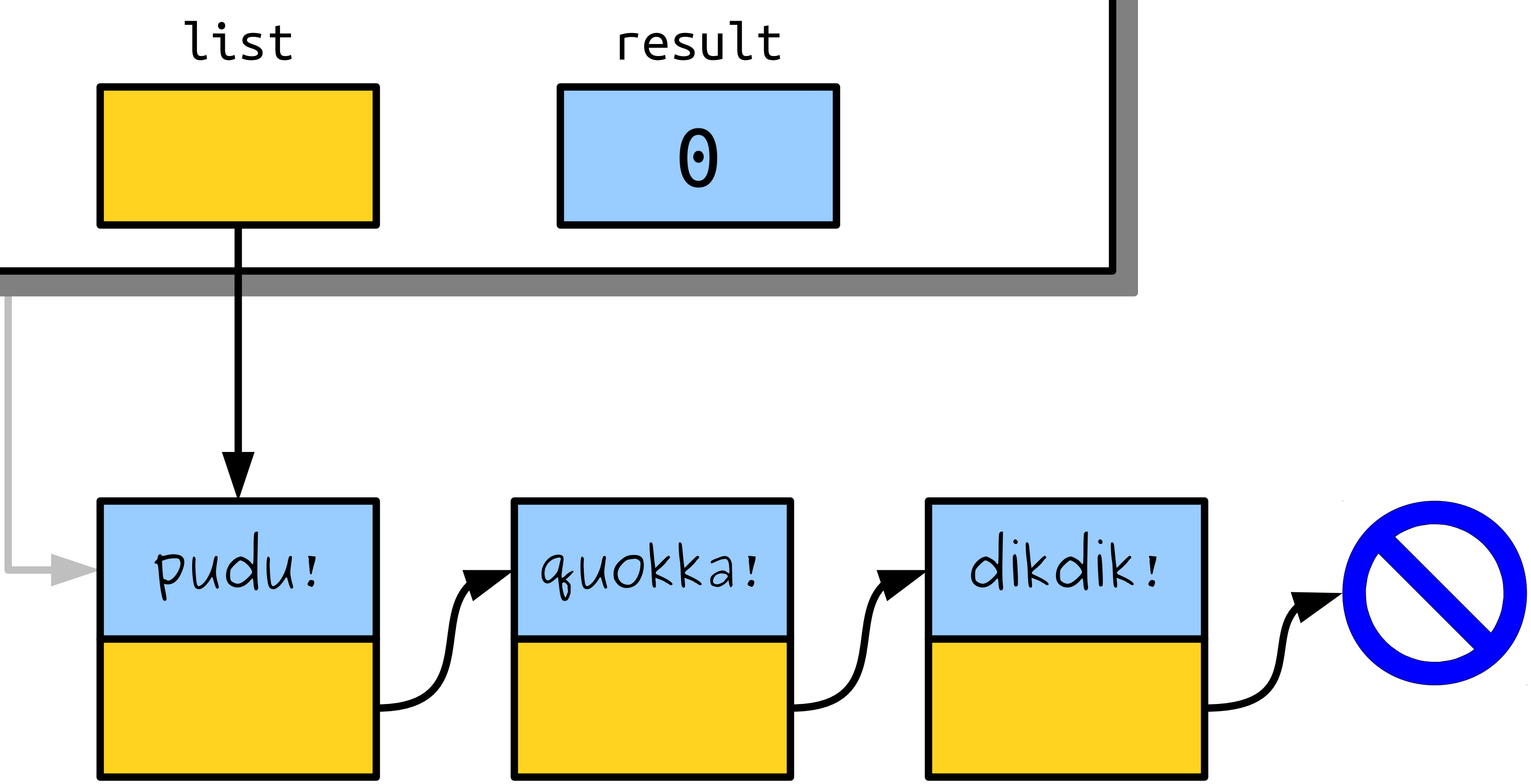


```
int main() {
```

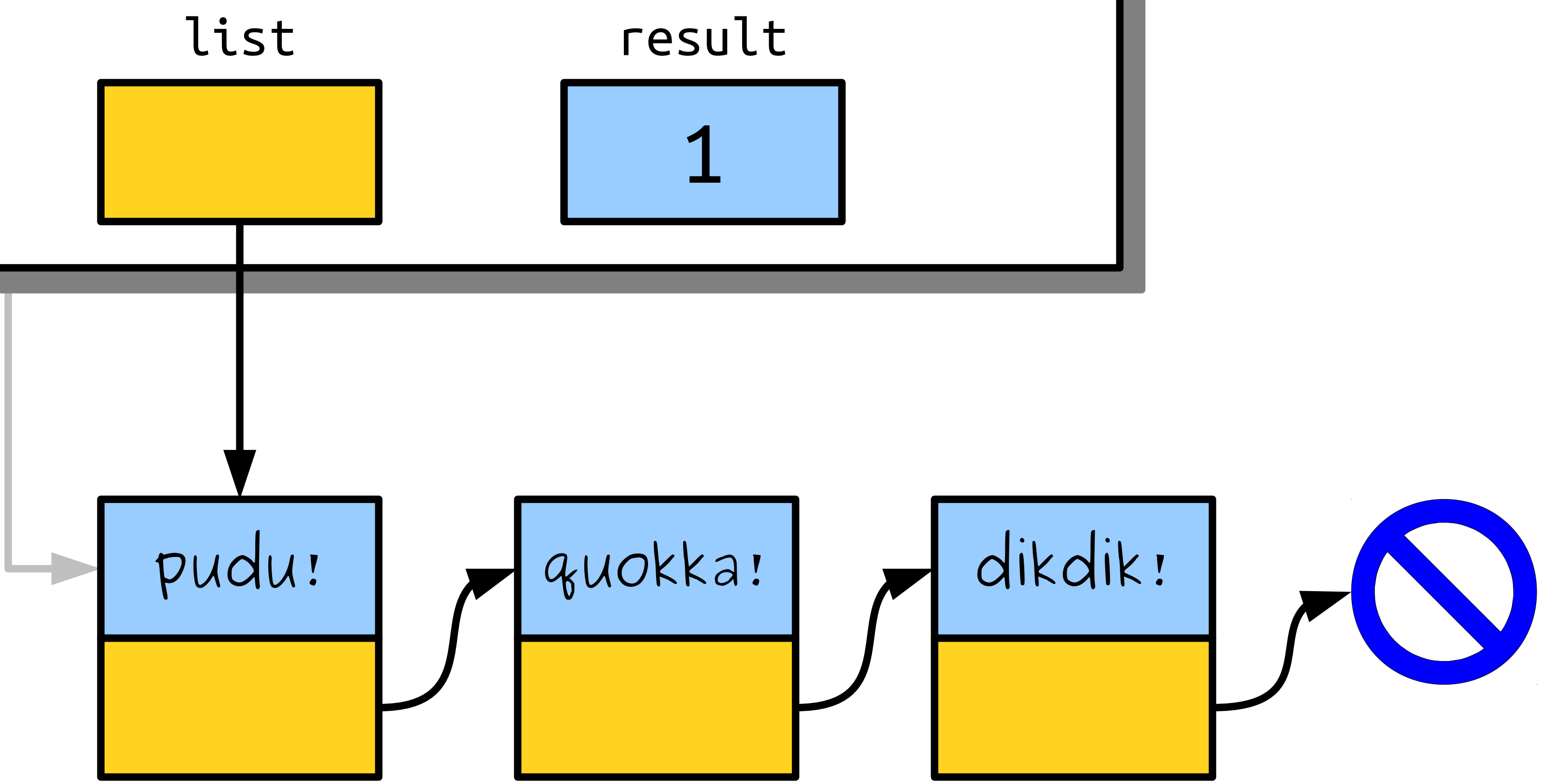
```
int lengthOf(Cell* list) {
    int result = 0;
    while (list != nullptr) {
        result++;
        list = list->next;
    }
    return result;
}
```

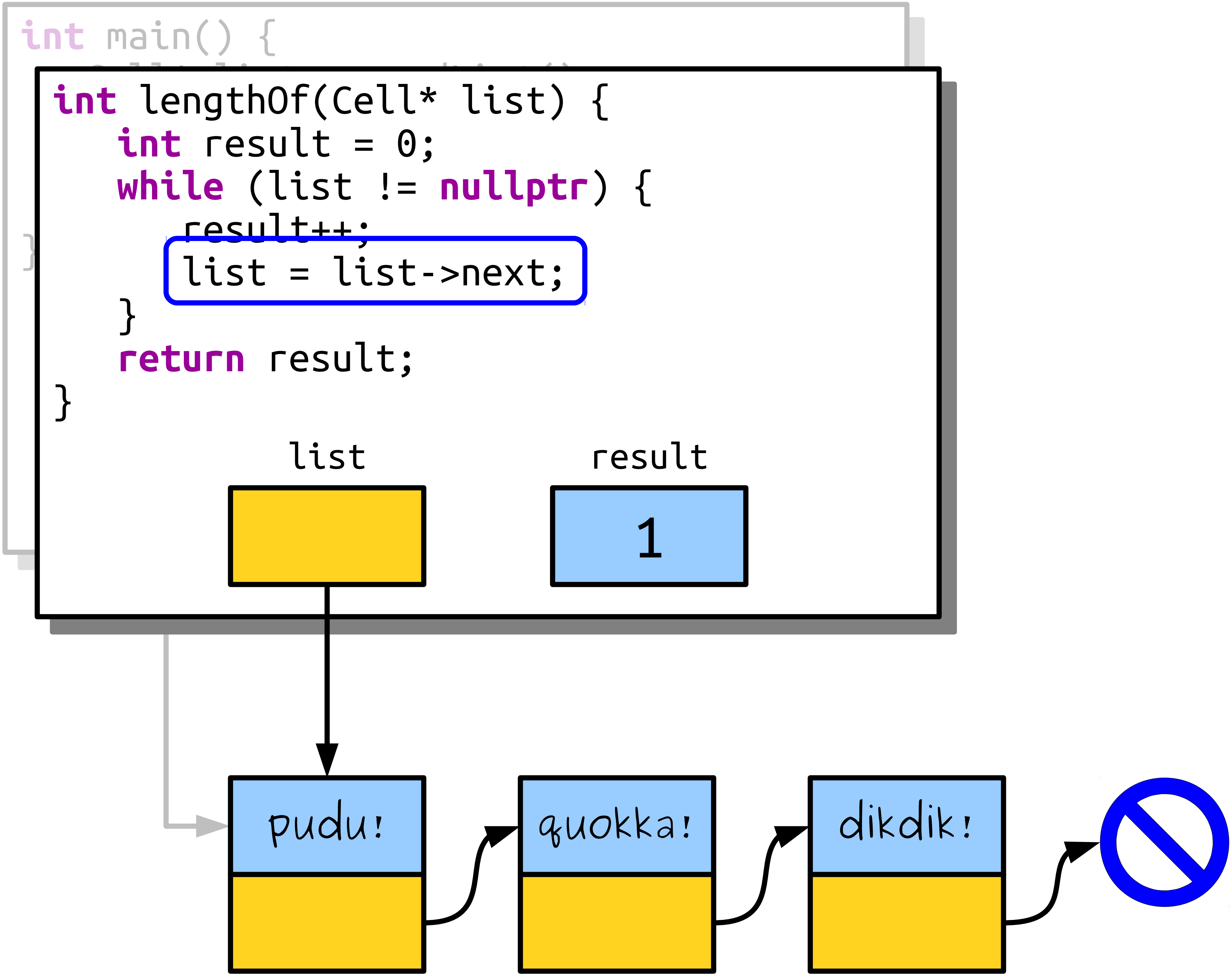


```
int lengthOf(Cell* list) {
    int result = 0;
    while (list != nullptr) {
        result++;
        list = list->next;
    }
    return result;
}
```



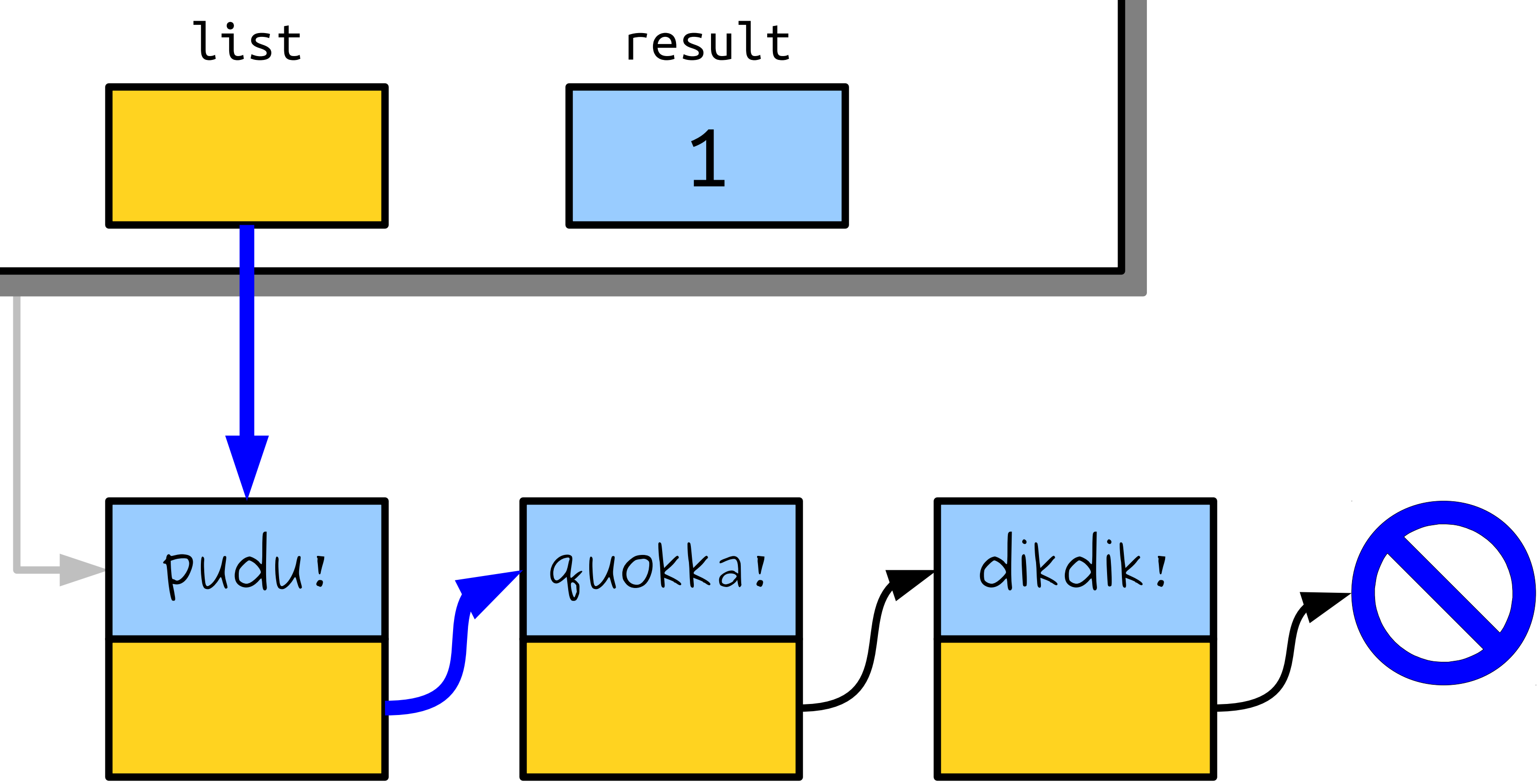
```
int lengthOf(Cell* list) {
    int result = 0;
    while (list != nullptr) {
        result++;
        list = list->next;
    }
    return result;
}
```



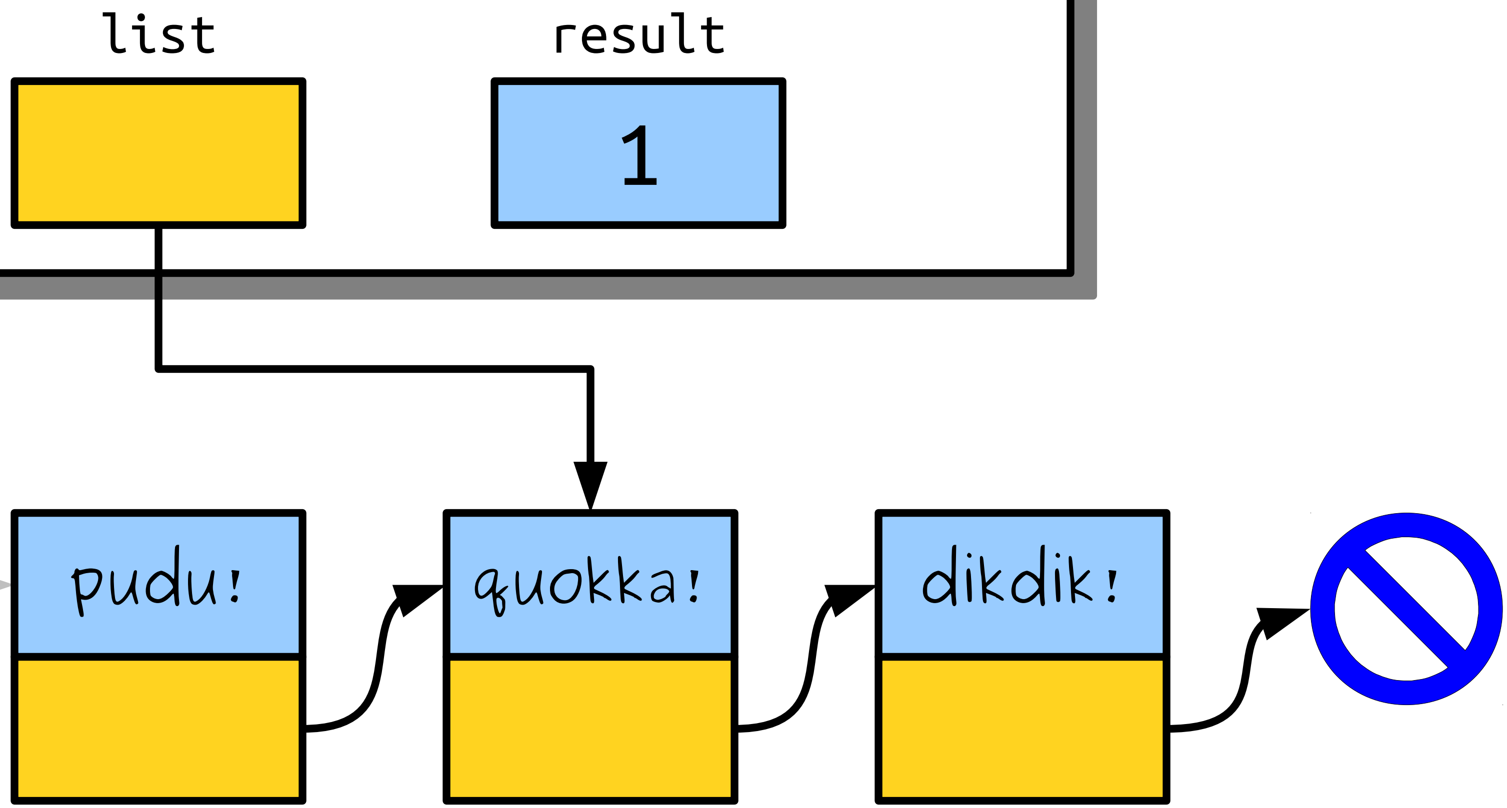

int main() {
int lengthOf(Cell* list) {
int result = 0;
while (list != nullptr) {
result++;
list = list->next;
}
return result;
}
list
result
1
pudu!
quokka!
dikdik!

```
int main() {
```

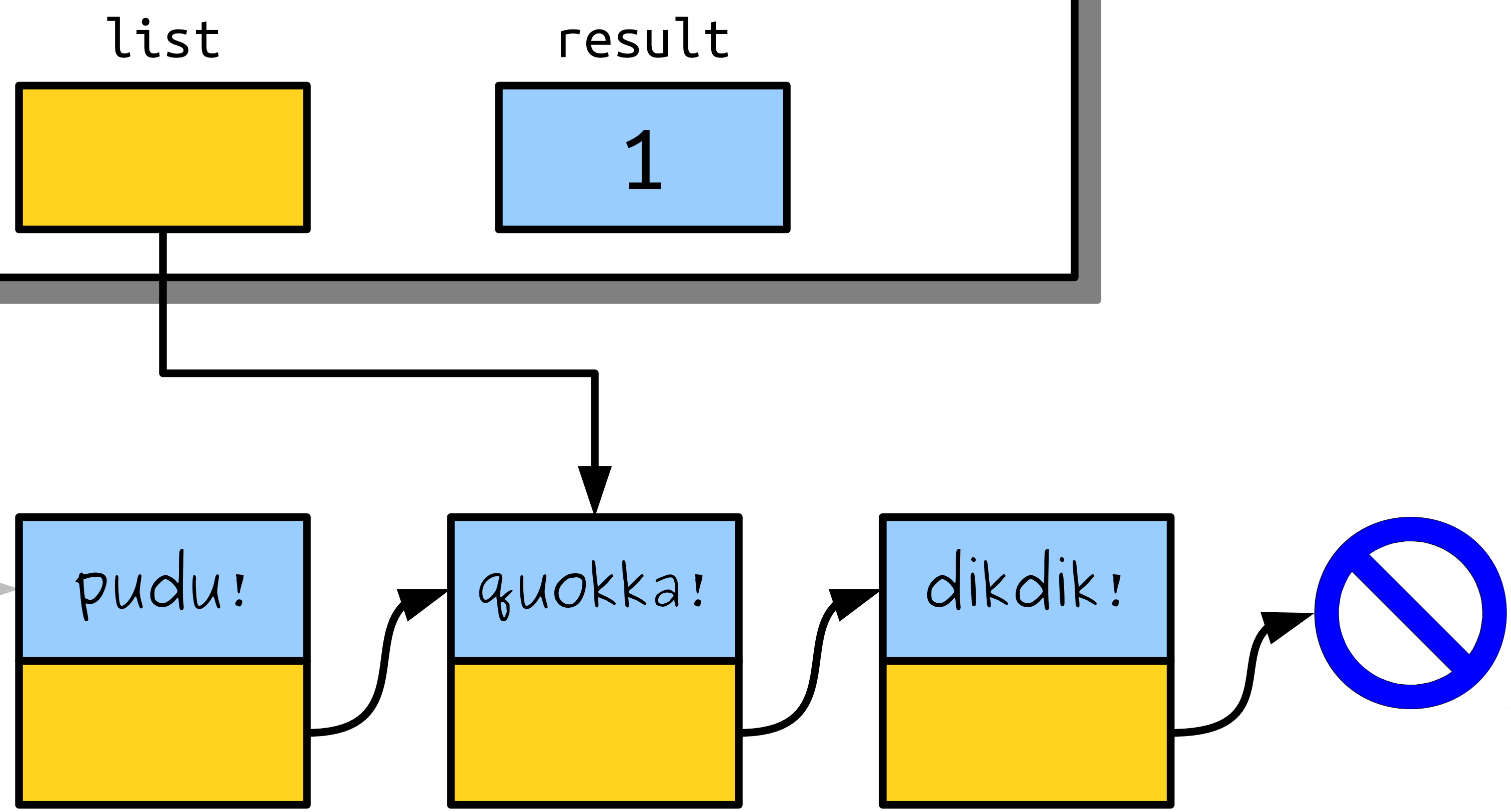
```
int lengthOf(Cell* list) {
    int result = 0;
    while (list != nullptr) {
        result++;
        list = list->next;
    }
    return result;
}
```



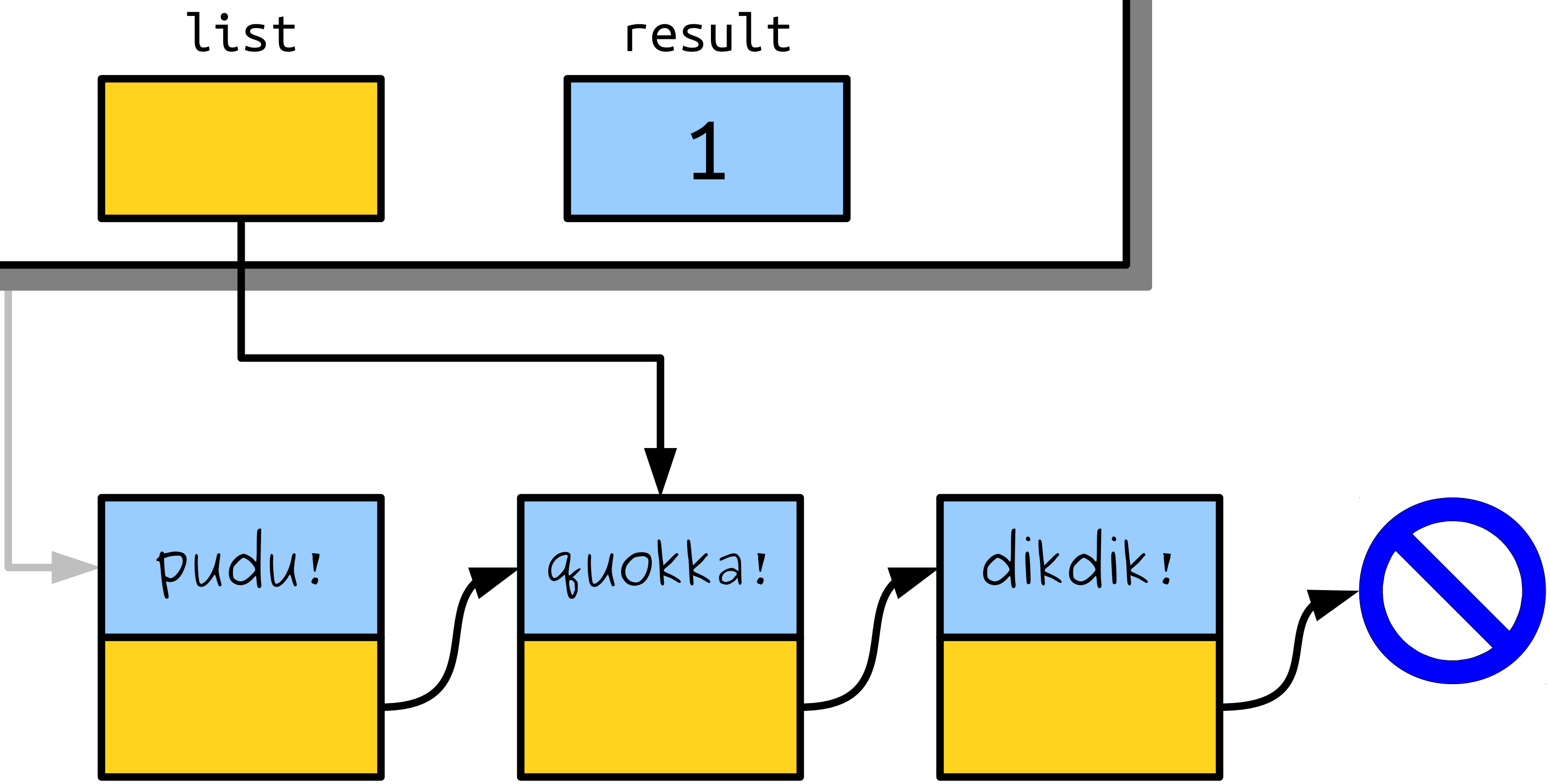
```
int lengthOf(Cell* list) {
    int result = 0;
    while (list != nullptr) {
        result++;
        list = list->next;
    }
    return result;
}
```



```
int lengthOf(Cell* list) {
    int result = 0;
    while (list != nullptr) {
        result++;
        list = list->next;
    }
    return result;
}
```

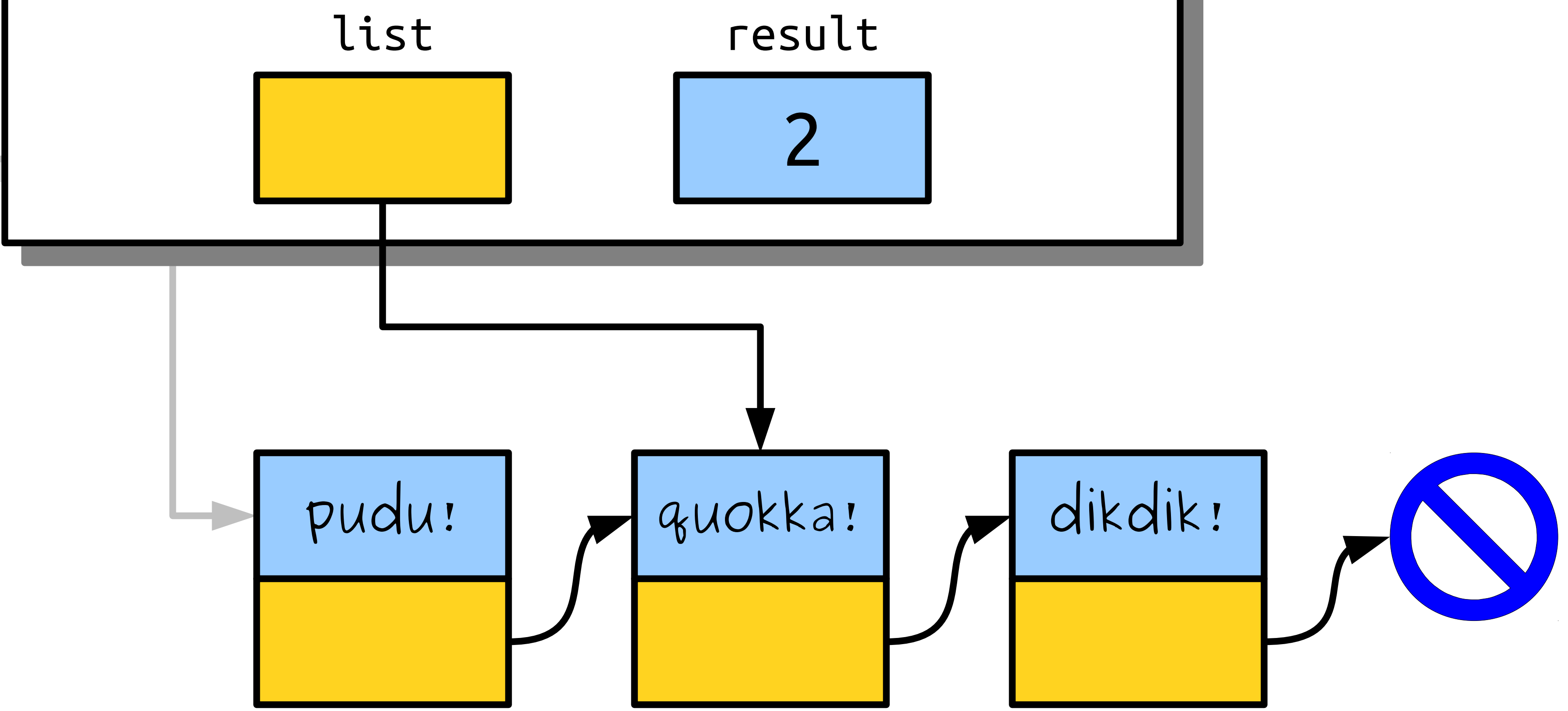


```
int lengthOf(Cell* list) {
    int result = 0;
    while (list != nullptr) {
        result++;
        list = list->next;
    }
    return result;
}
```

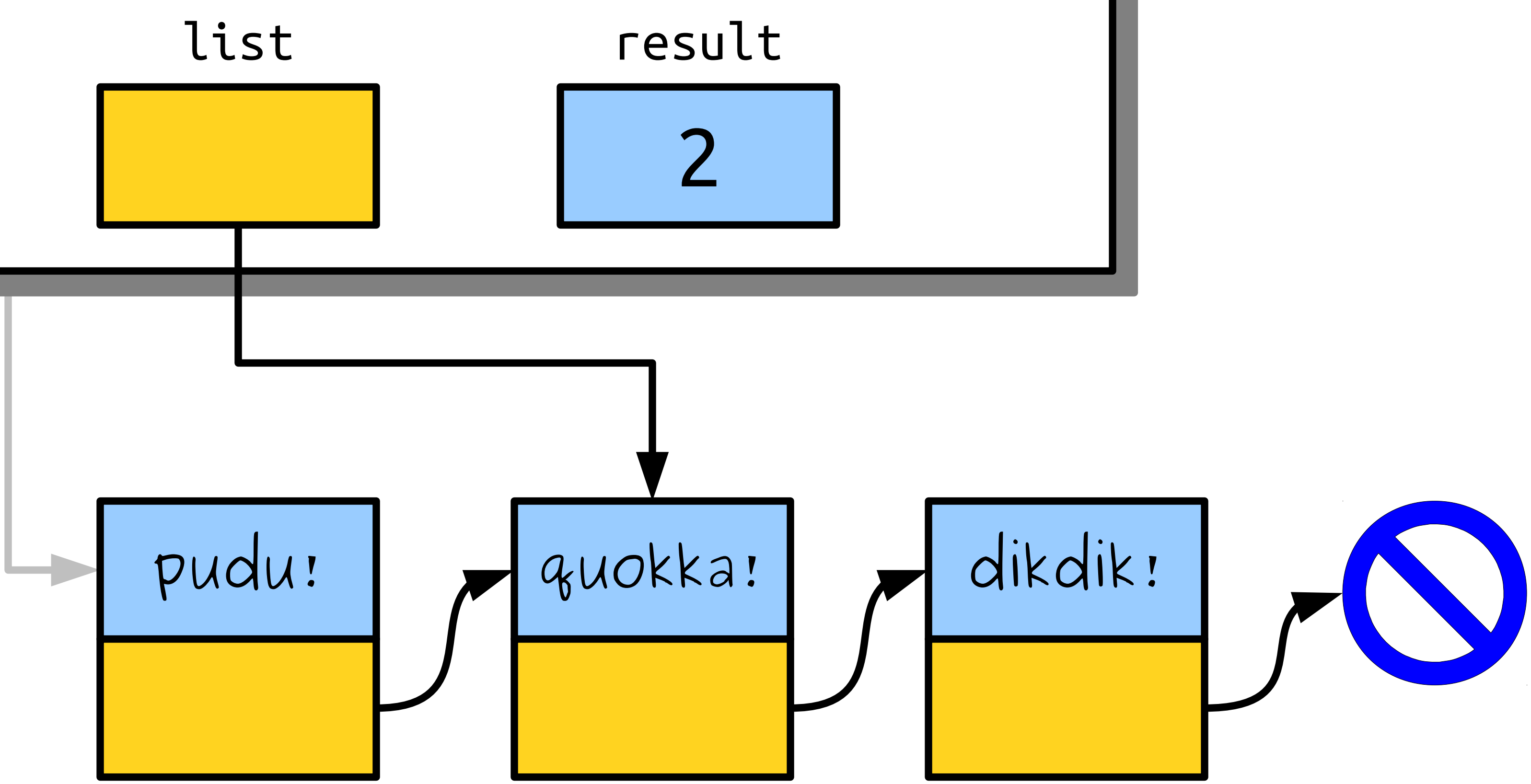


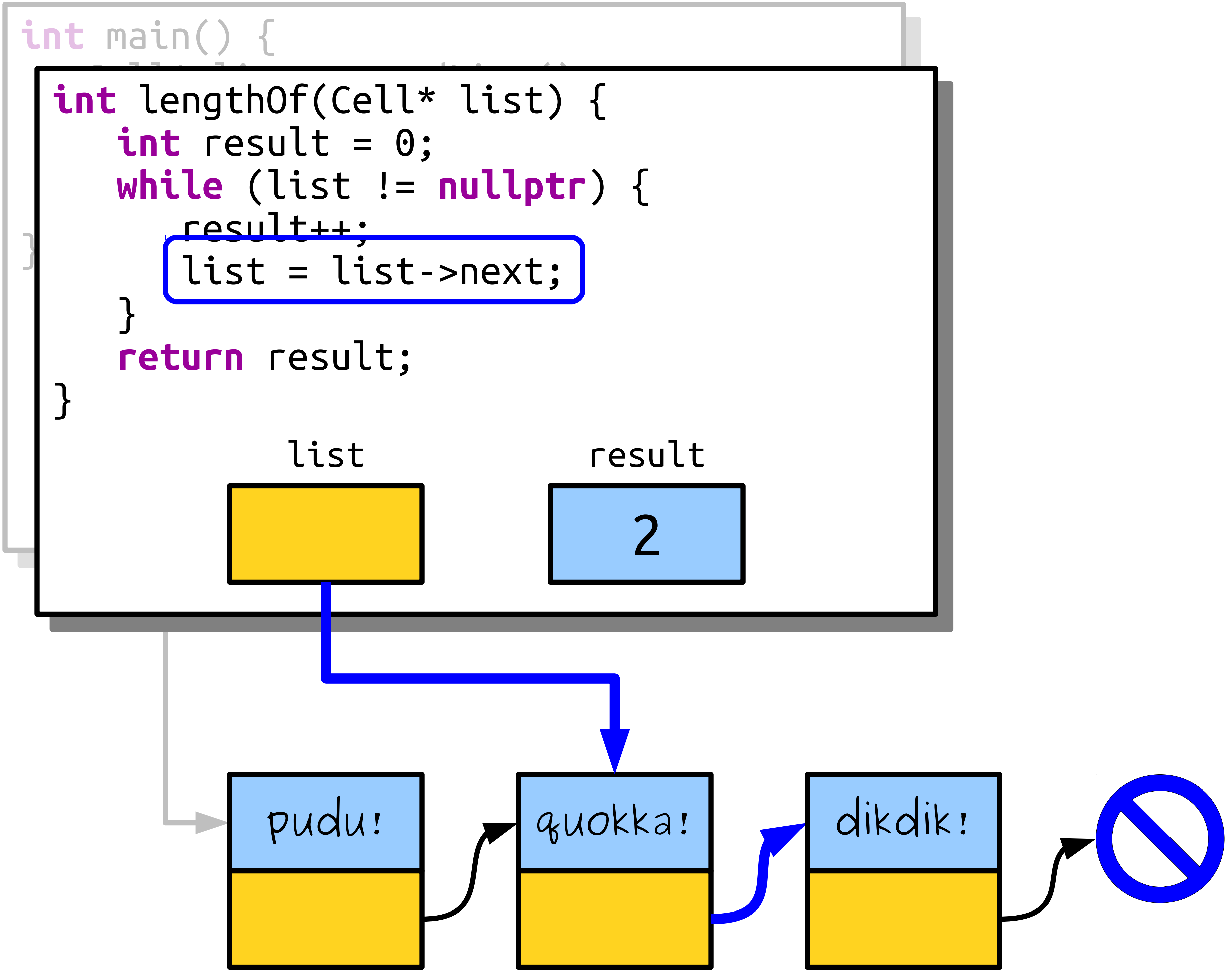
```
int main() {
```

```
int lengthOf(Cell* list) {
    int result = 0;
    while (list != nullptr) {
        result++;
        list = list->next;
    }
    return result;
}
```

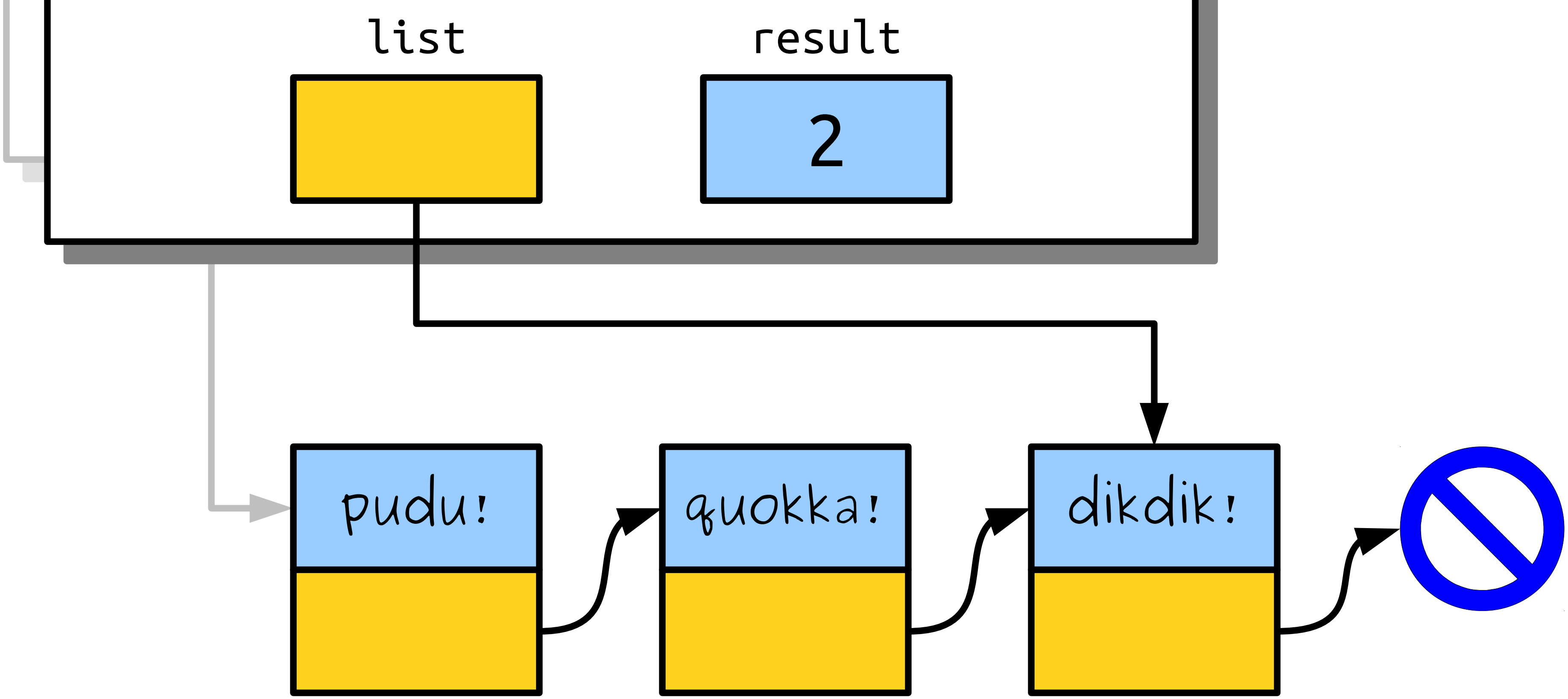


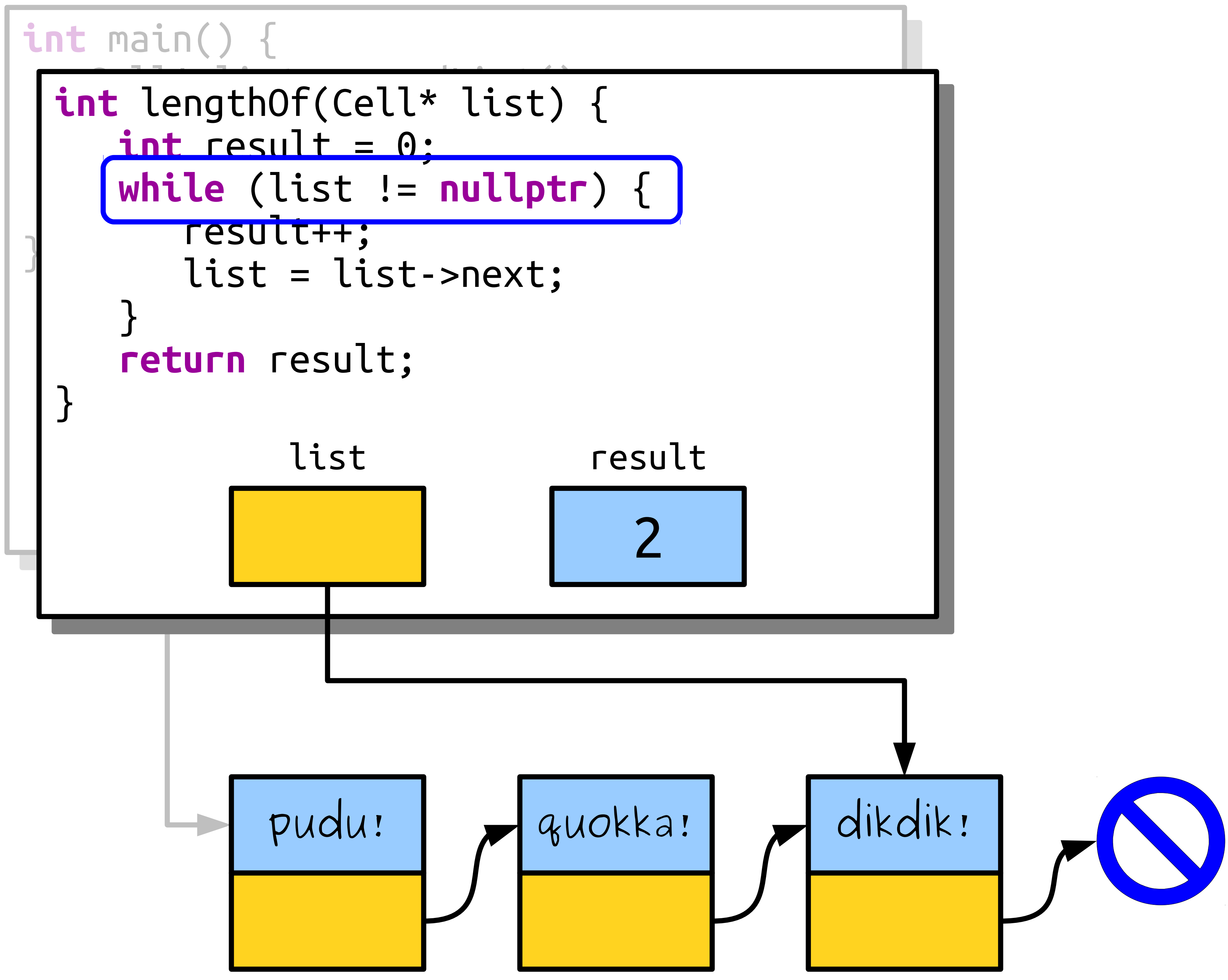
```
int lengthOf(Cell* list) {
    int result = 0;
    while (list != nullptr) {
        result++;
        list = list->next;
    }
    return result;
}
```

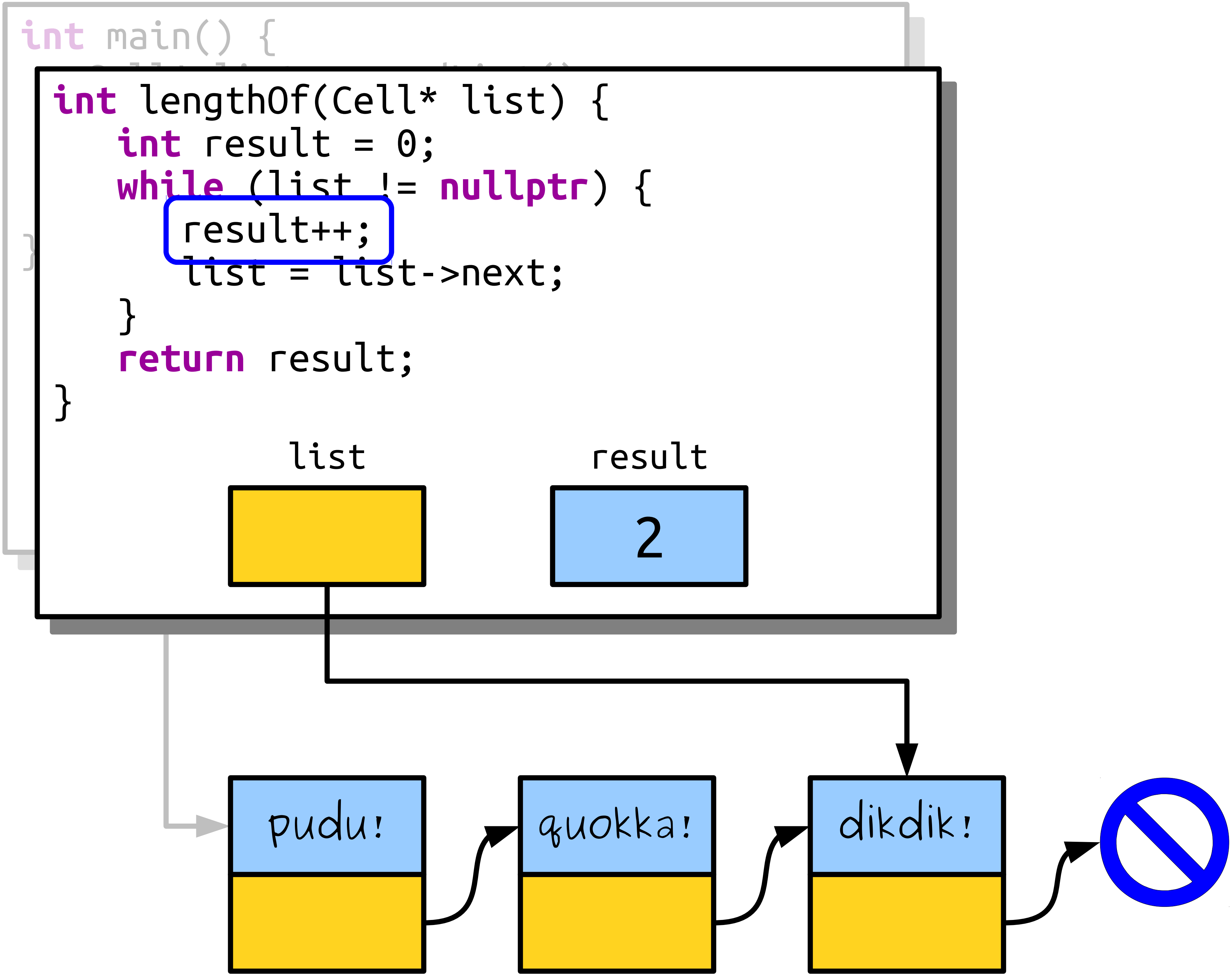


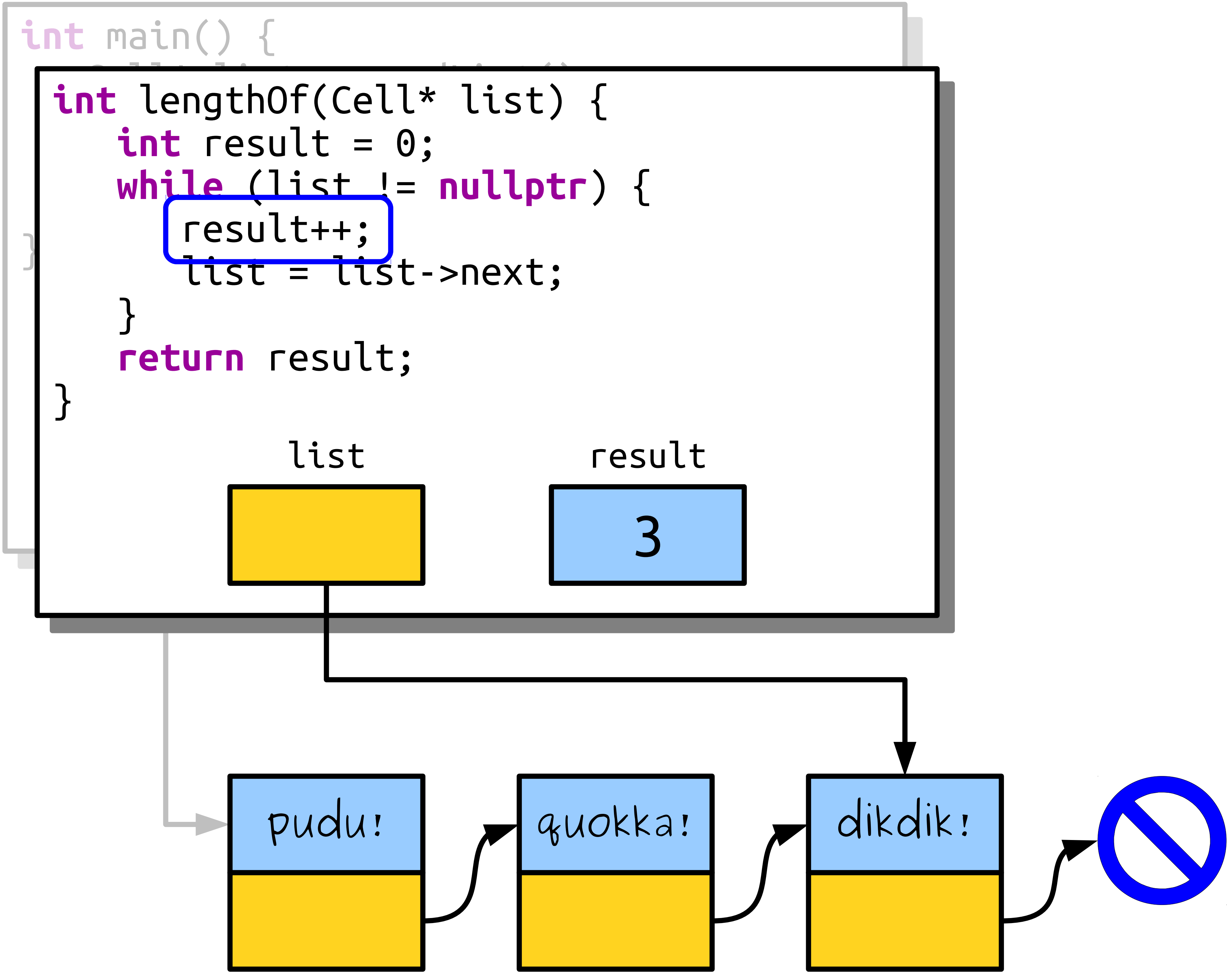

int main() {
int lengthOf(Cell* list) {
int result = 0;
while (list != nullptr) {
result++;
list = list->next;
}
return result;
}
list
result
2
pudu!
quokka!
dikdik!

```
int lengthOf(Cell* list) {
    int result = 0;
    while (list != nullptr) {
        result++;
        list = list->next;
    }
    return result;
}
```

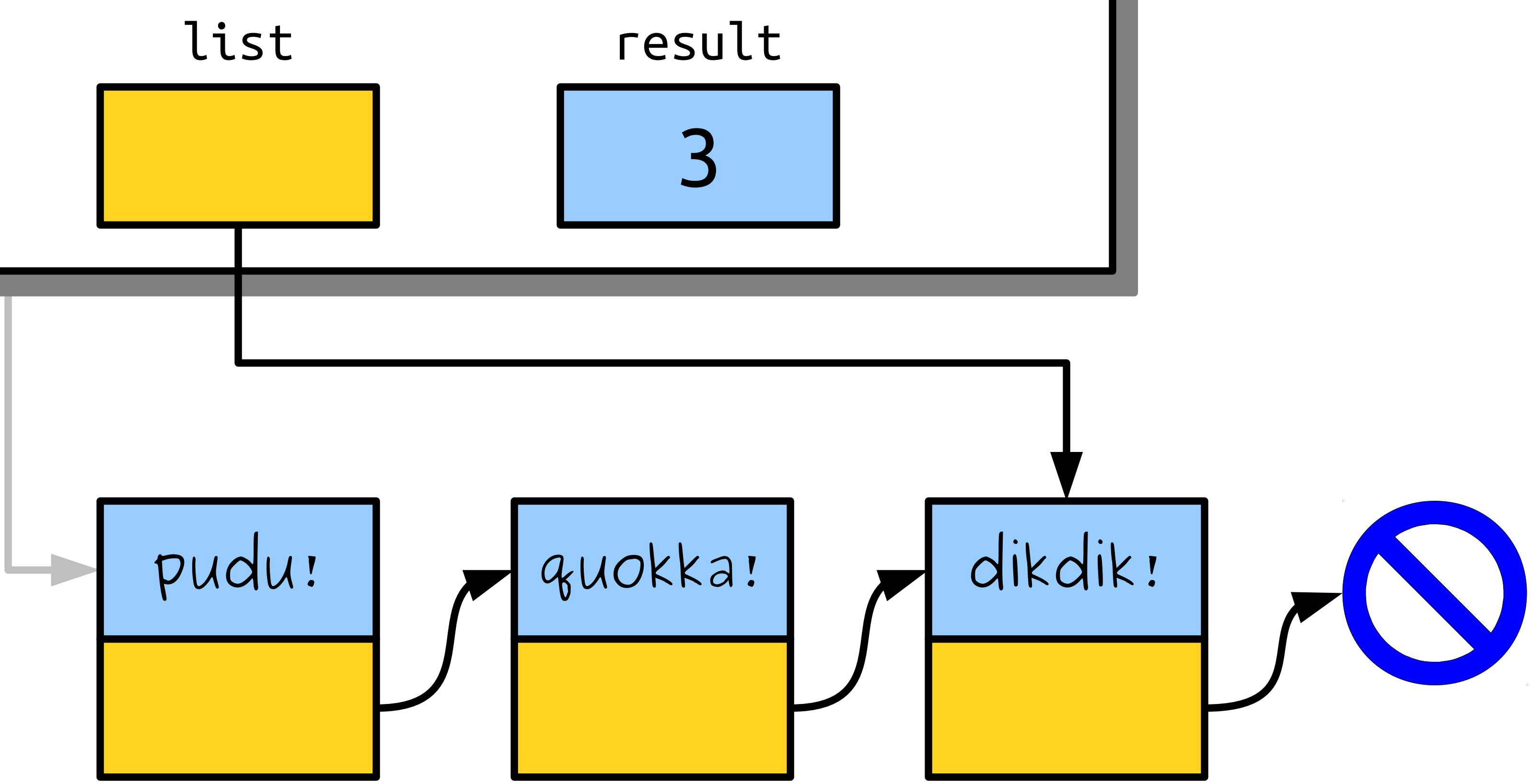



int main() {
int lengthOf(Cell* list) {
int result = 0;
while (list != nullptr) {
result++;
list = list->next;
}
return result;
}
list
result
2
pudu!
quokka!
dikdik!


int main() {
int lengthOf(Cell* list) {
int result = 0;
while (list != nullptr) {
result++;
list = list->next;
}
return result;
}
list
result
2
pudu!
quokka!
dikdik!

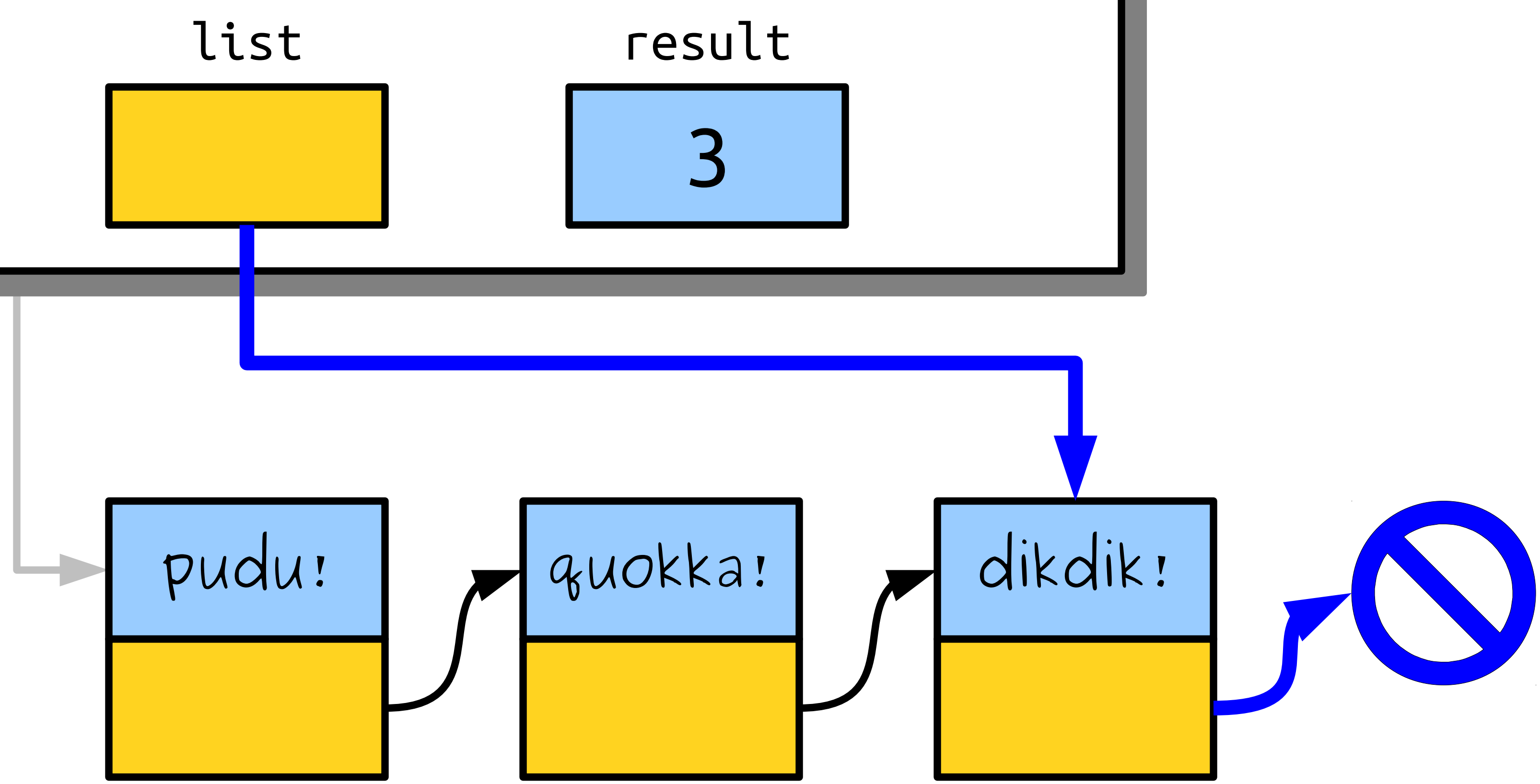

int main() {
int lengthOf(Cell* list) {
int result = 0;
while (list != nullptr) {
result++;
list = list->next;
}
return result;
}
list
result
3
pudu!
quokka!
dikdik!

```
int lengthOf(Cell* list) {
    int result = 0;
    while (list != nullptr) {
        result++;
        list = list->next;
    }
    return result;
}
```

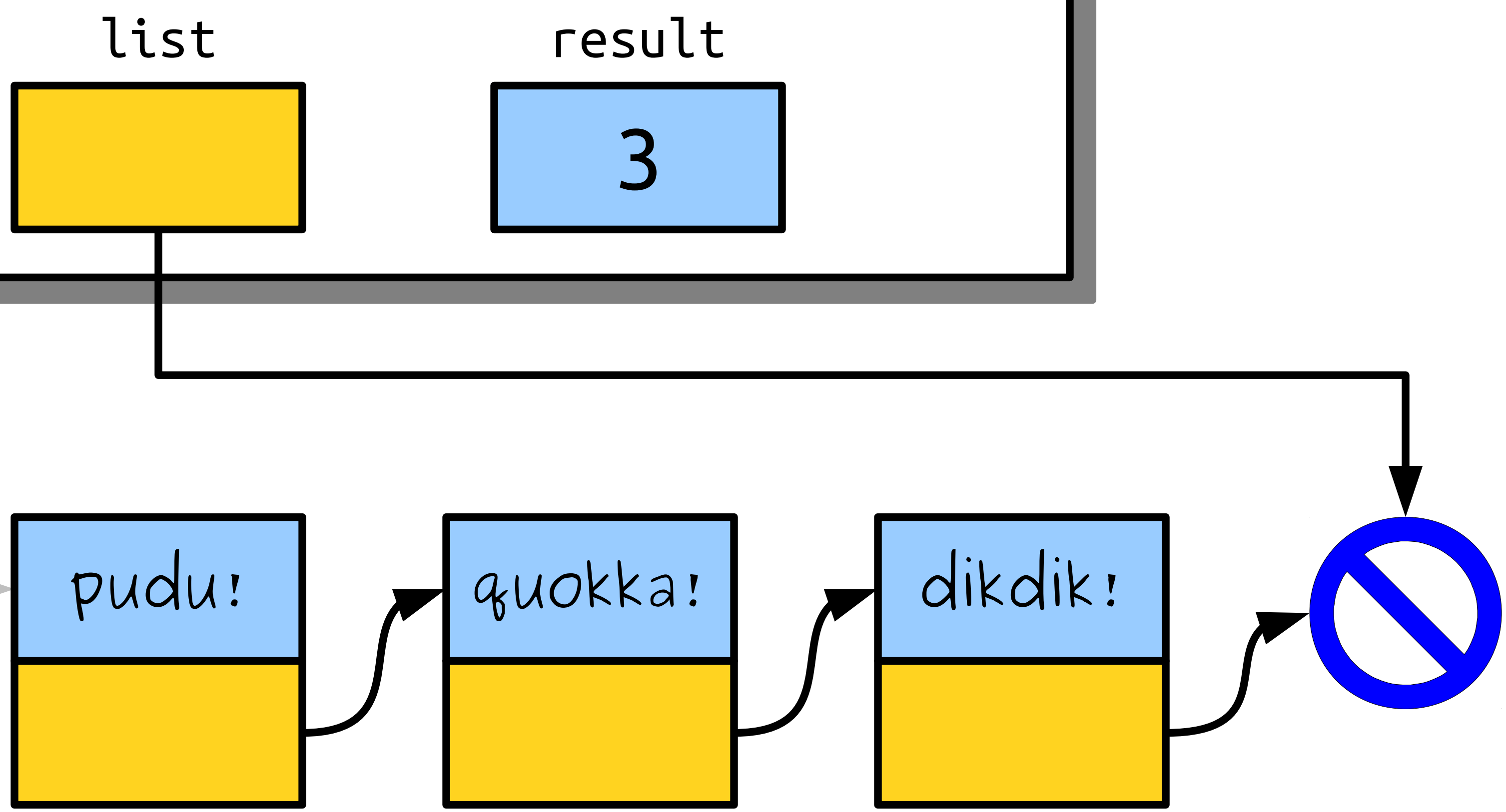


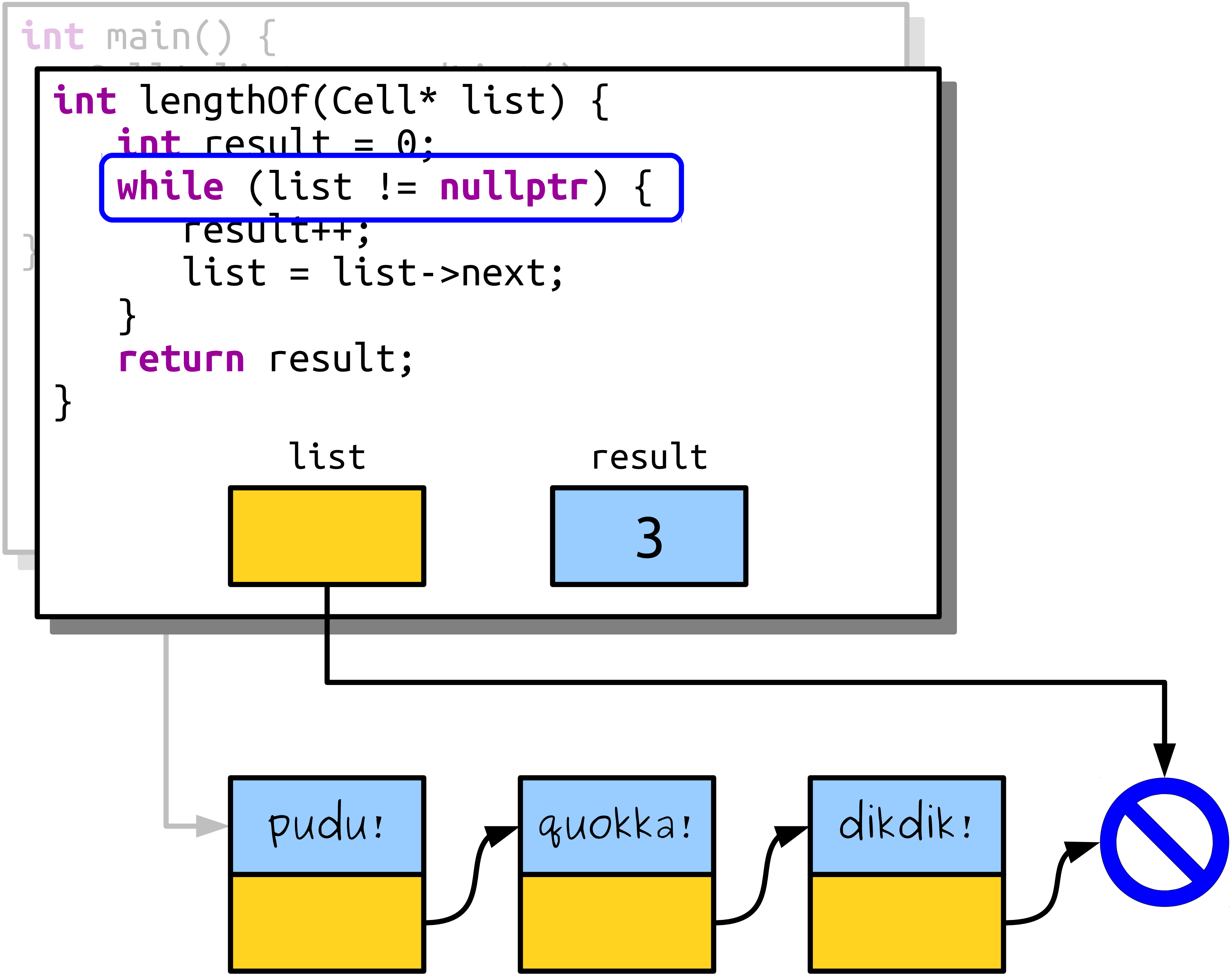
```
int main() {
```

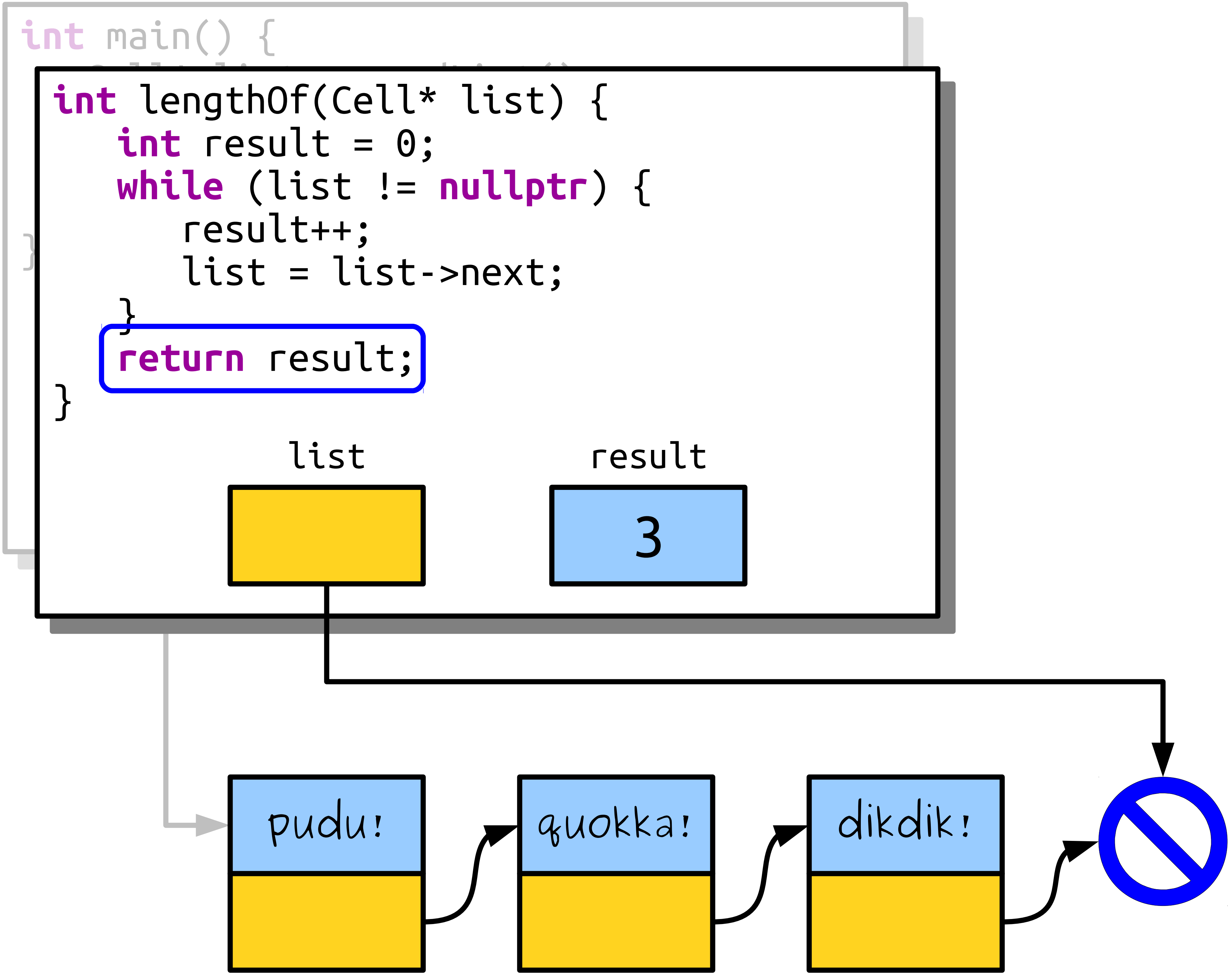
```
int lengthOf(Cell* list) {
    int result = 0;
    while (list != nullptr) {
        result++;
        list = list->next;
    }
    return result;
}
```



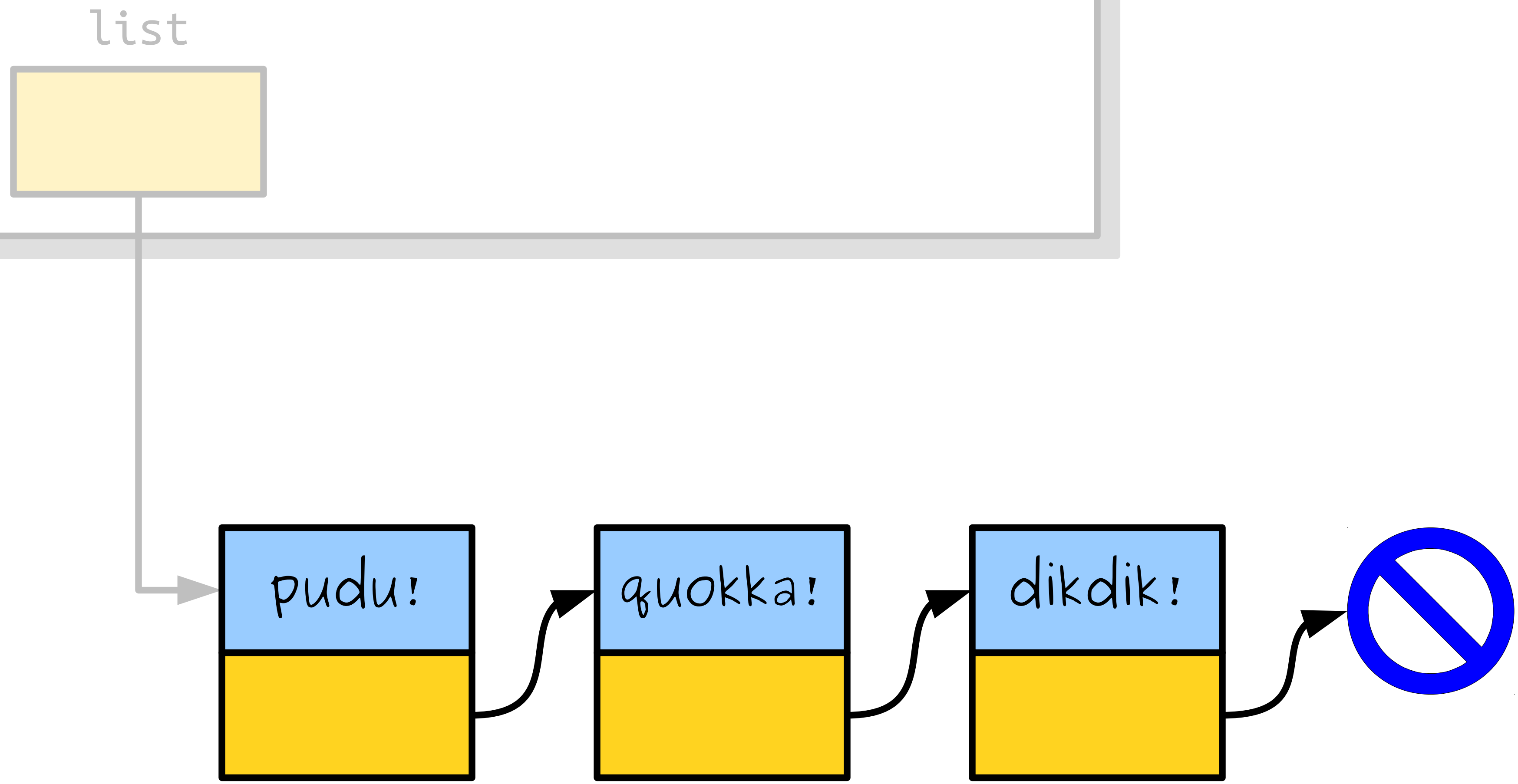
```
int lengthOf(Cell* list) {
    int result = 0;
    while (list != nullptr) {
        result++;
        list = list->next;
    }
    return result;
}
```



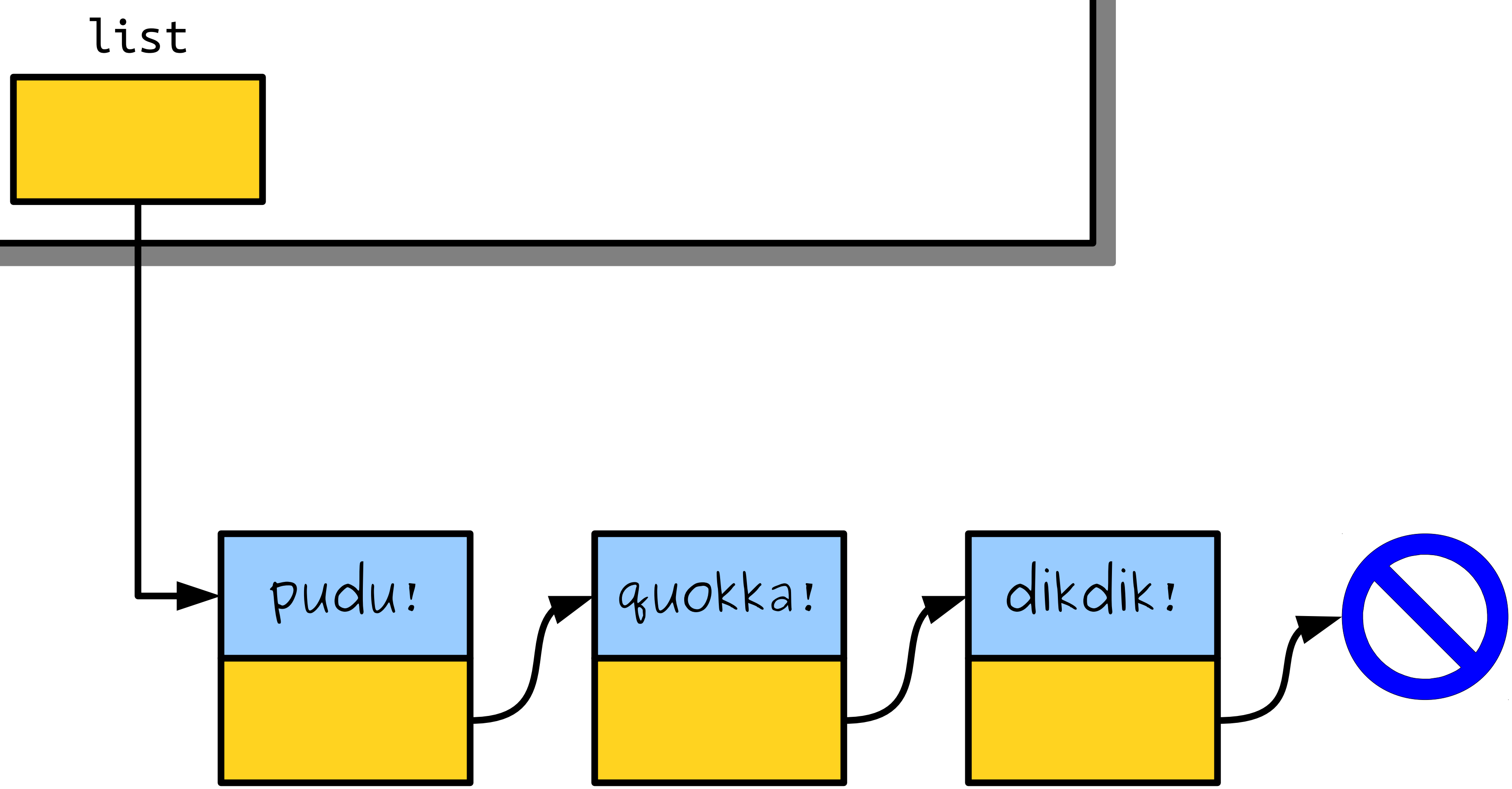

int main() {
int lengthOf(Cell* list) {
int result = 0;
while (list != nullptr) {
result++;
list = list->next;
}
return result;
}
list
result
3
pudu!
quokka!
dikdik!


int main() {
int lengthOf(Cell* list) {
int result = 0;
while (list != nullptr) {
result++;
list = list->next;
}
return result;
}
list
result
3
pudu!
quokka!
dikdik!

```
int main() {
   Cell* list = readList();
   cout << lengthOf(list) << endl;

   /* … other listy things. … */
}
```



```
int main() {
    Cell* list = readList();
    cout << lengthOf(list) << endl;

    /* … other listy things. … */
}
```

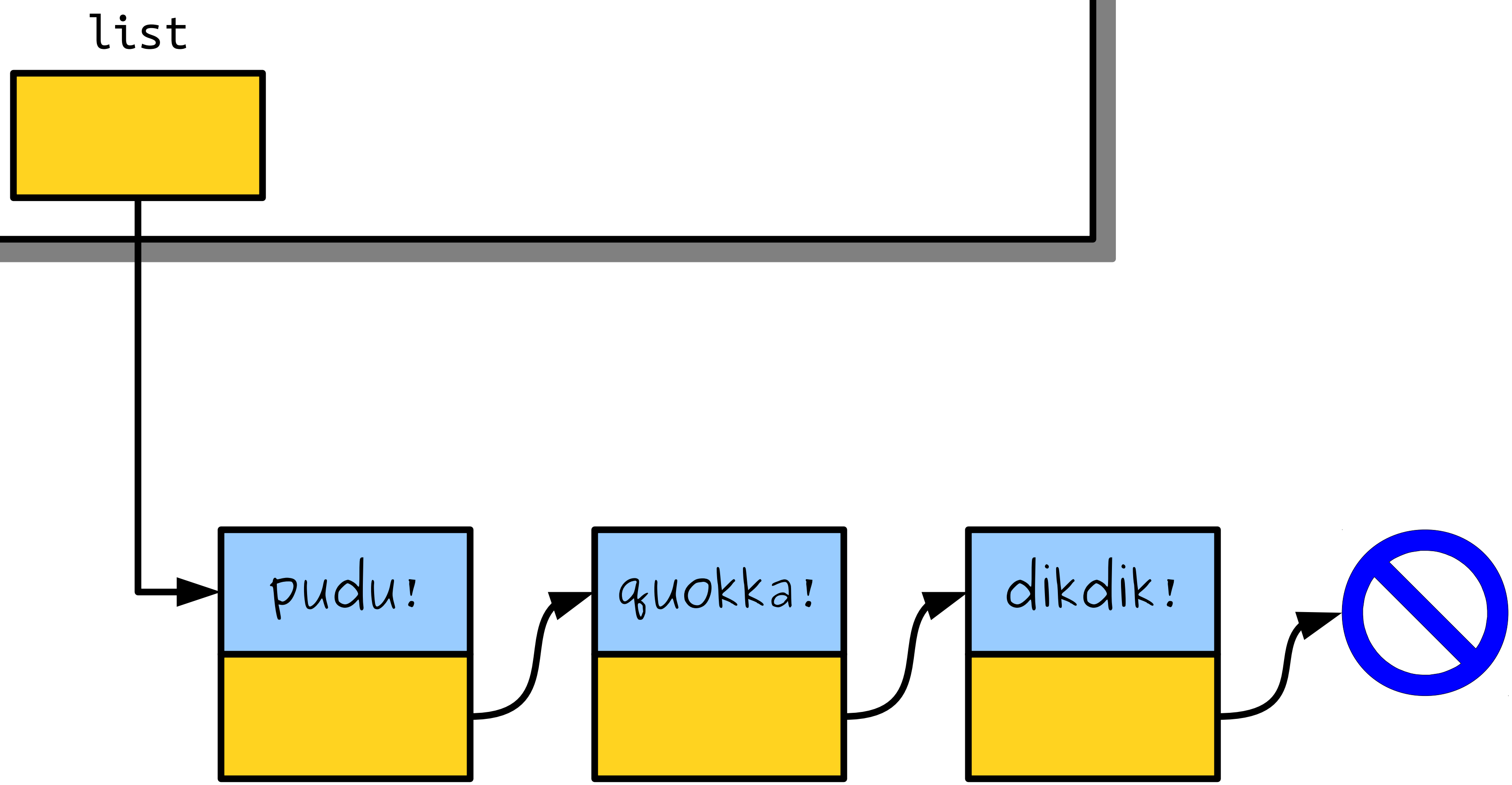


Printing a List

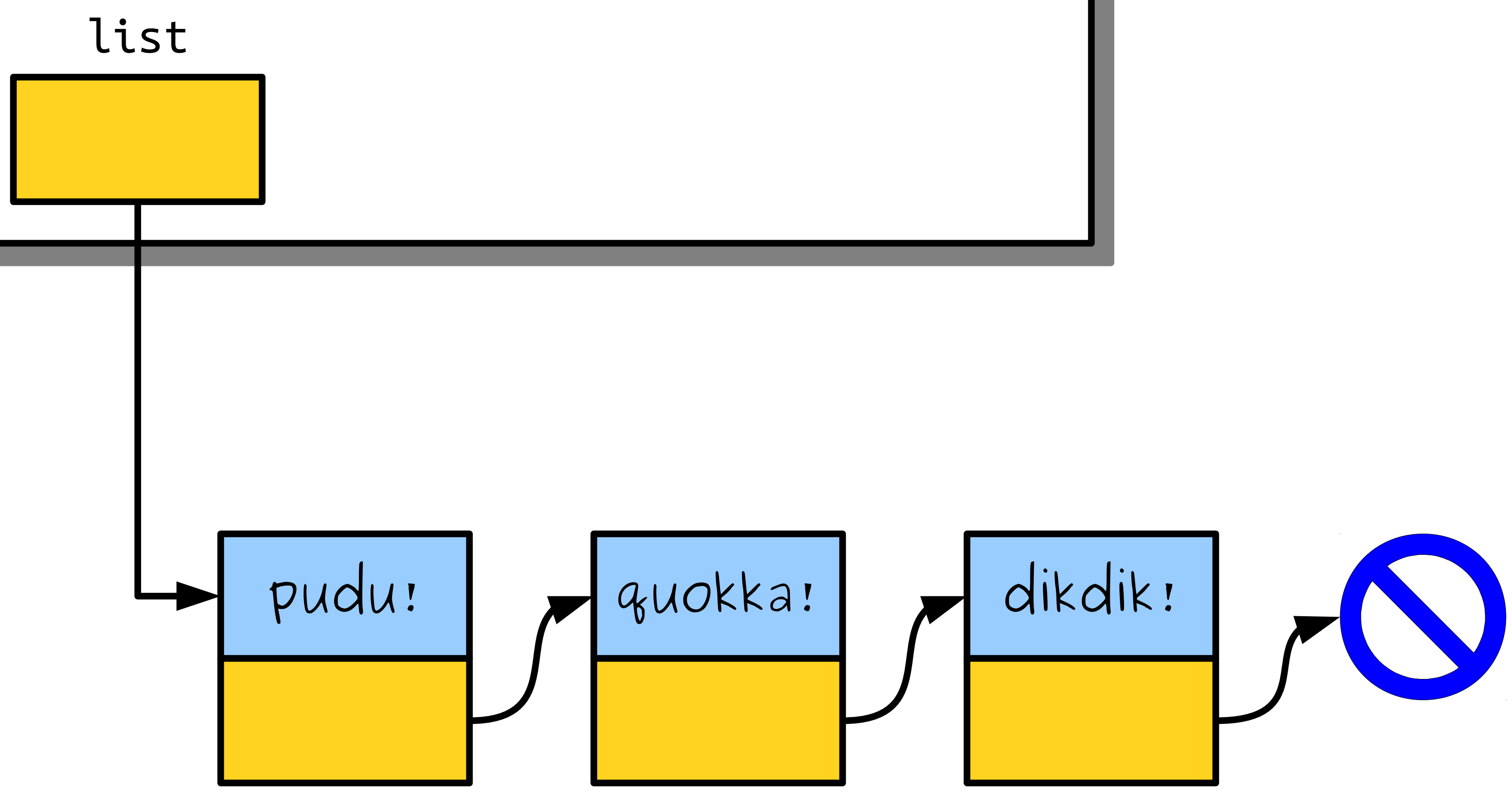
```
int main() {
   Cell* list = readList();
   printList(list);

   /* … other listy things. … */
}
```

```c
int main() {
    Cell* list = readList();
    printList(list);

    /* … other listy things. … */
}
```

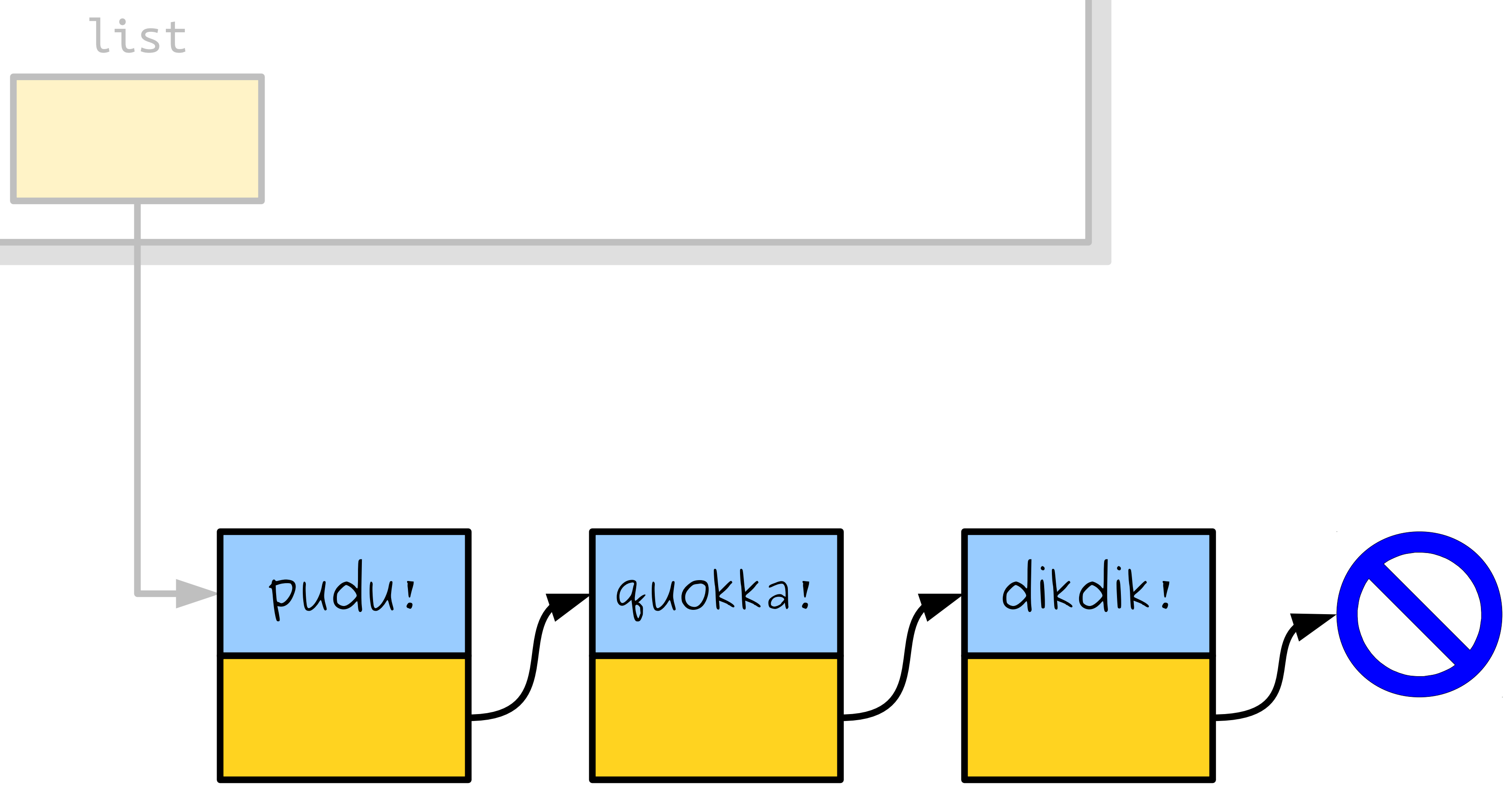
```
int main() {
    Cell* list = readList();
    printList(list);

    /* … other listy things. … */
}
```



```
int main() {
    Cell* list = readList();
    printList(list);

    /* … other listy things. … */
}
```

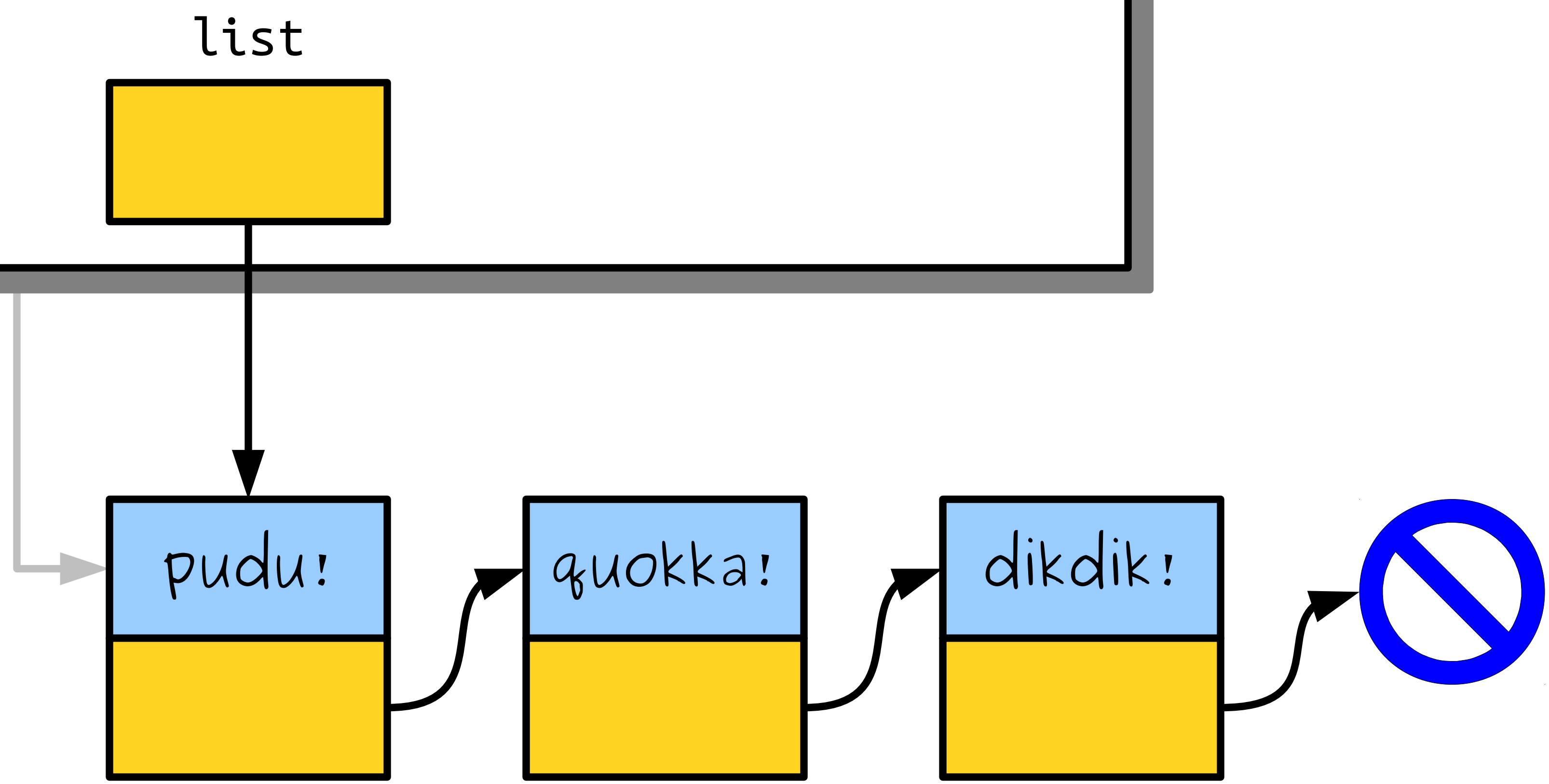


```
int main() {
    Cell* list = readList();
    printList(list);

    /* … other listy things. … */
}
```



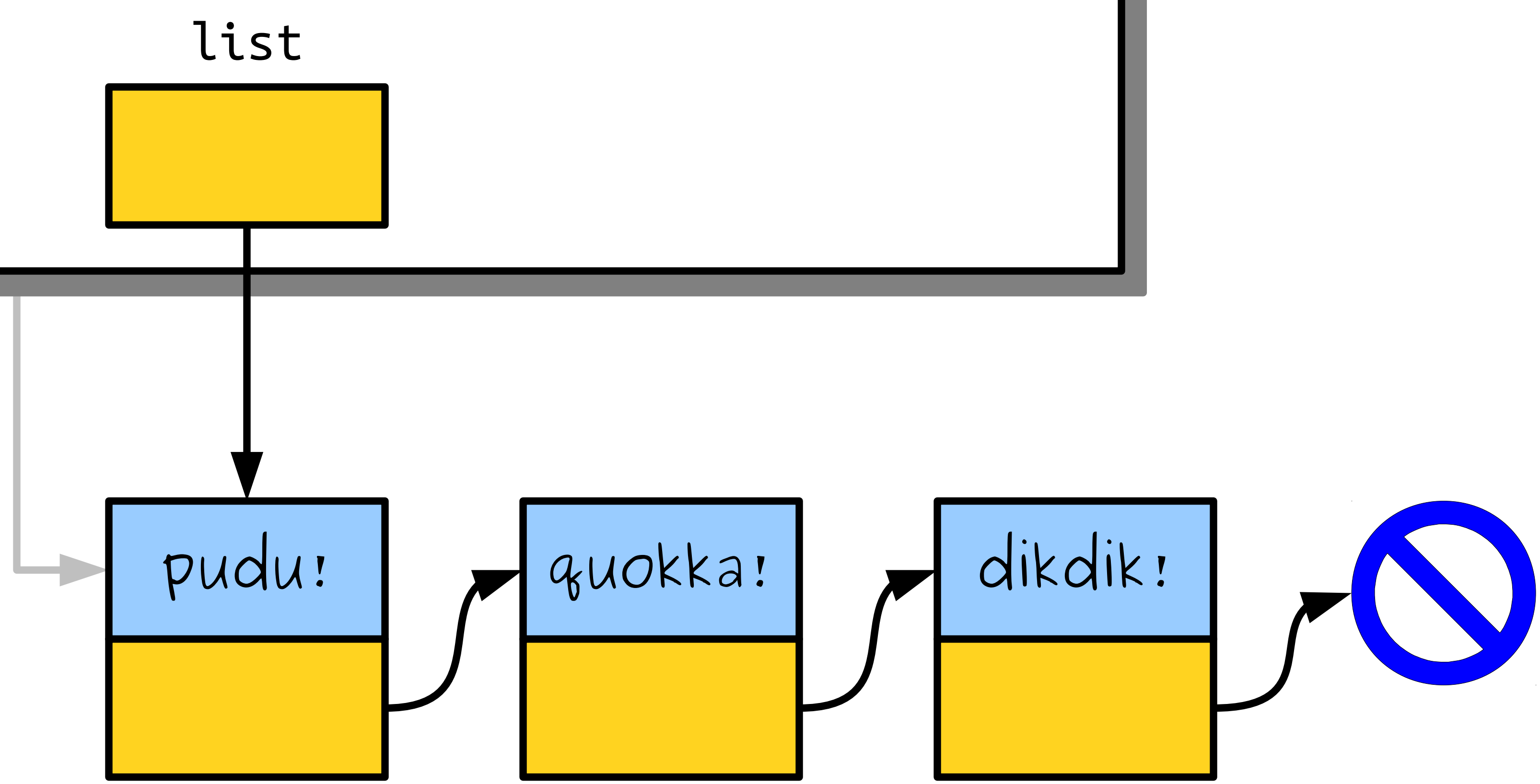
```
int main() {
```

```
void printList(Cell* list) {
    while (list != nullptr) {
        cout << list->value << endl;
        list = list->next;
    }
}
```



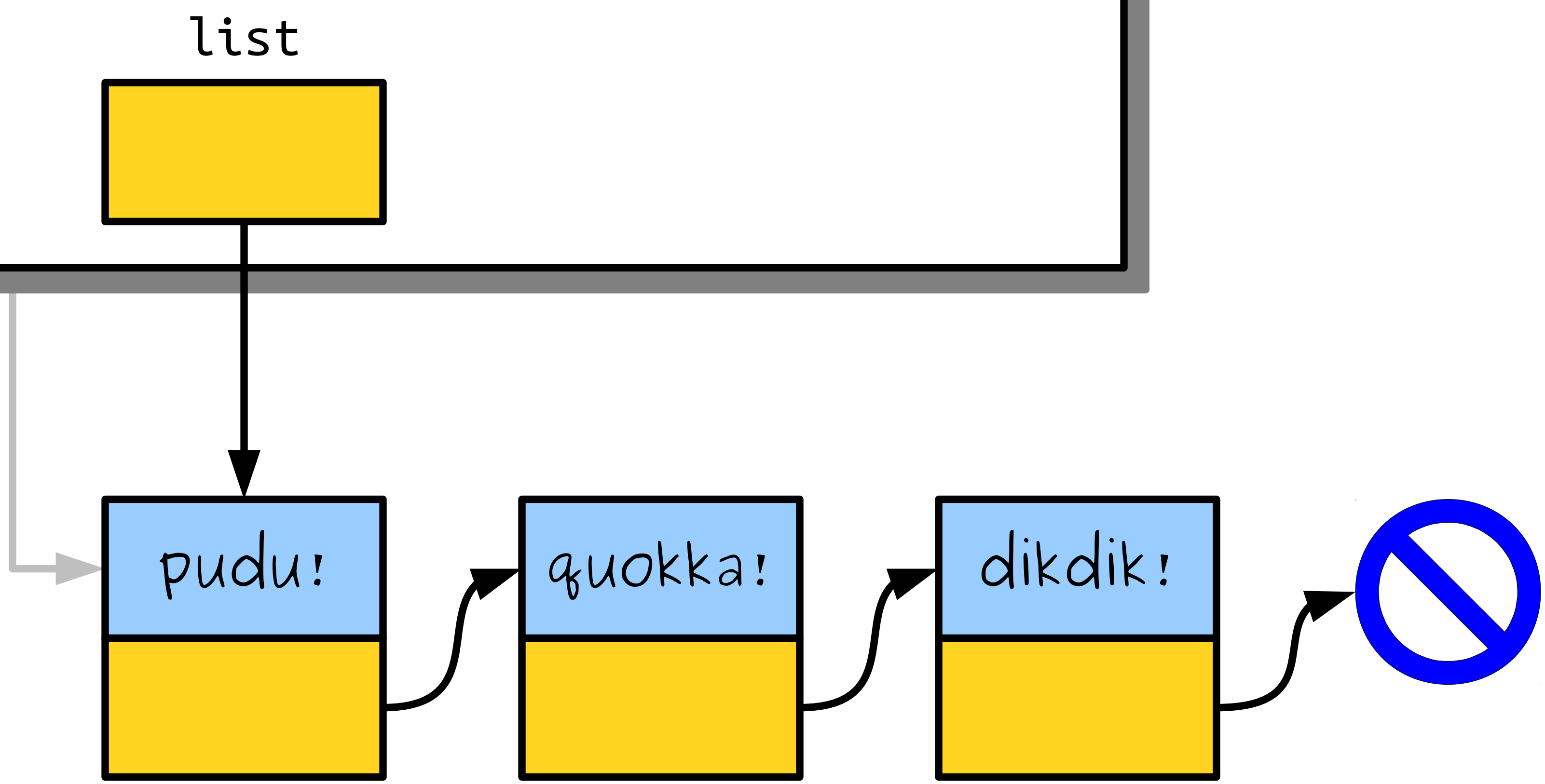
```
int main() {
```

```
void printList(Cell* list) {
    while (list != nullptr) {
        cout << list->value << endl;
        list = list->next;
    }
}
```



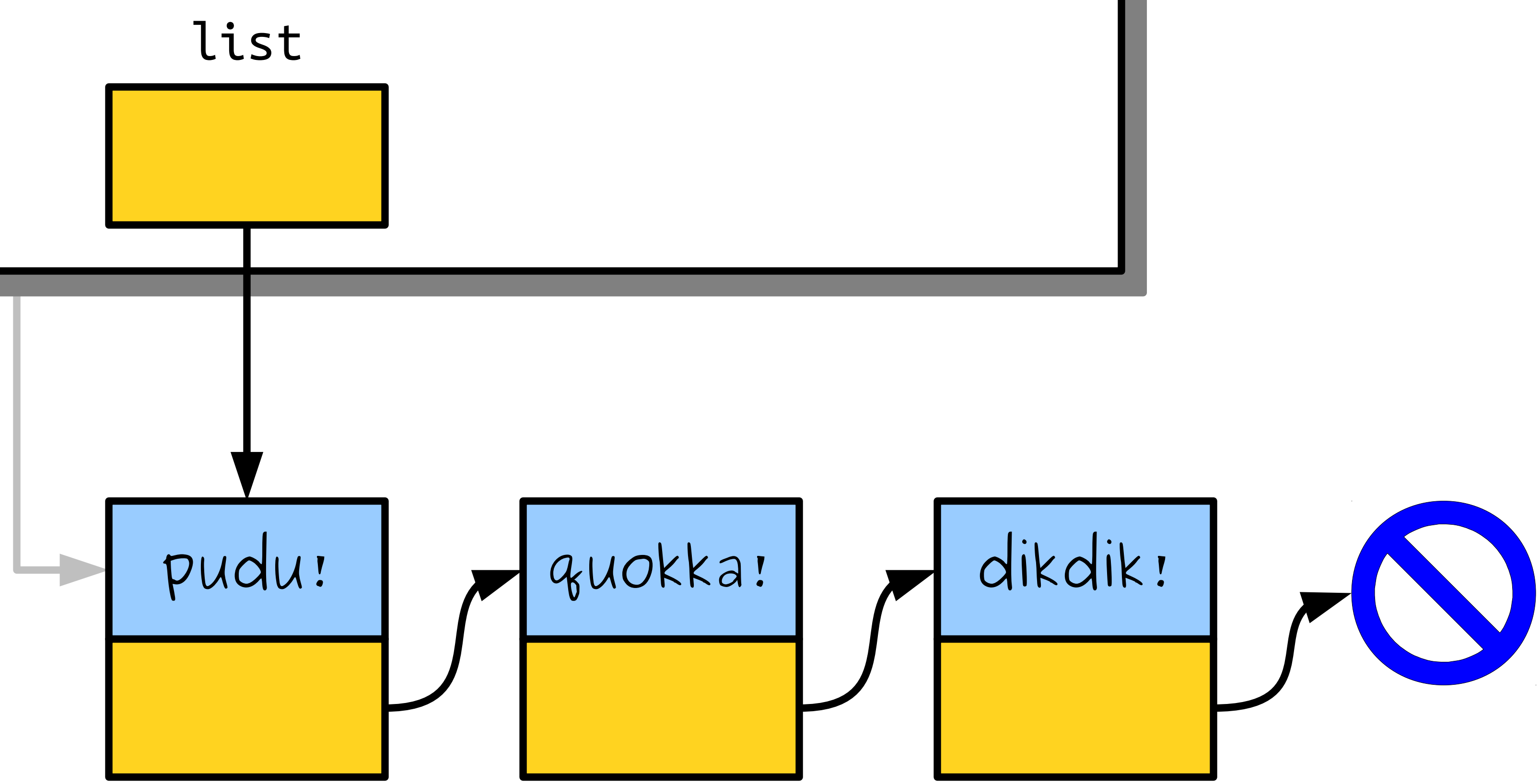
```
int main() {
```

```
void printList(Cell* list) {
    while (list != nullptr) {
        cout << list->value << endl;
        list = list->next;
    }
}
```



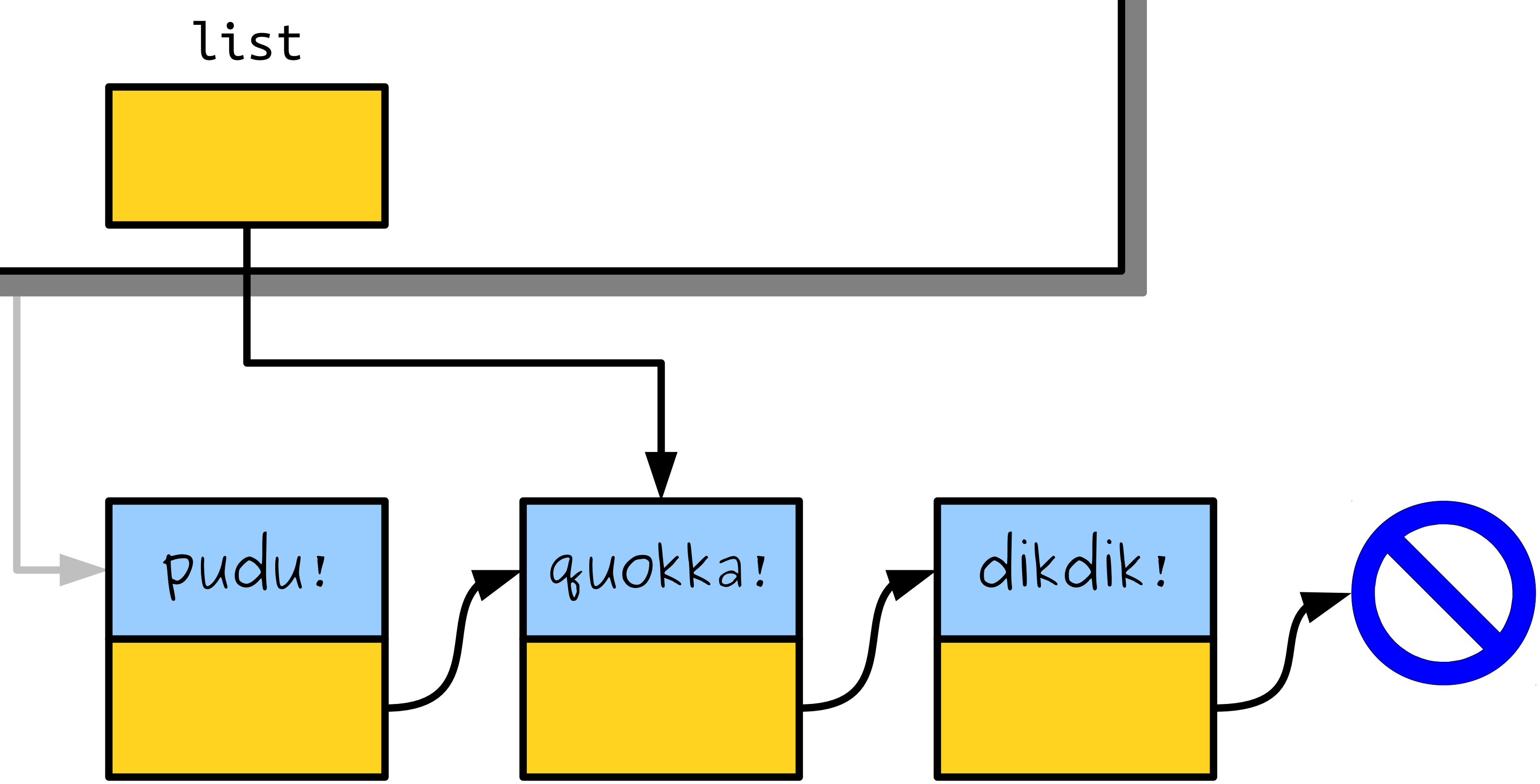
```
int main() {
```

```
void printList(Cell* list) {
    while (list != nullptr) {
        cout << list->value << endl;
        list = list->next;
    }
}
```

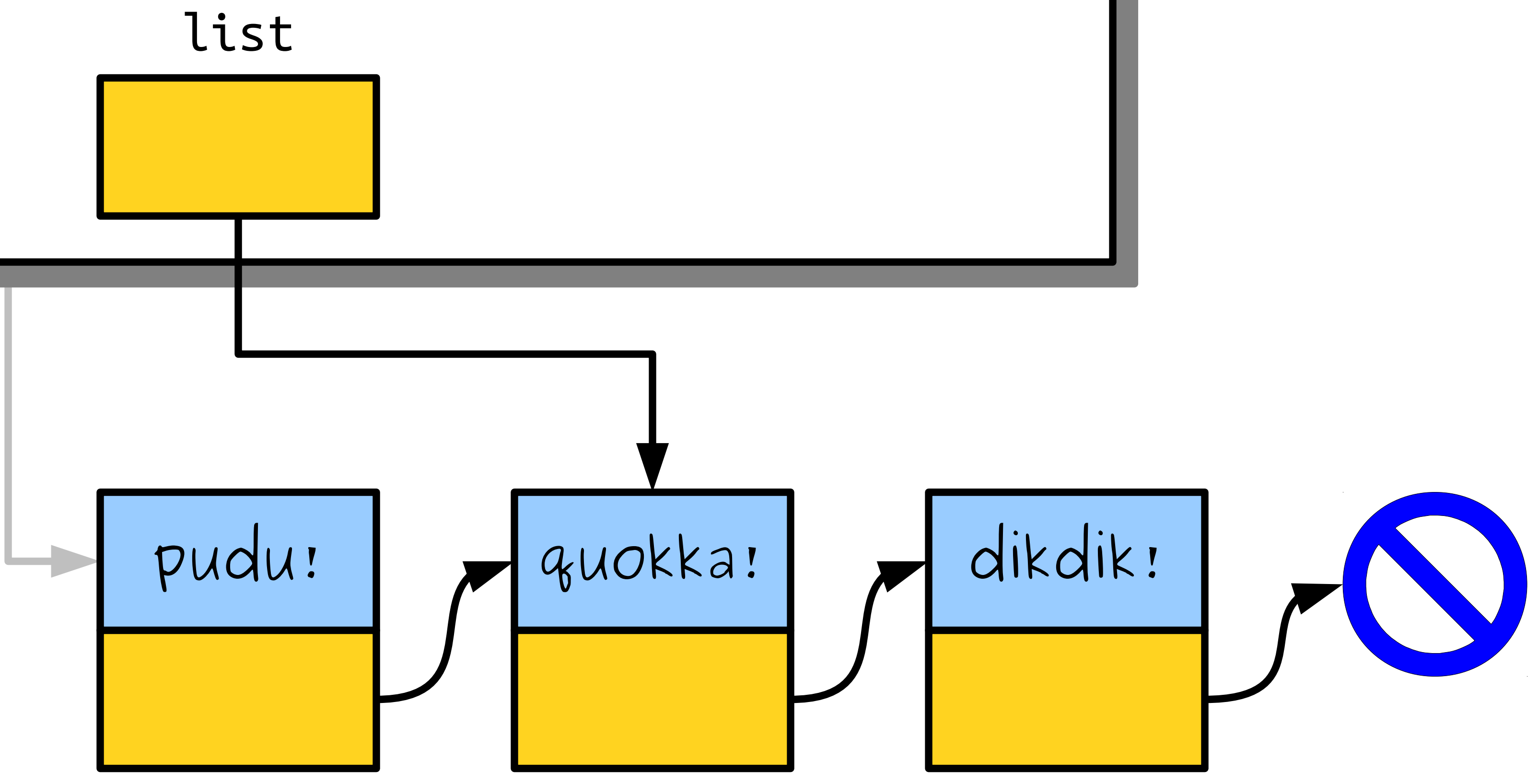


```
int main() {
```

```
void printList(Cell* list) {
    while (list != nullptr) {
        cout << list->value << endl;
        list = list->next;
    }
}
```

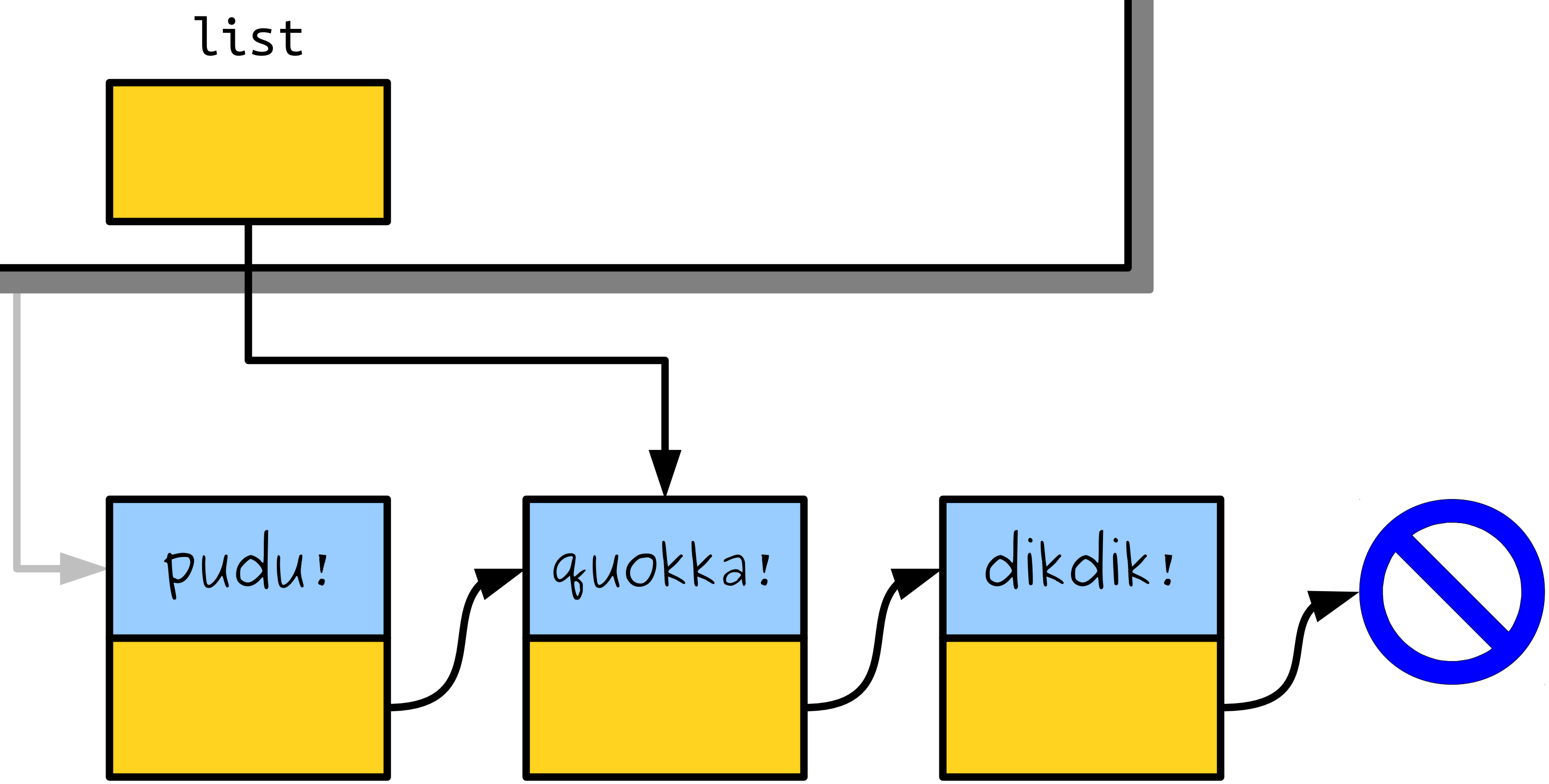


```
int main() {
void printList(Cell* list) {
    while (list != nullptr) {
        cout << list->value << endl;
        list = list->next;
    }
}
```



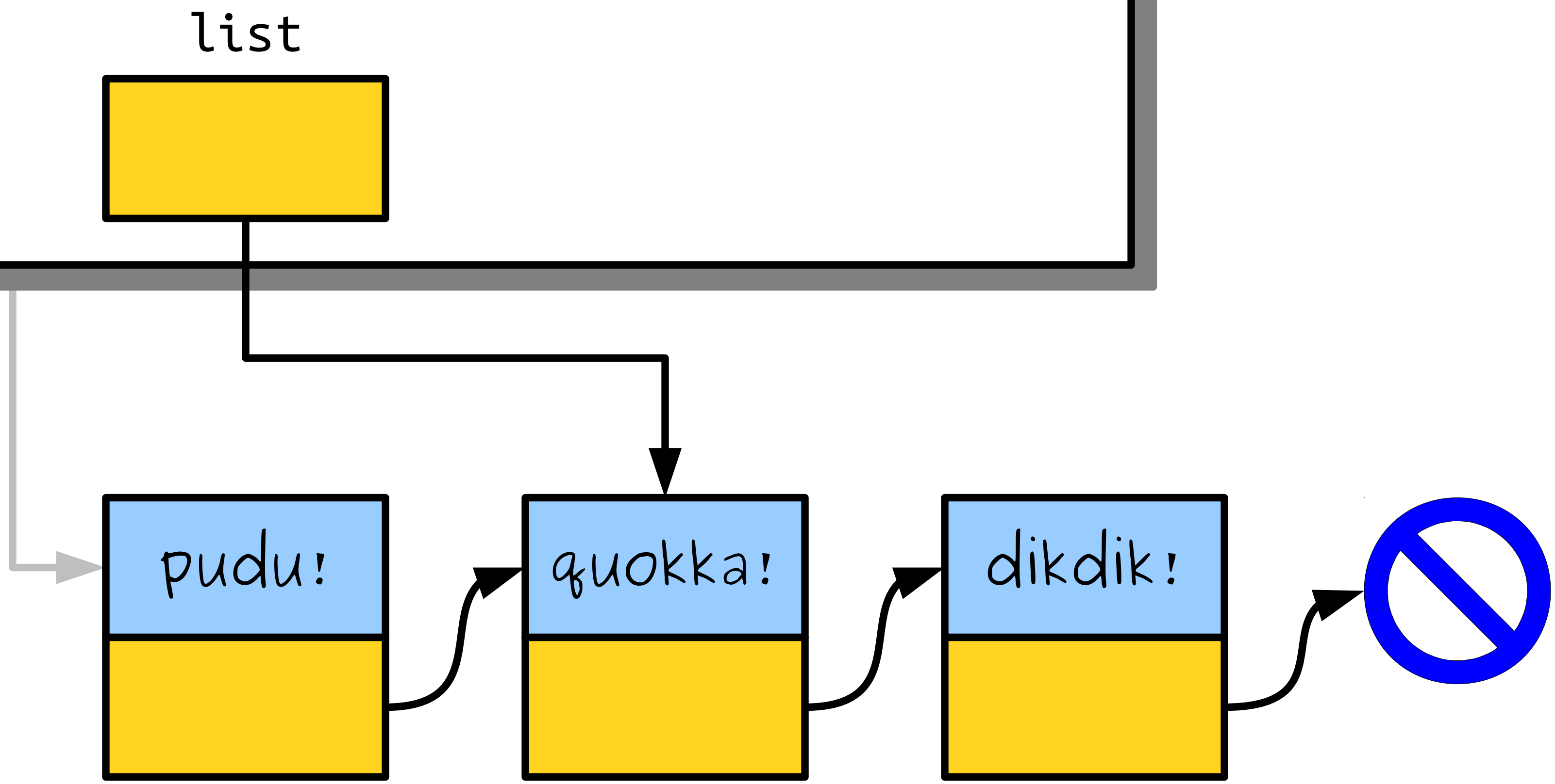
```
int main() {
```

```
void printList(Cell* list) {
    while (list != nullptr) {
        cout << list->value << endl;
        list = list->next;
    }
}
```



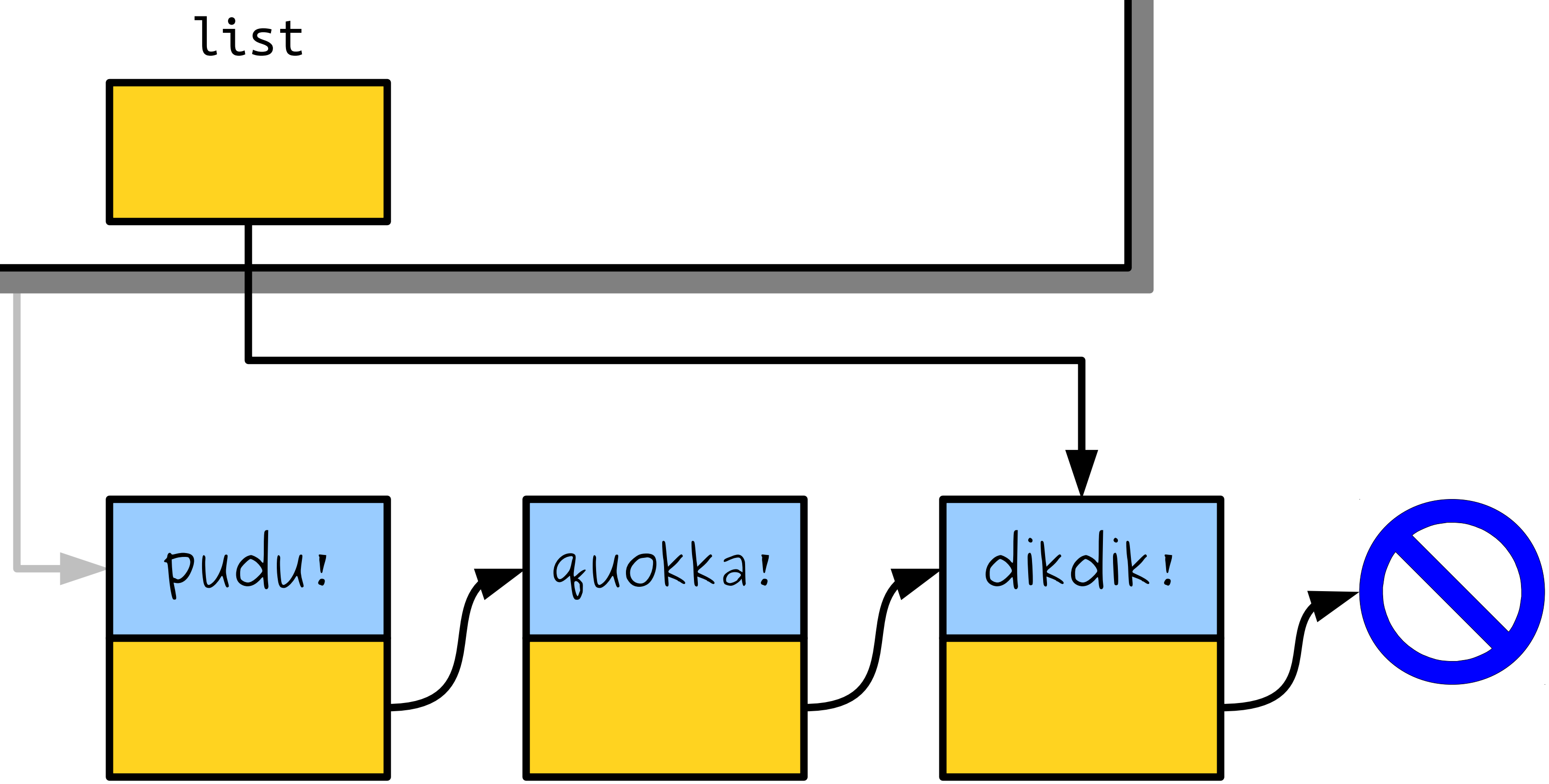
```
int main() {
```

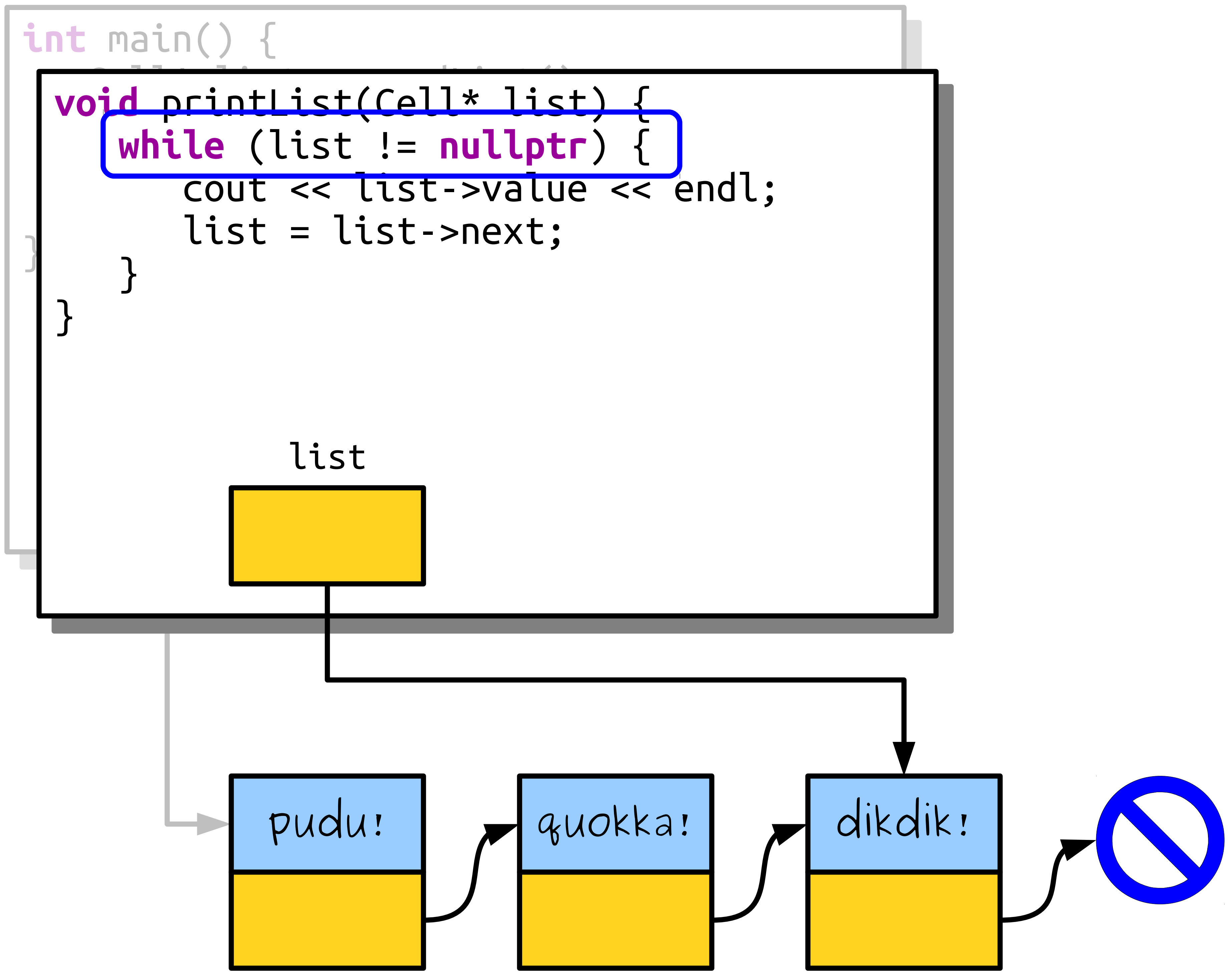
```
void printList(Cell* list) {
    while (list != nullptr) {
        cout << list->value << endl;
        list = list->next;
    }
}
```



```
int main() {
```

```
void printList(Cell* list) {
    while (list != nullptr) {
        cout << list->value << endl;
        list = list->next;
    }
}
```

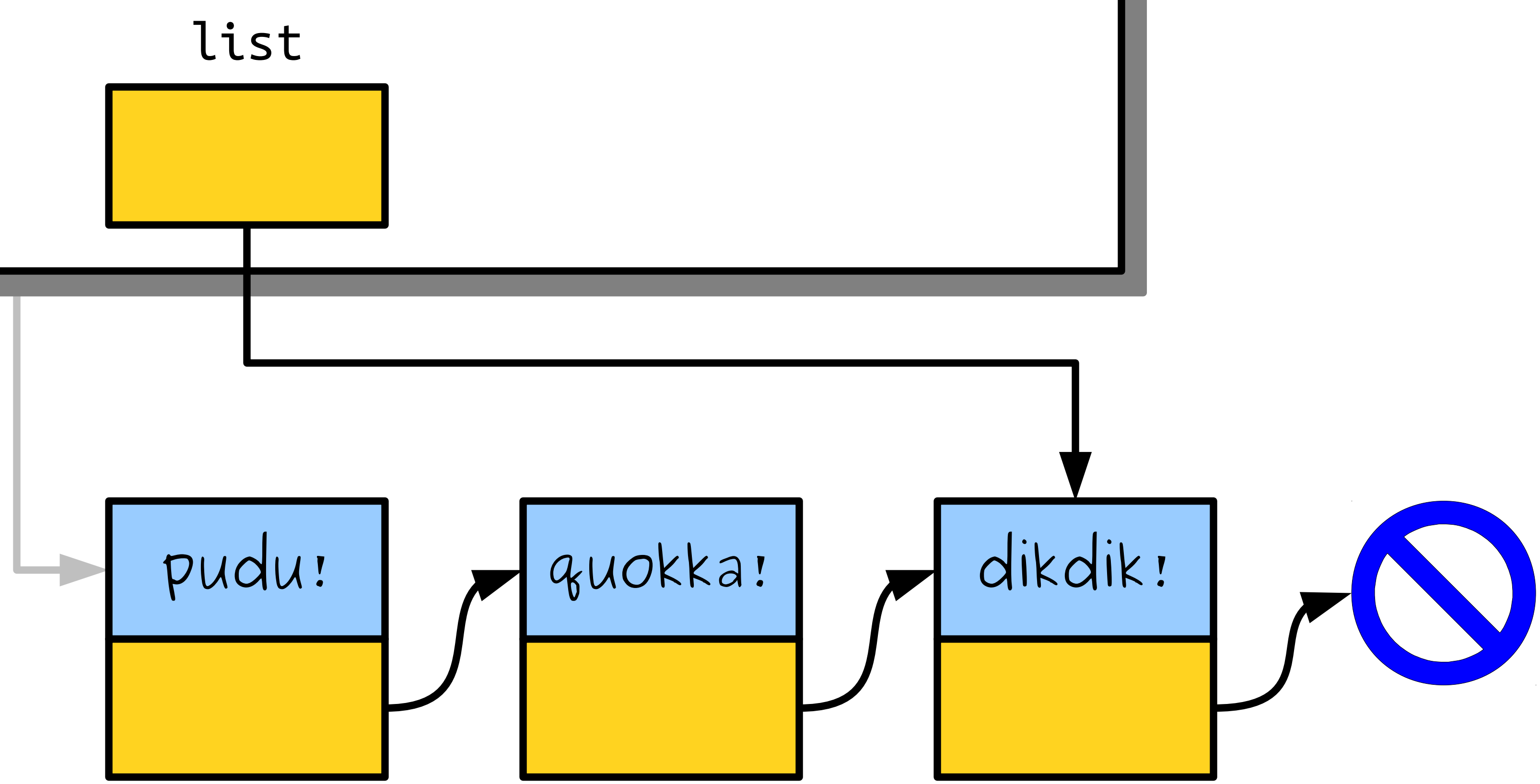




int main() {
void printList(Cell* list) {
while (list != nullptr) {
cout << list->value << endl;
list = list->next;
}
}
list
pudu!
quokka!
dikdik!

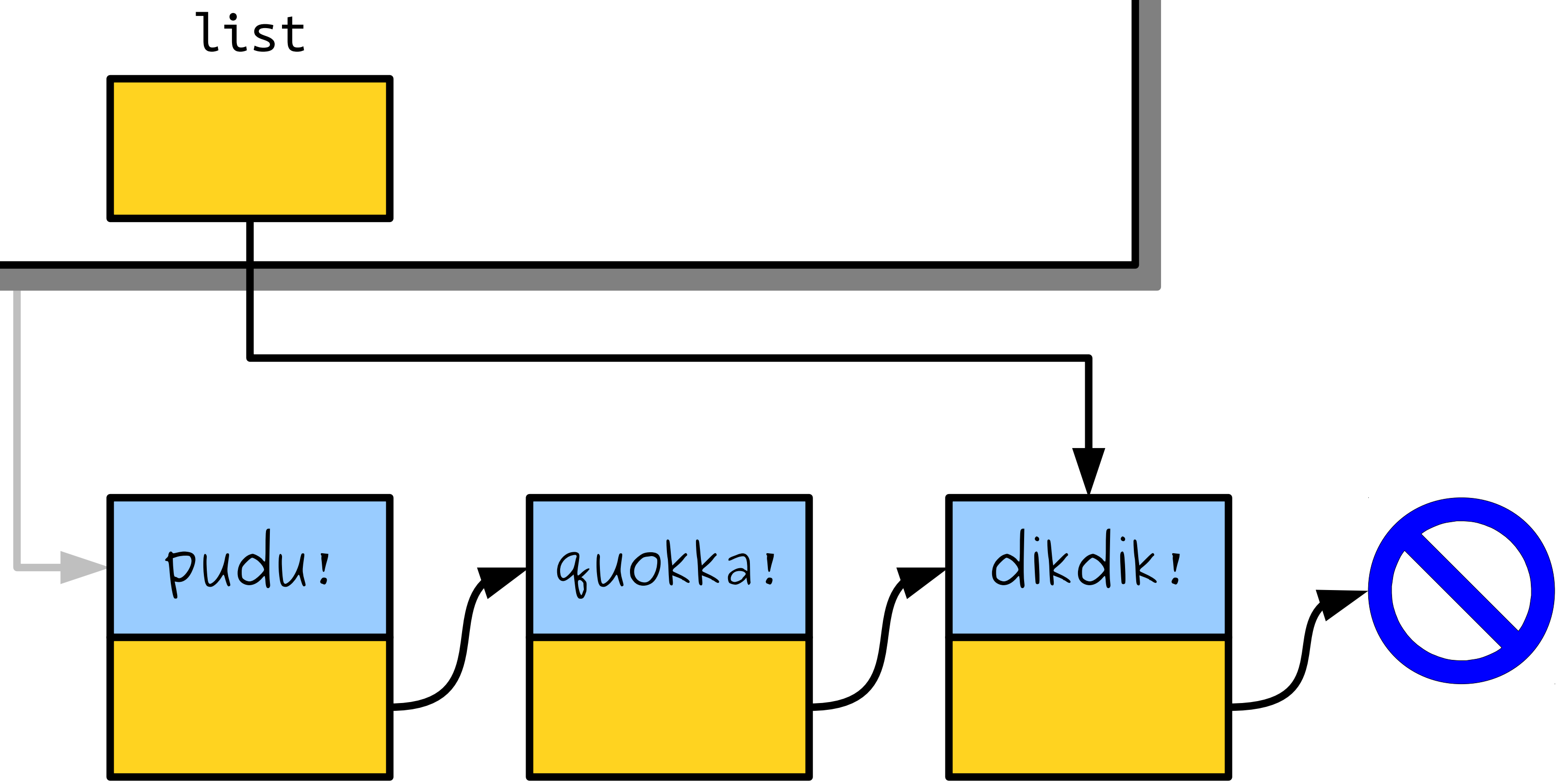
```
int main() {
```

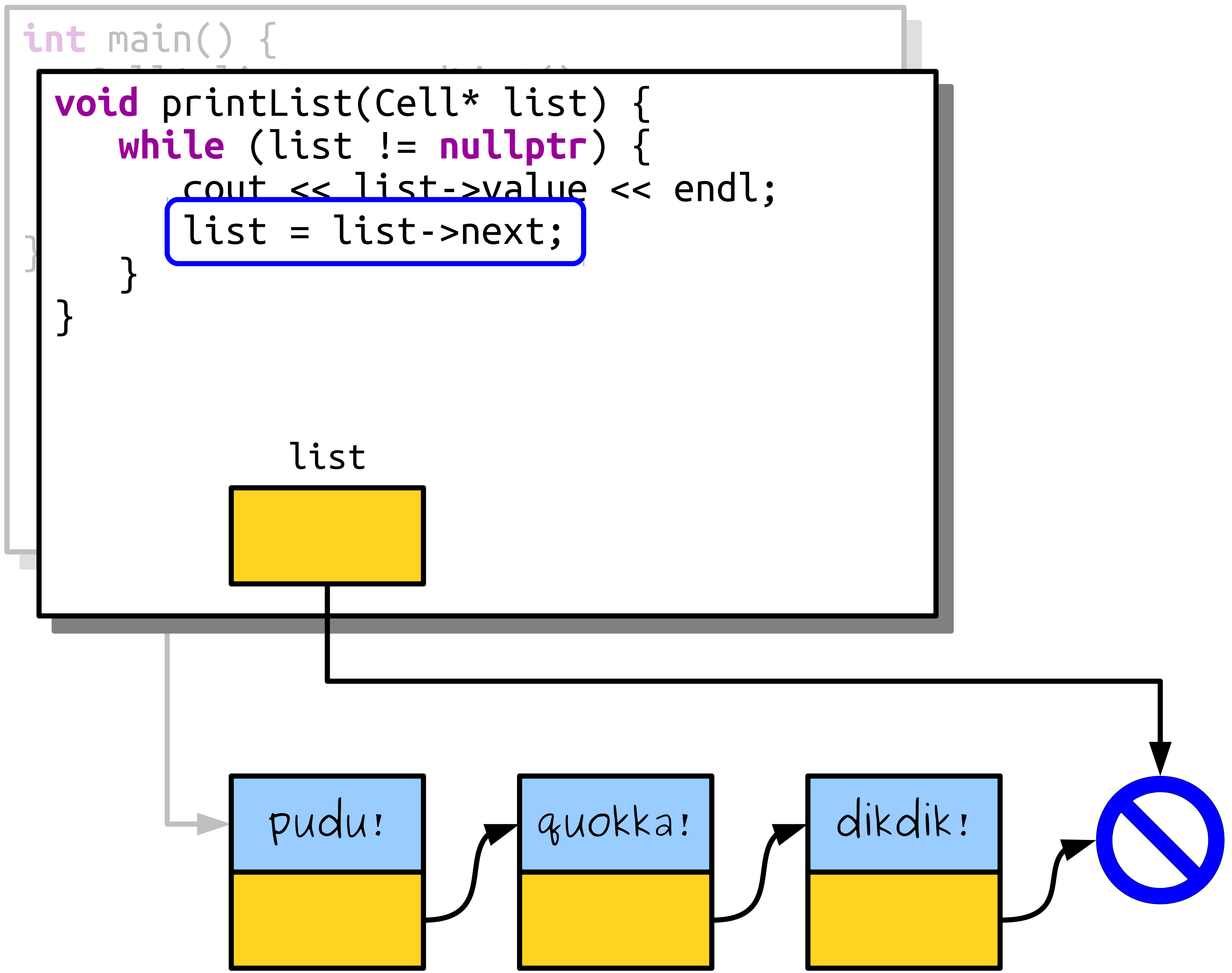
```
void printList(Cell* list) {
    while (list != nullptr) {
        cout << list->value << endl;
        list = list->next;
    }
}
```



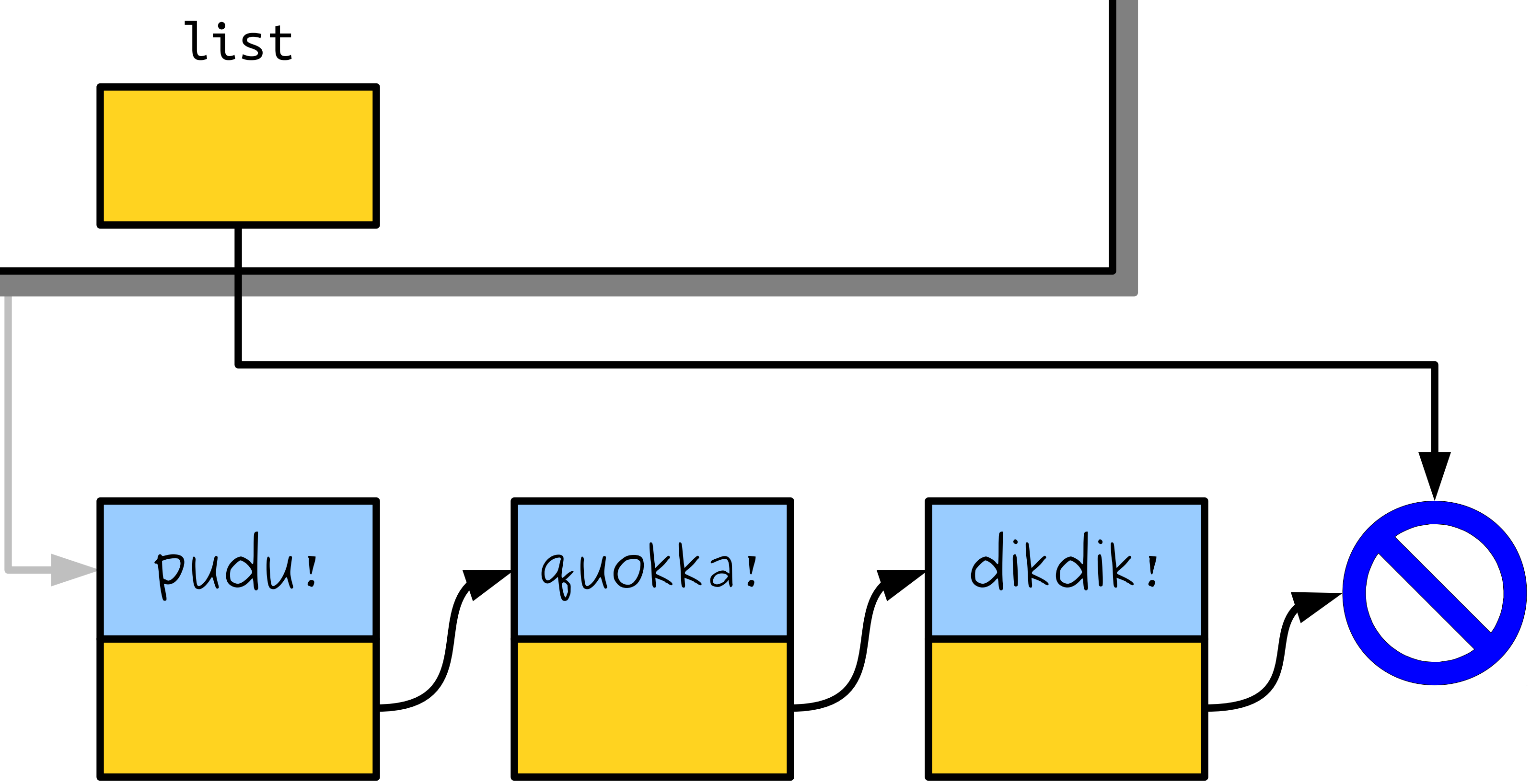
```
int main() {
```

```
void printList(Cell* list) {
    while (list != nullptr) {
        cout << list->value << endl;
        list = list->next;
    }
}
```

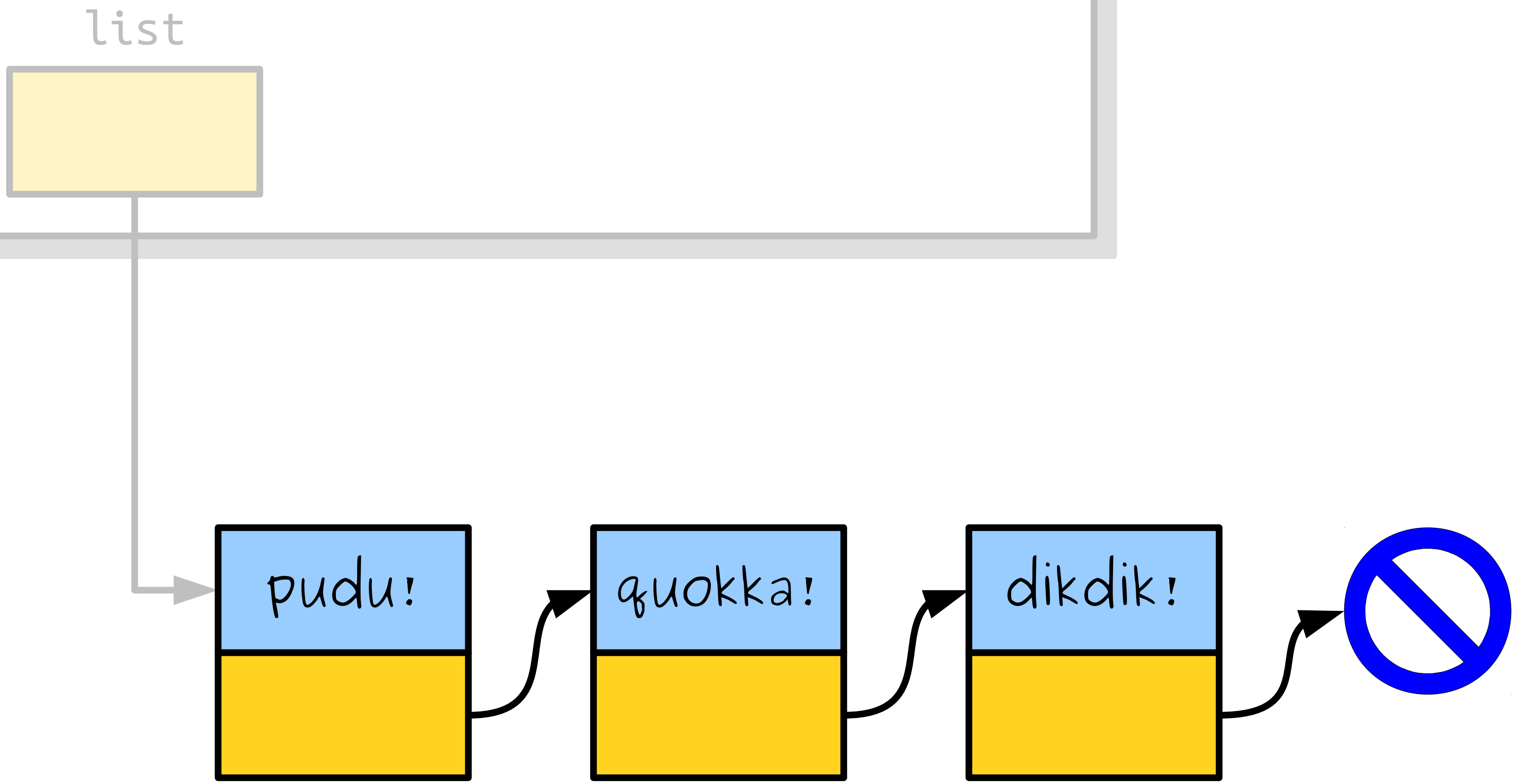



int main() {
}
void printList(Cell* list) {
while (list != nullptr) {
cout << list->value << endl;
list = list->next;
}
}
list
pudu!
quokka!
dikdik!

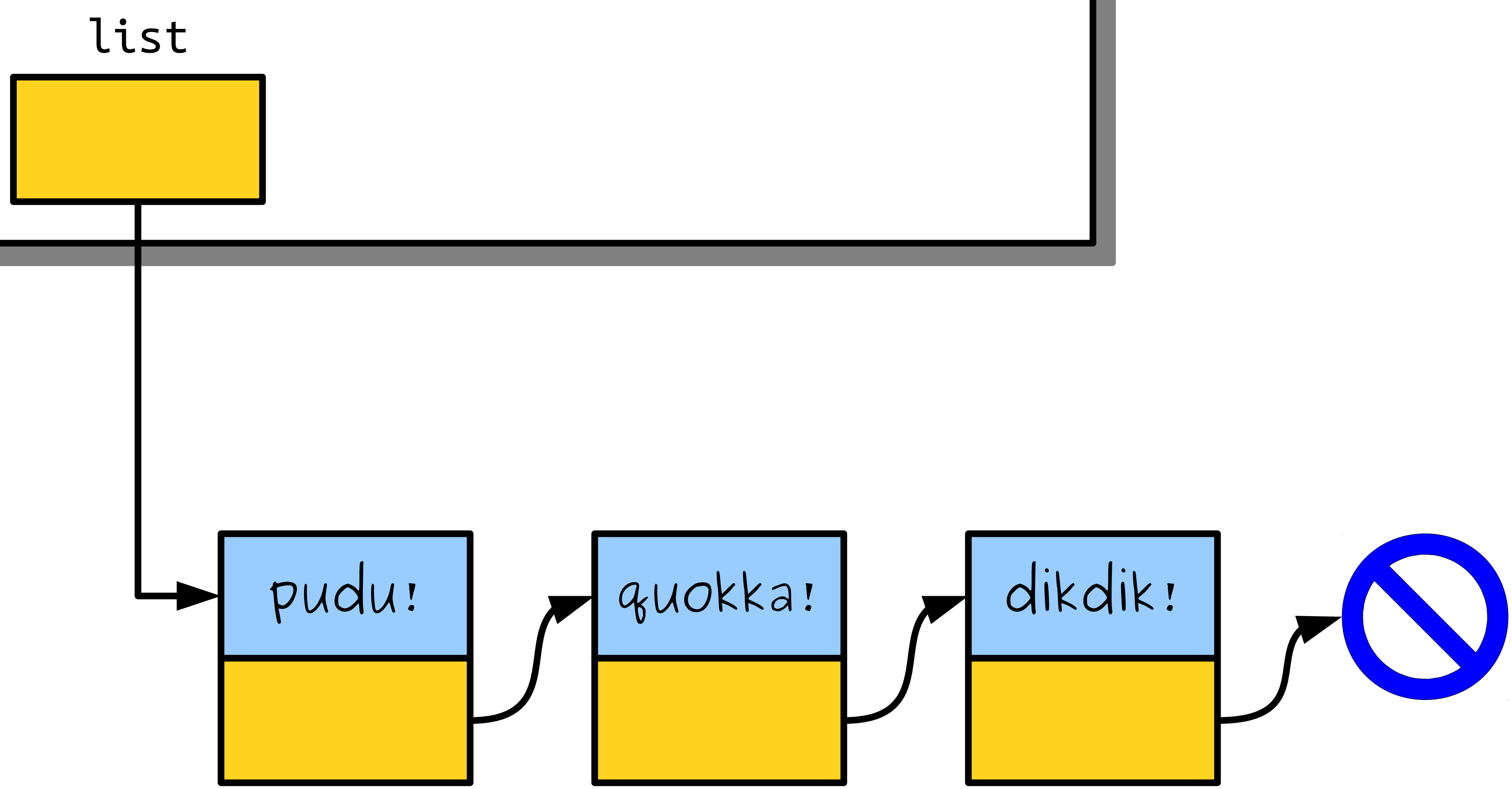
```
void printList(Cell* list) {
    while (list != nullptr) {
        cout << list->value << endl;
        list = list->next;
    }
}
```



```
int main() {
    Cell* list = readList();
    printList(list);

    /* … other listy things. … */
}
```



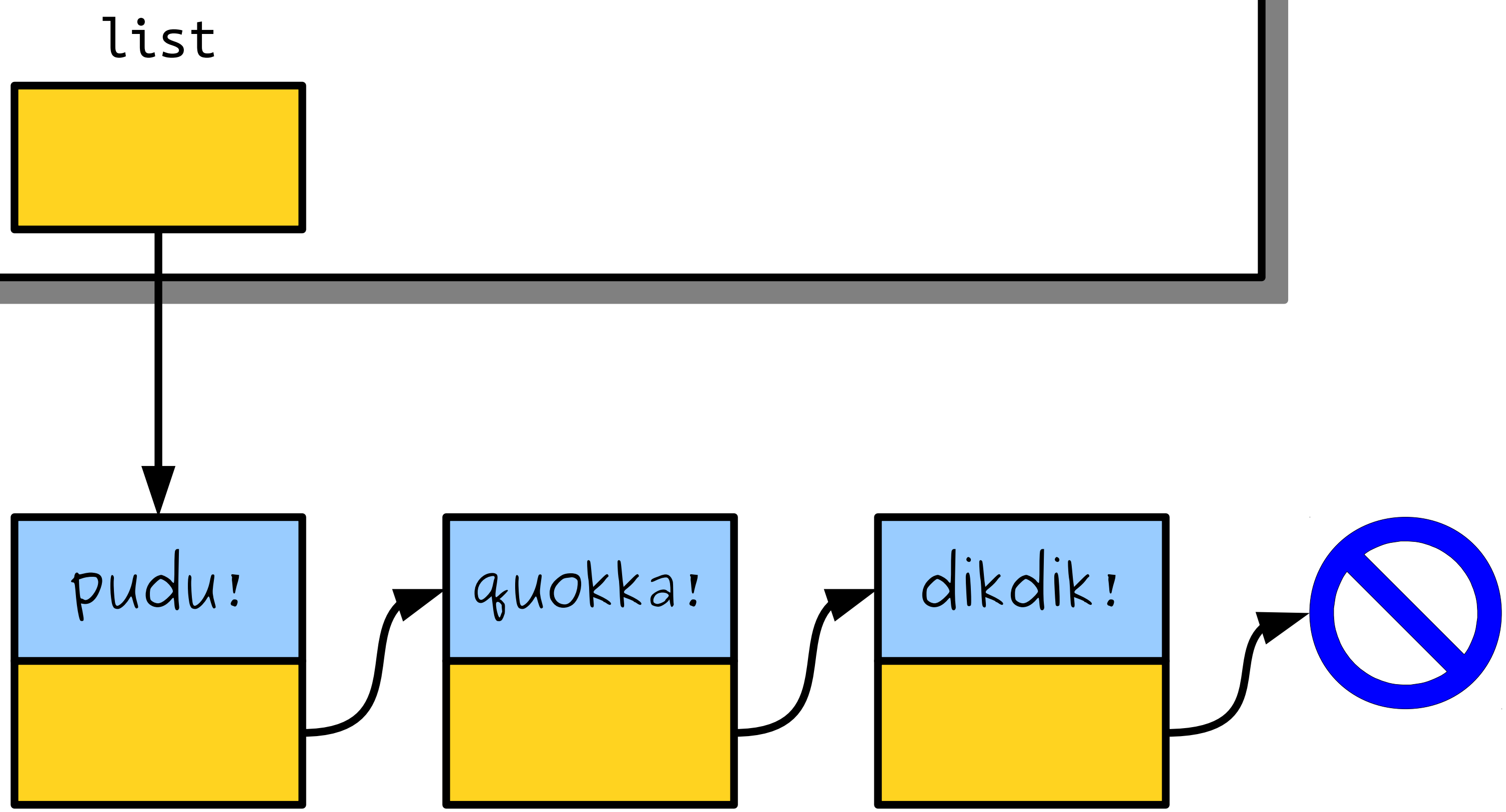
```
int main() {
   Cell* list = readList();
   printList(list);

   /* … other listy things. … */
}
```



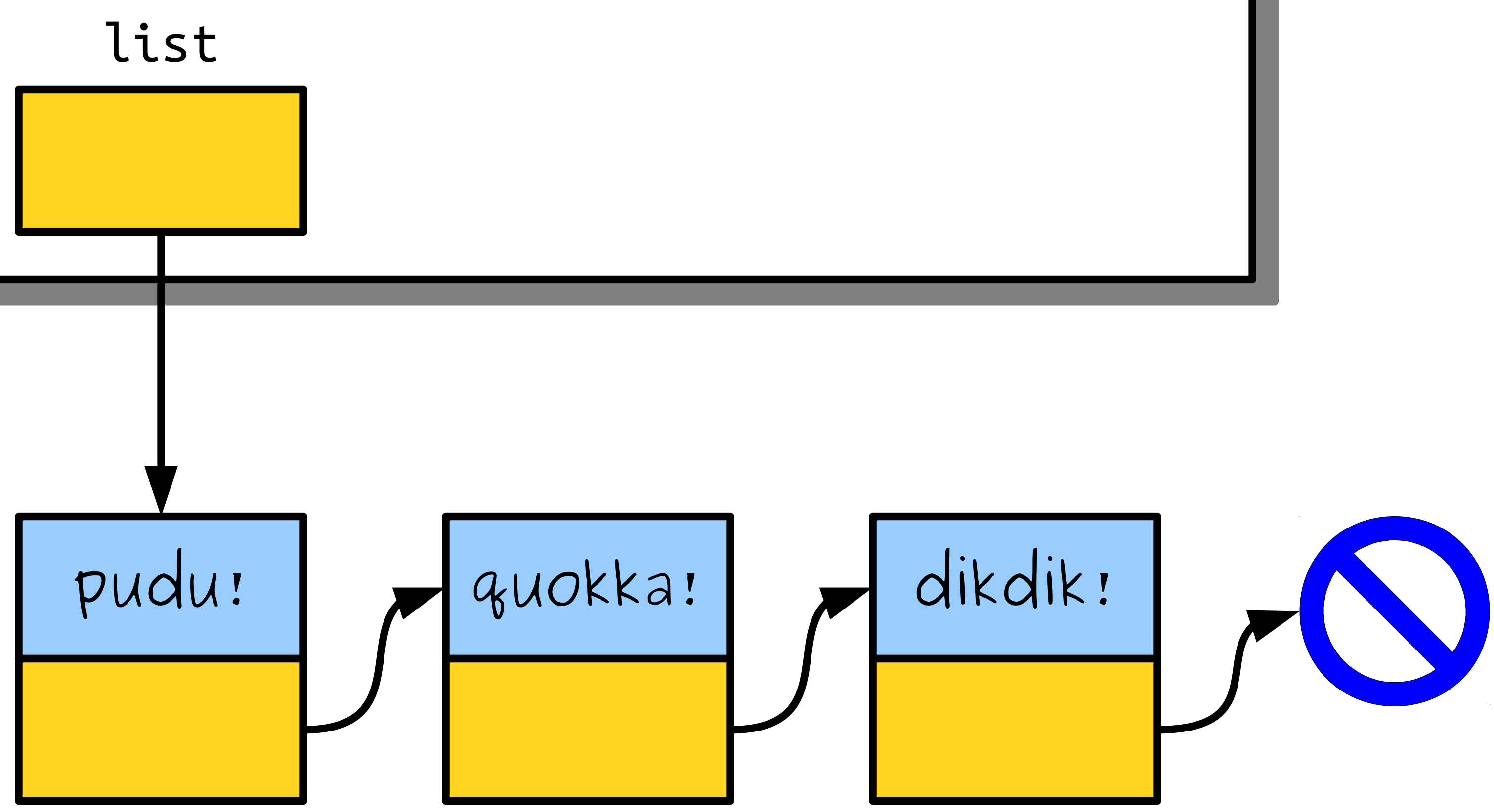
Freeing a Linked List, Iteratively

First, the Wrong Way

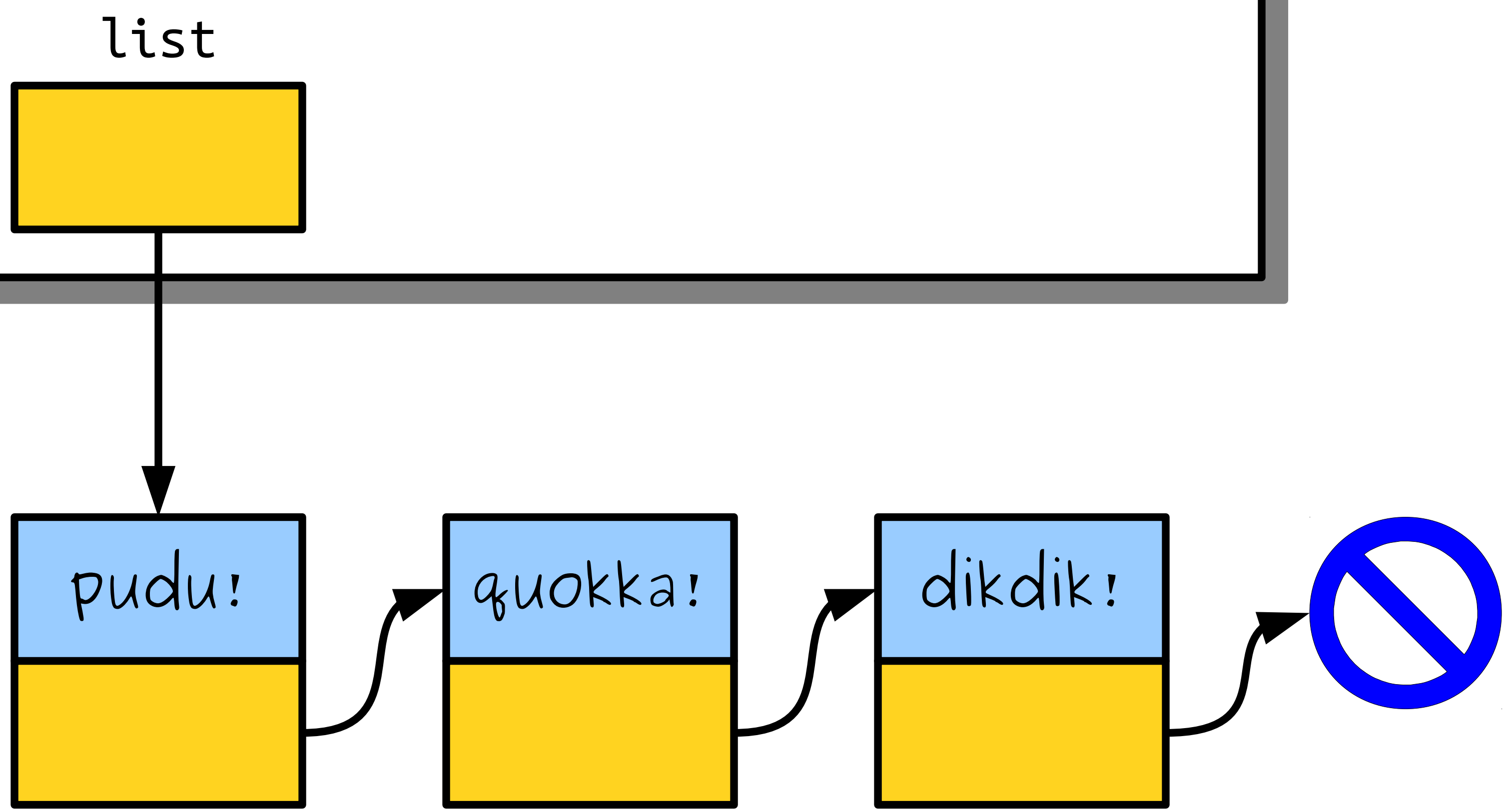
```
void deleteList(Cell* list) {
    // WRONG WRONG WRONG WRONG
    // WRONG WRONG WRONG WRONG

    while (list != nullptr) {
        delete list;
        list = list->next;
    }
}
```



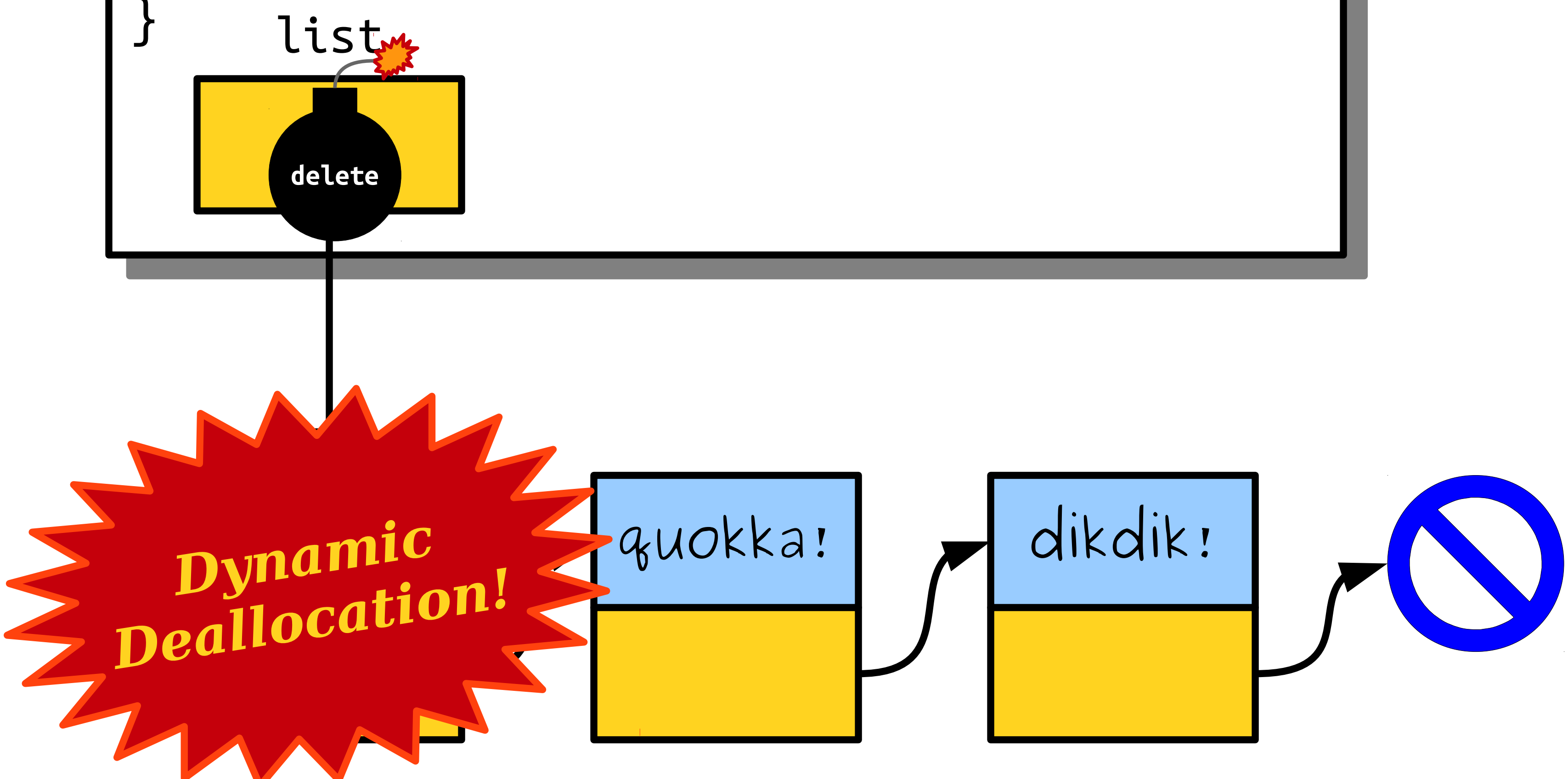
```
void deleteList(Cell* list) {
    // WRONG WRONG WRONG WRONG
    // WRONG WRONG WRONG WRONG

    while (list != nullptr) {
        delete list;
        list = list->next;
    }
}
```



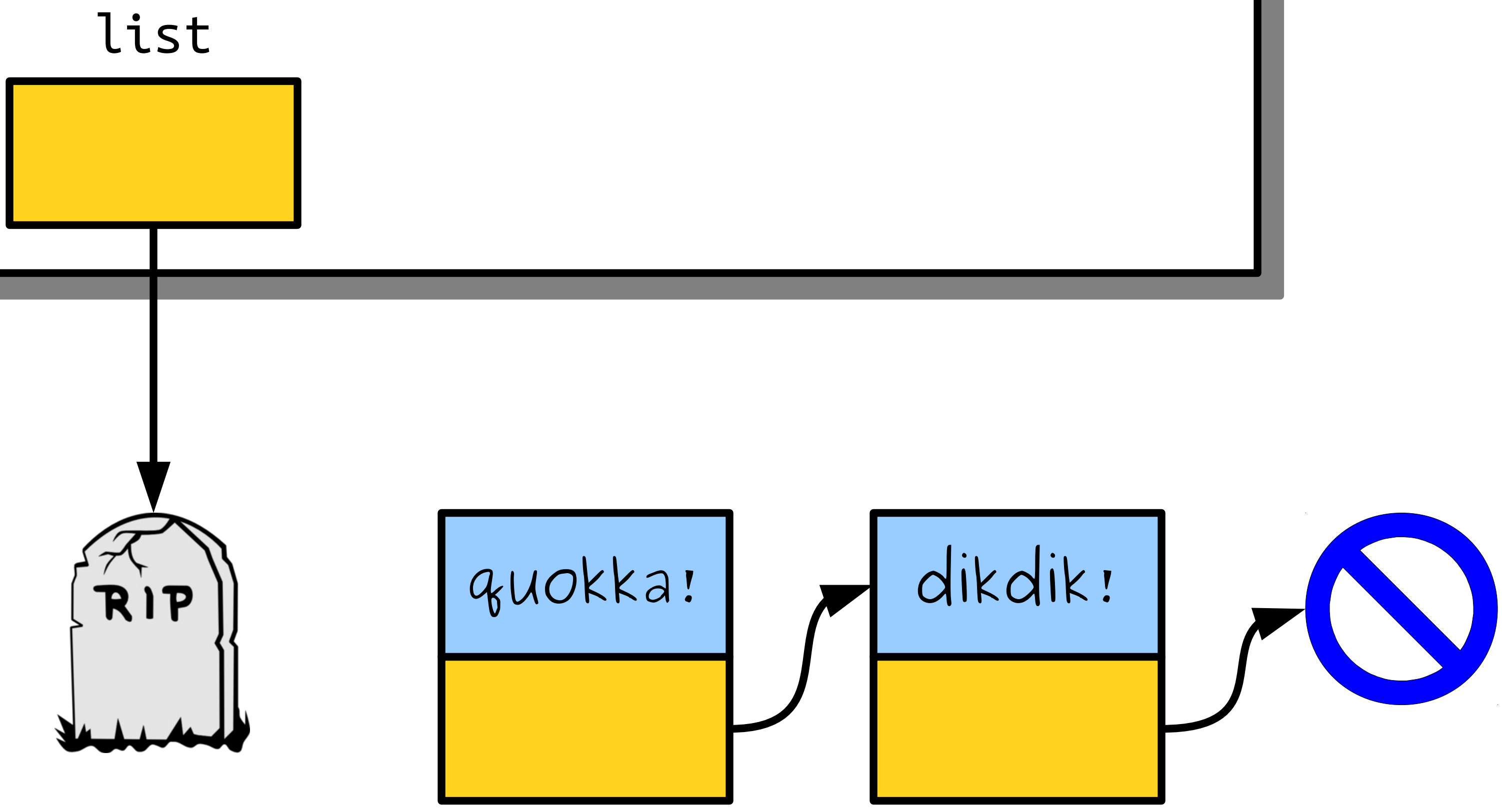
```
void deleteList(Cell* list) {
    // WRONG WRONG WRONG WRONG
    // WRONG WRONG WRONG WRONG

    while (list != nullptr) {
        delete list;
        list = list->next;
    }
}
```



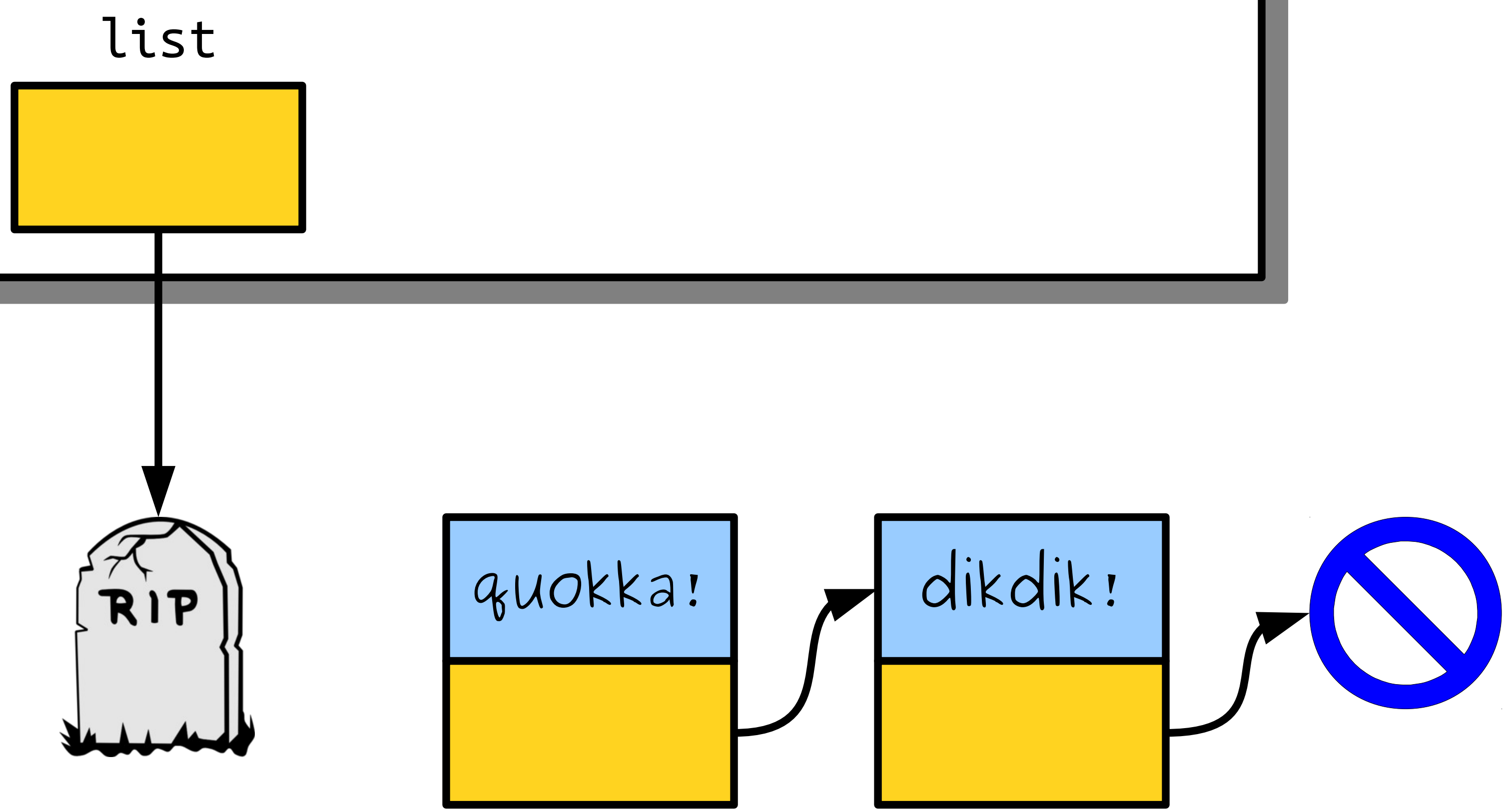
```
void deleteList(Cell* list) {
    // WRONG WRONG WRONG WRONG
    // WRONG WRONG WRONG WRONG

    while (list != nullptr) {
        delete list;
        list = list->next;
    }
}
```



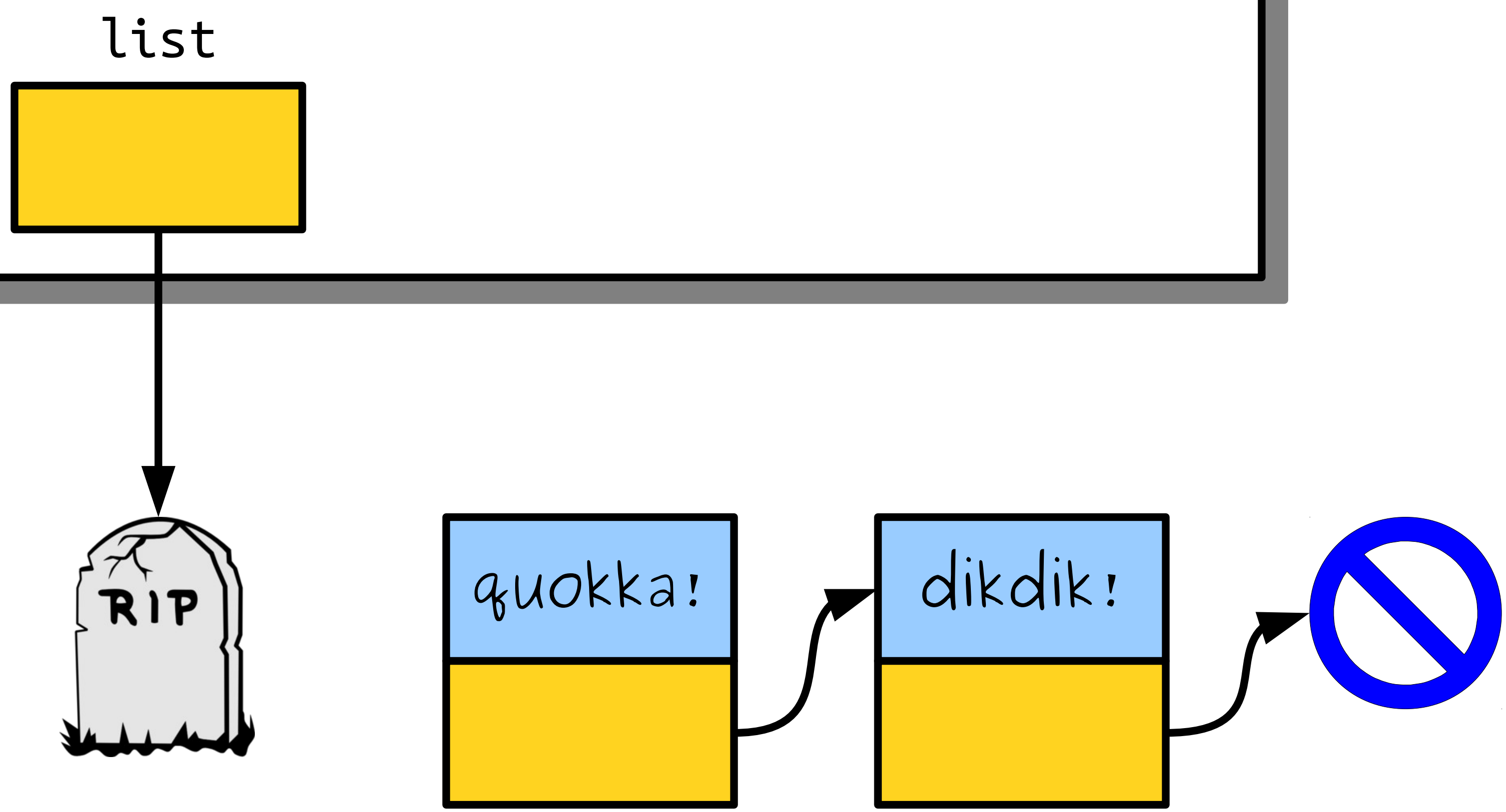
```
void deleteList(Cell* list) {
    // WRONG WRONG WRONG WRONG
    // WRONG WRONG WRONG WRONG

    while (list != nullptr) {
        delete list;
        list = list->next;
    }
}
```



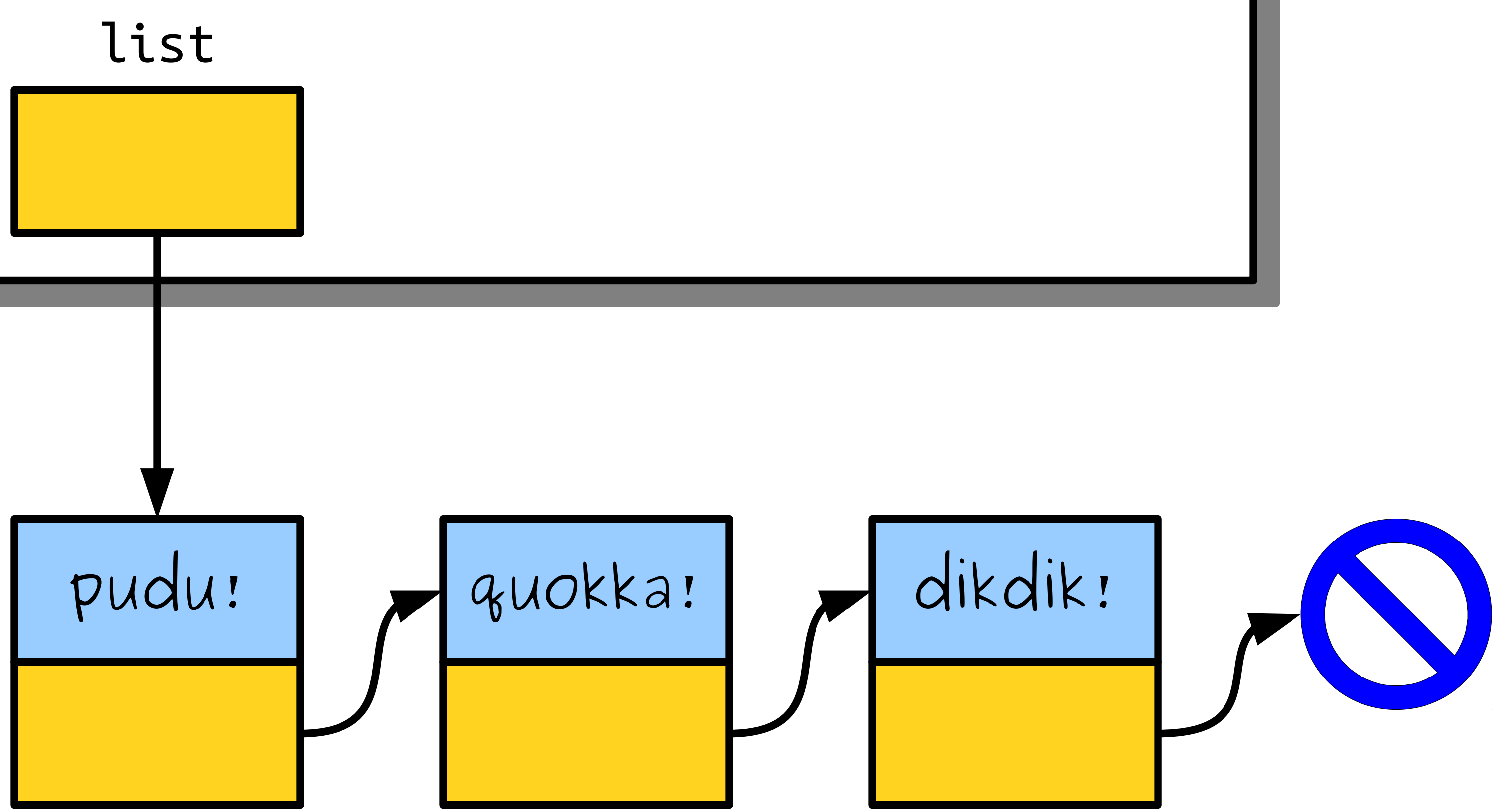
```
void deleteList(Cell* list) {
    // WRONG WRONG WRONG WRONG
    // WRONG WRONG WRONG WRONG

    while (list != nullptr) {
        delete list;
        list = list->next;
    }
}
```



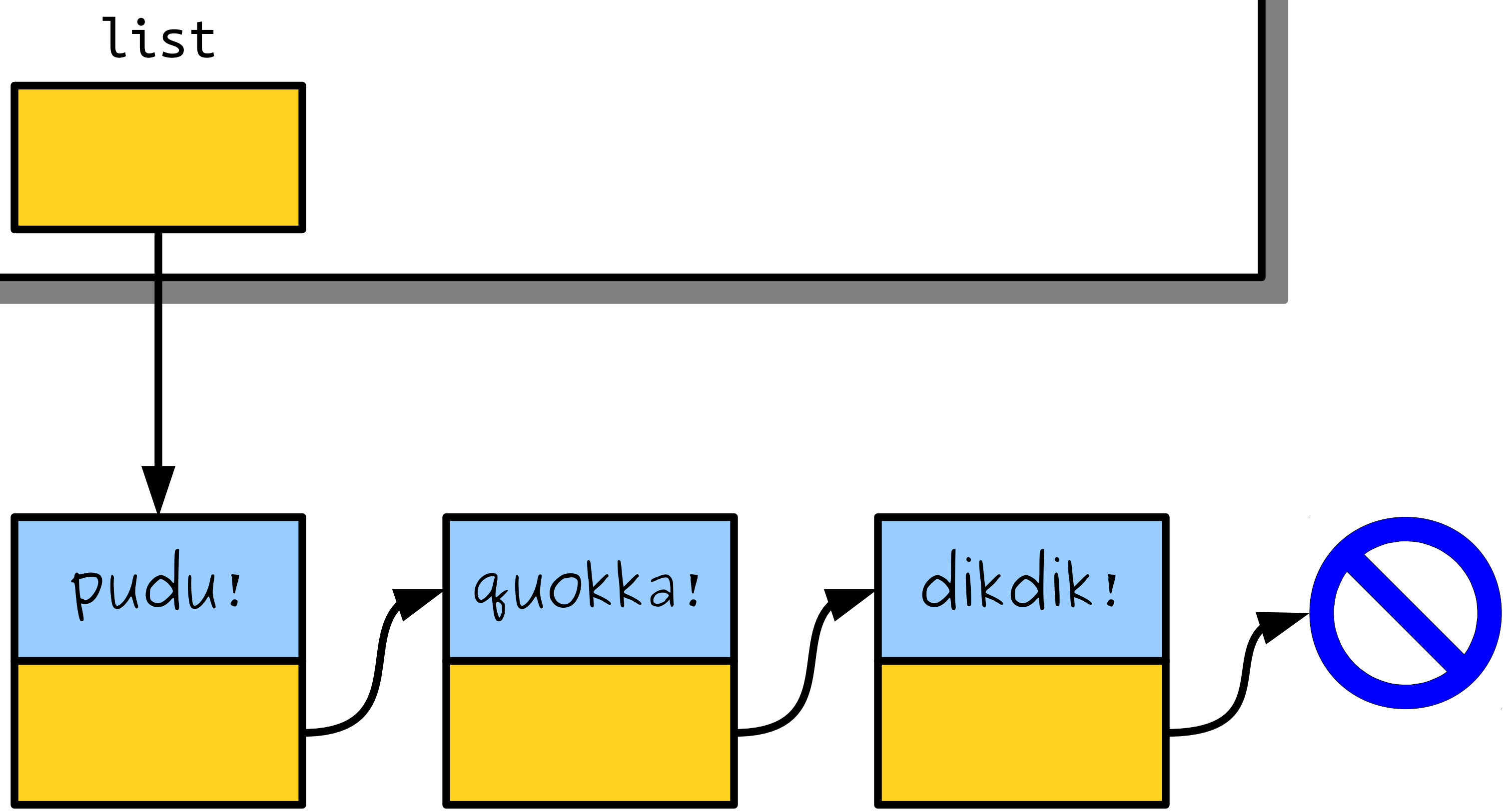
```
void deleteList(Cell* list) {
    // WRONG WRONG WRONG WRONG
    // WRONG WRONG WRONG WRONG

    while (list != nullptr) {
        delete list;
        list = list->next;
    }
}
```



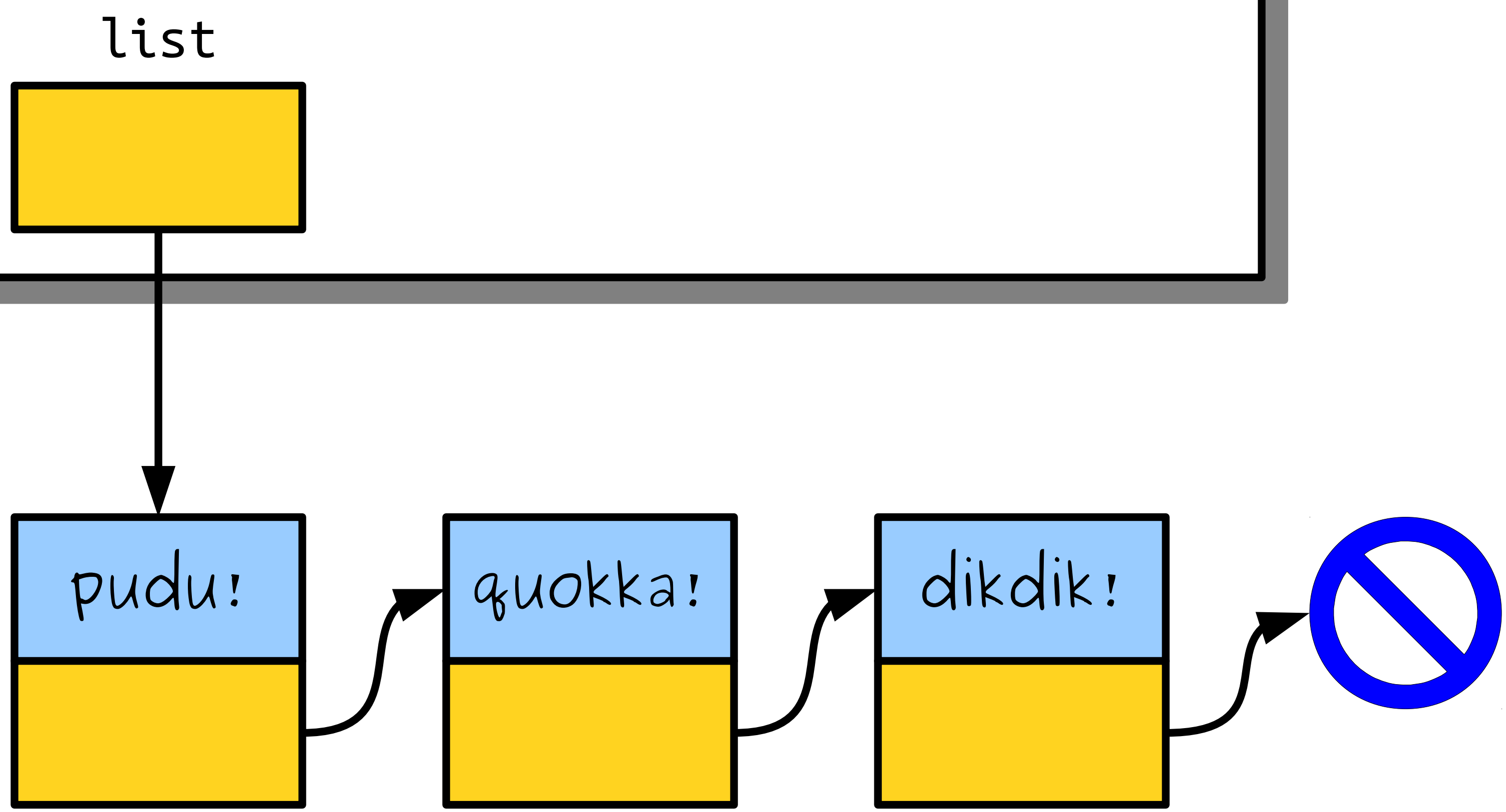
```
void deleteList(Cell* list) {
    while (list != nullptr) {

        delete list;
        list = list->next;
    }
}
```



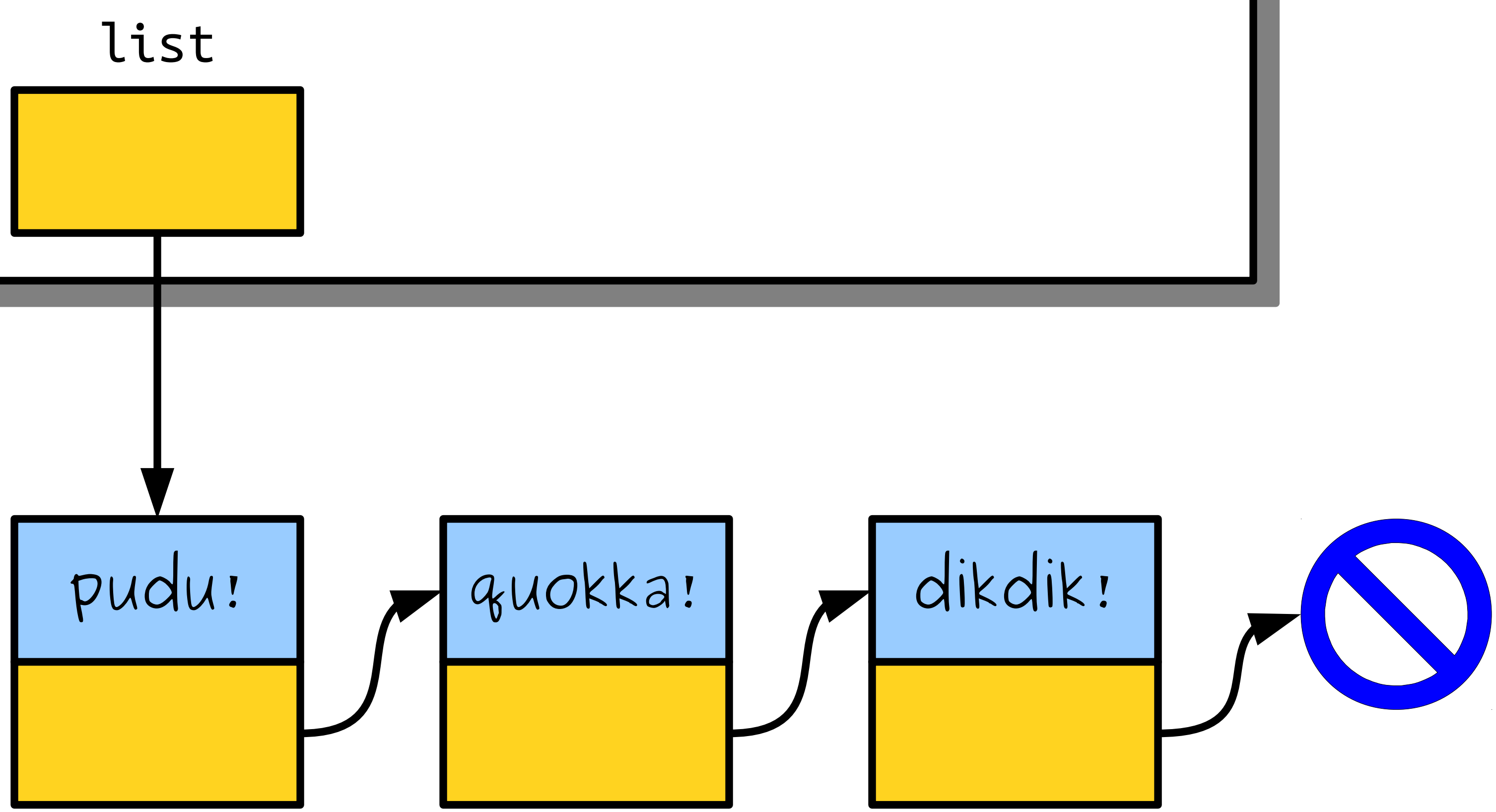
```
void deleteList(Cell* list) {
    while (list != nullptr) {
        Cell* next = list->next;
        delete list;
        list = list->next;
    }
}
```



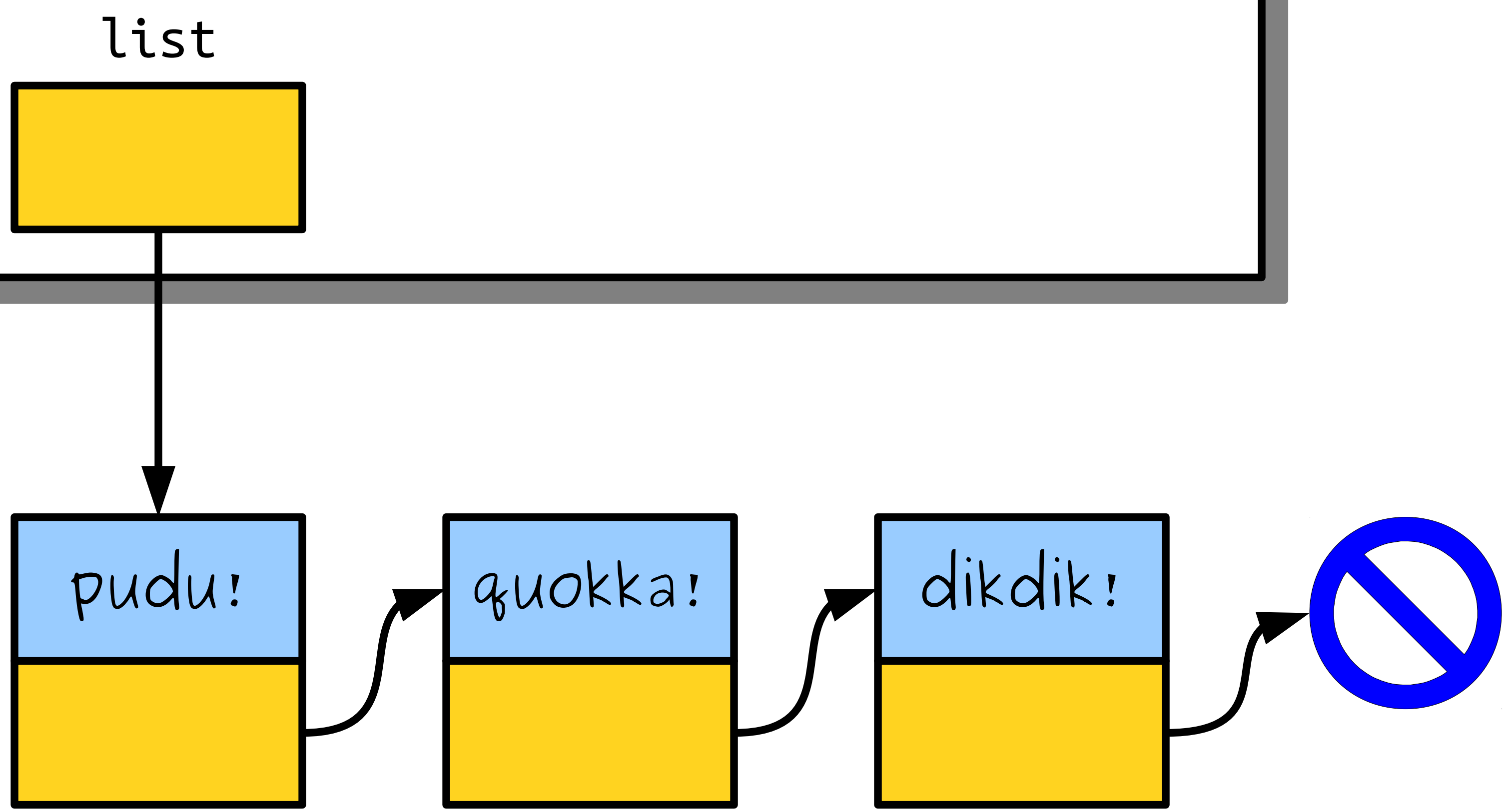
```
void deleteList(Cell* list) {
    while (list != nullptr) {
        Cell* next = list->next;
        delete list;
        list = next;
    }
}
```



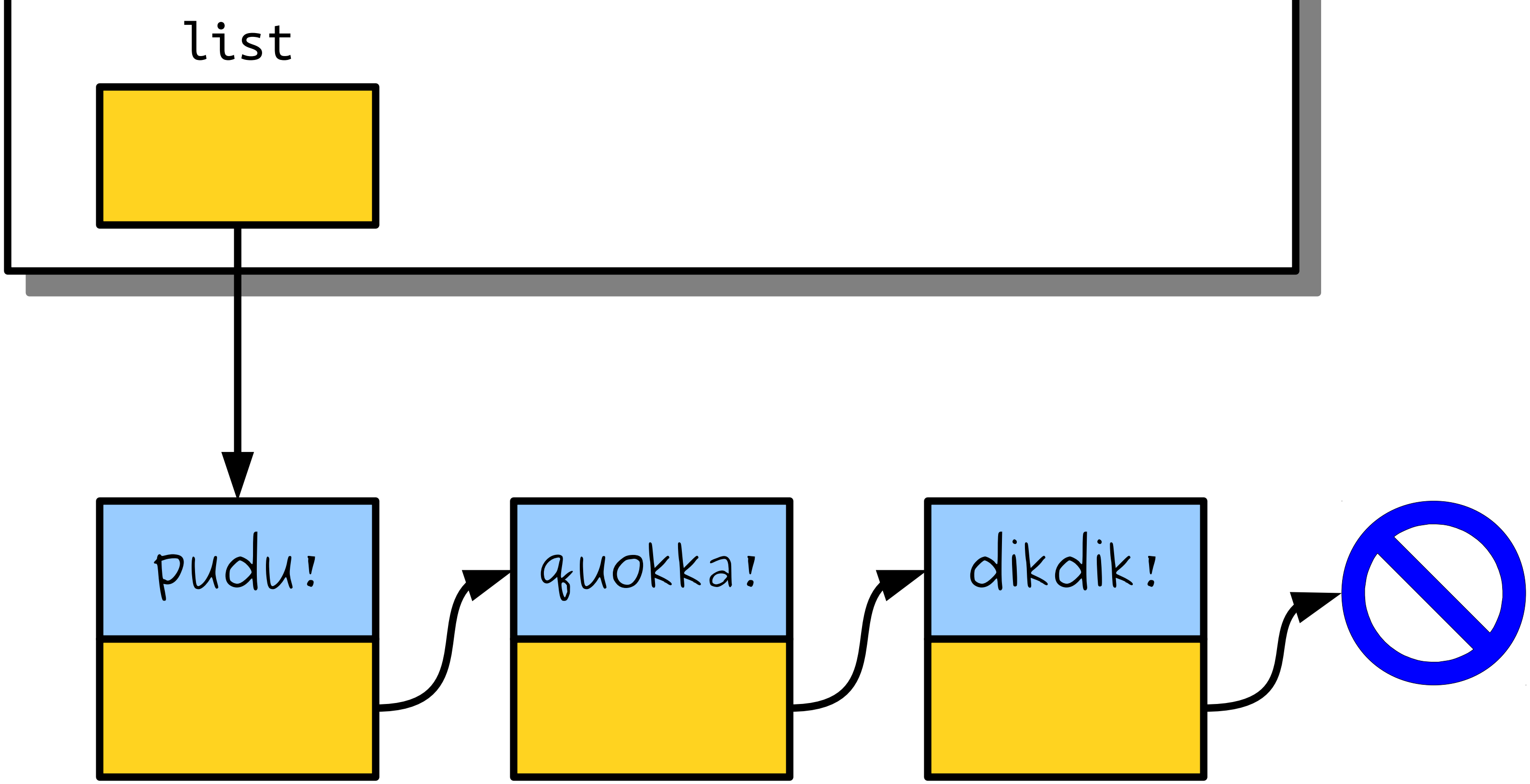
```
void deleteList(Cell* list) {
   while (list != nullptr) {
      Cell* next = list->next;
      delete list;
      list = next;
   }
}
```



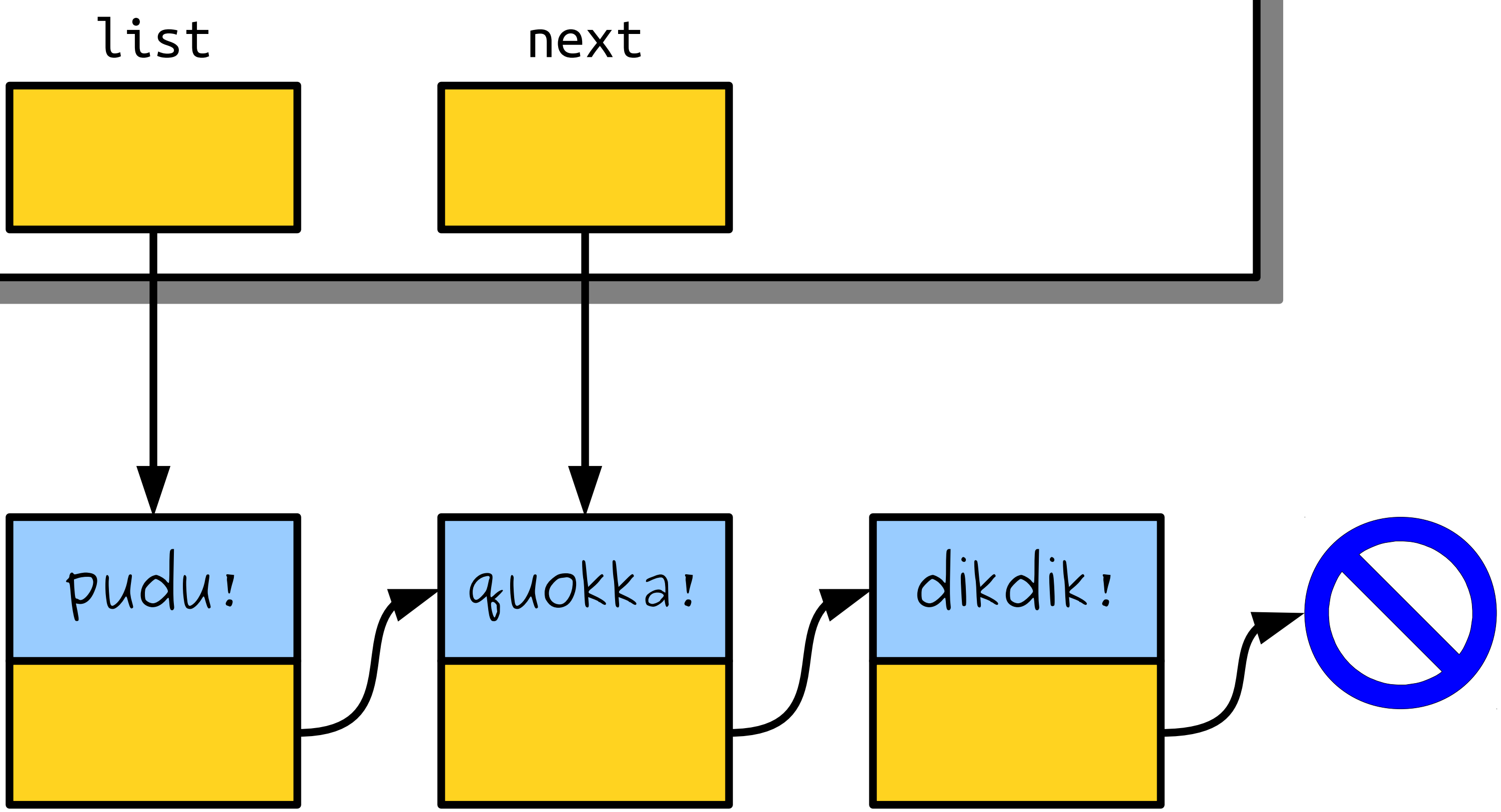
```
void deleteList(Cell* list) {
    while (list != nullptr) {
        Cell* next = list->next;
        delete list;
        list = next;
    }
}
```



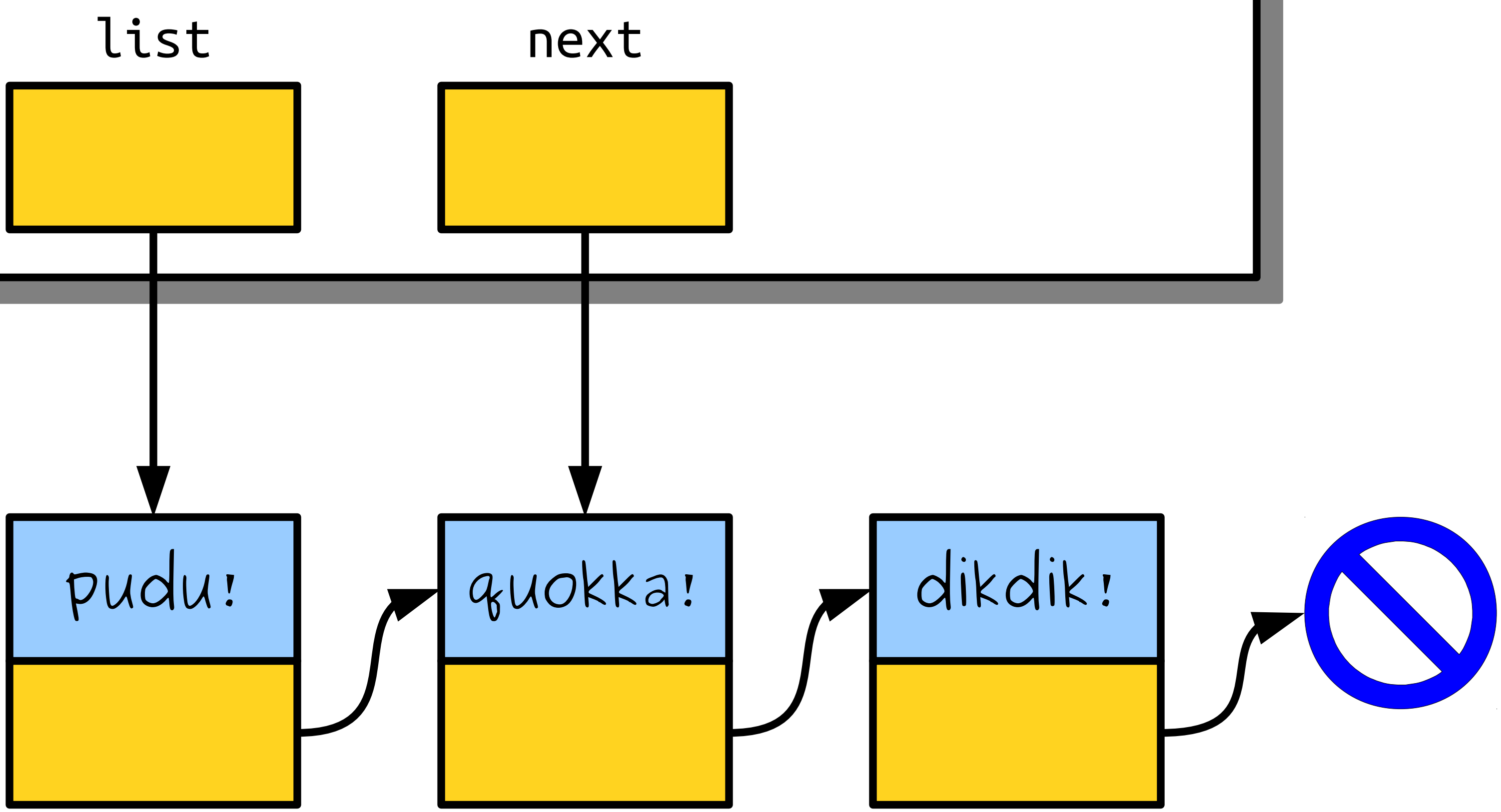
```
void deleteList(Cell* list) {
    while (list != nullptr) {
        Cell* next = list->next;
        delete list;
        list = next;
    }
}
```



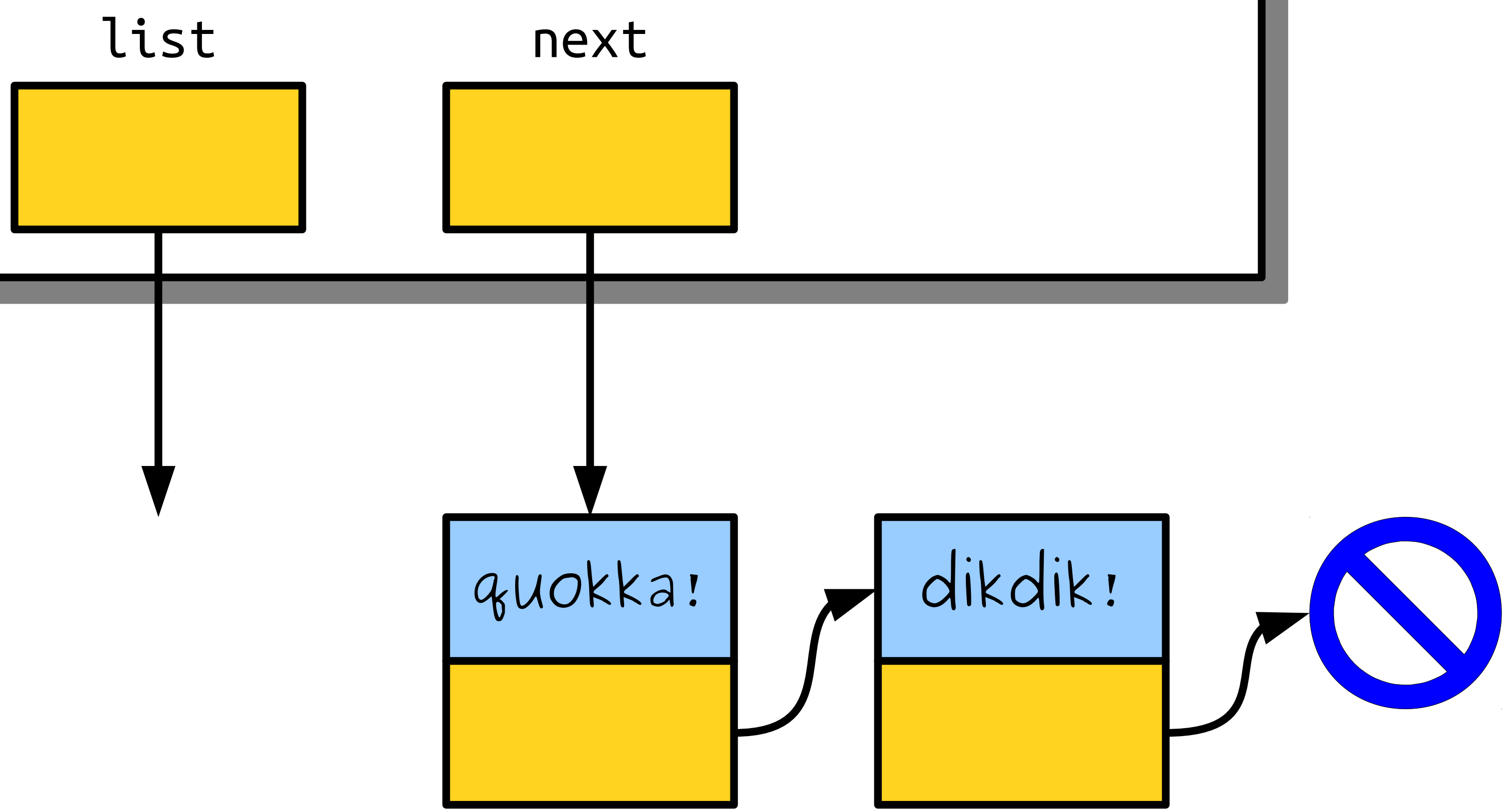
```
void deleteList(Cell* list) {
    while (list != nullptr) {
        Cell* next = list->next;
        delete list;
        list = next;
    }
}
```



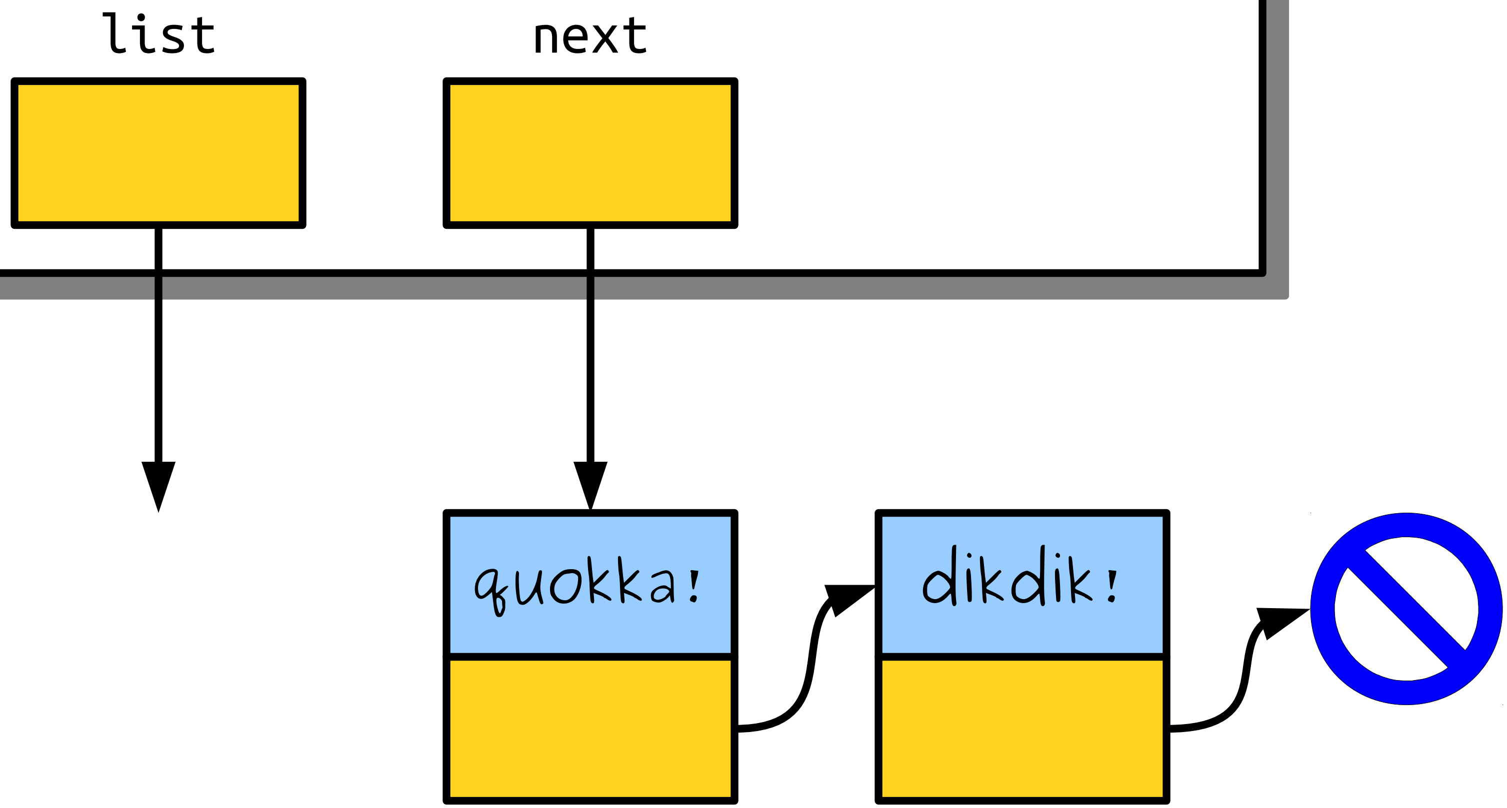
```
void deleteList(Cell* list) {
    while (list != nullptr) {
        Cell* next = list->next;
        delete list;
        list = next;
    }
}
```



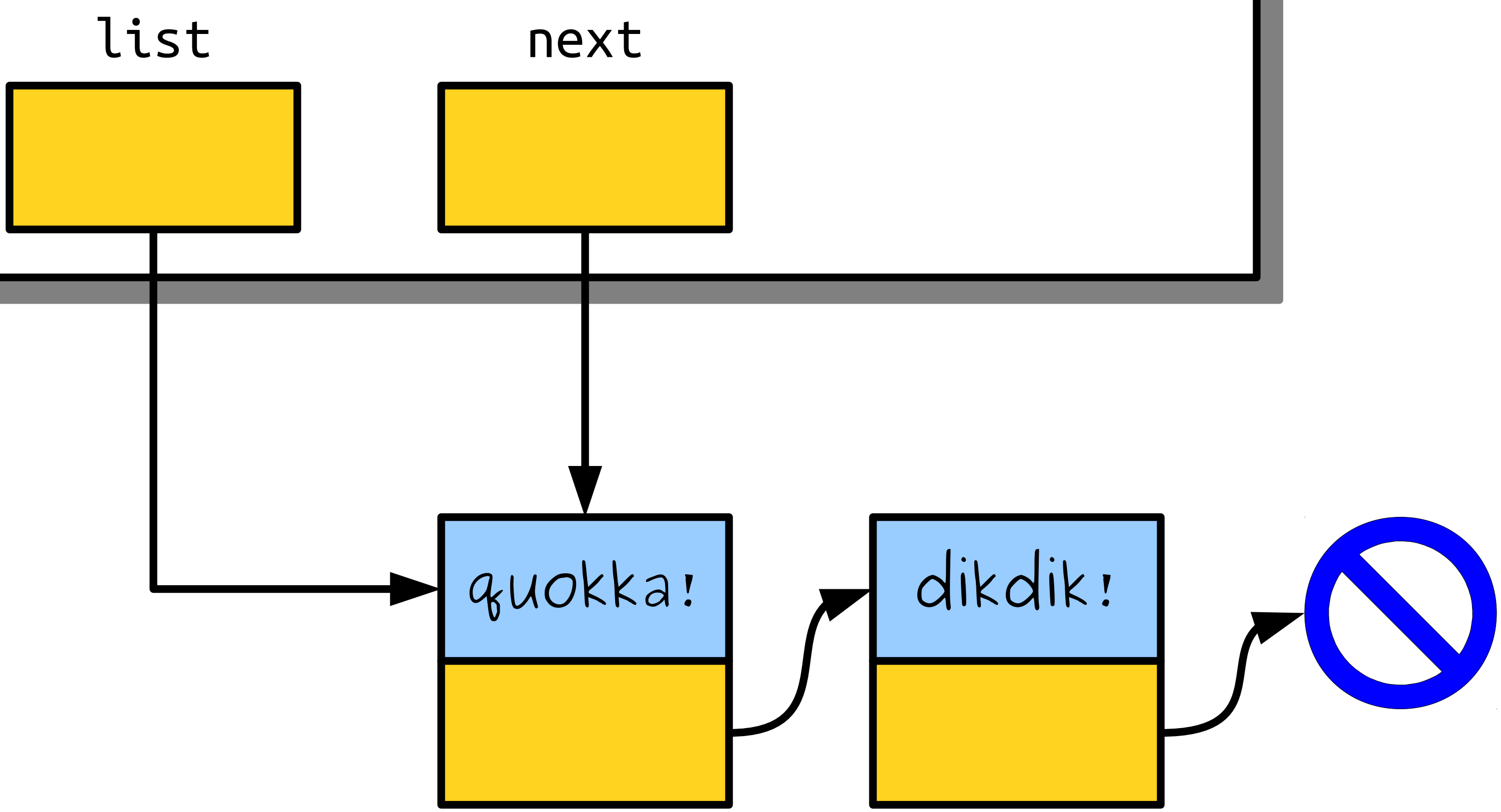
```
void deleteList(Cell* list) {
    while (list != nullptr) {
        Cell* next = list->next;
        delete list;
        list = next;
    }
}
```



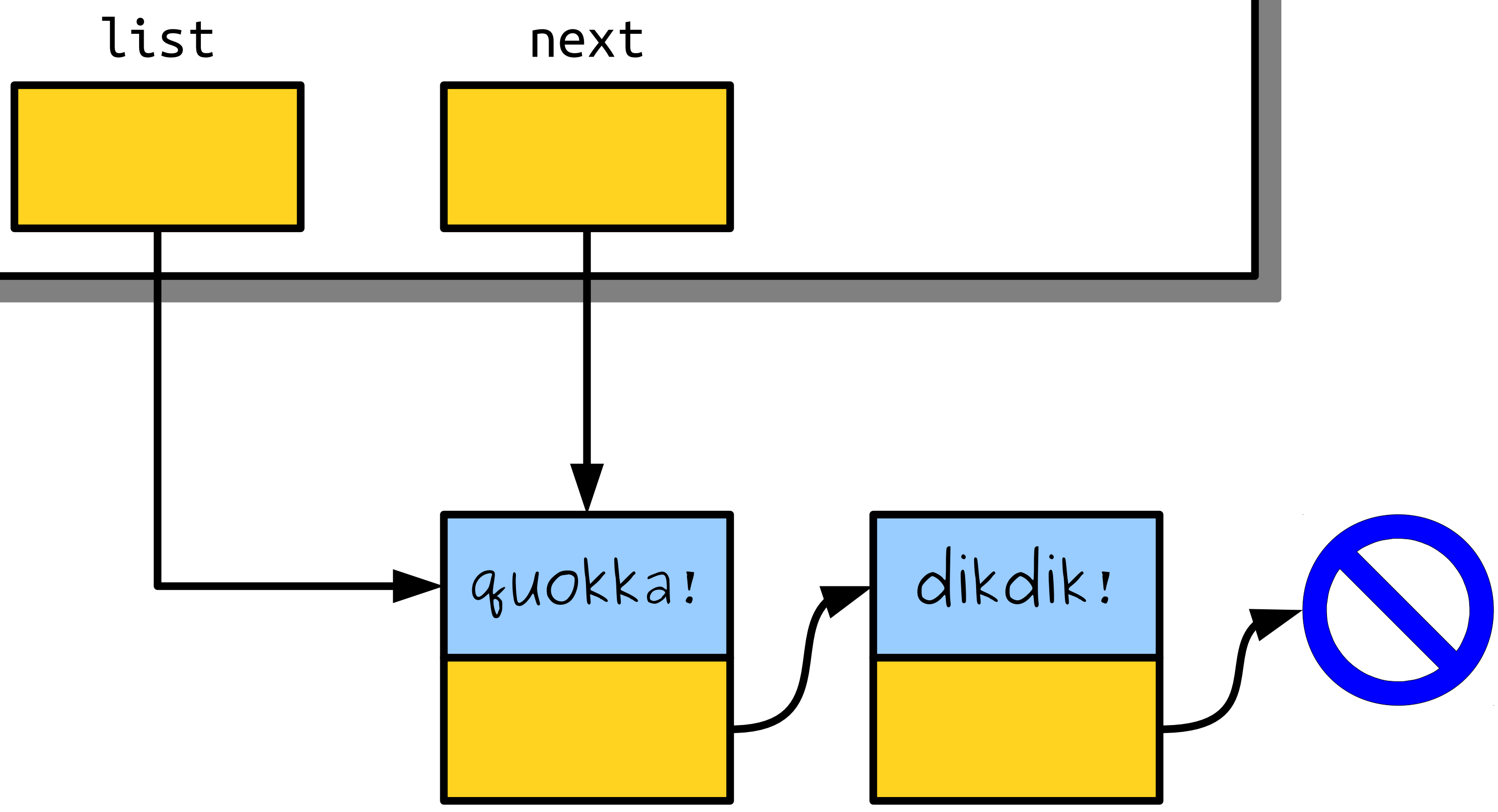
```
void deleteList(Cell* list) {
    while (list != nullptr) {
        Cell* next = list->next;
        delete list;
        list = next;
    }
}
```



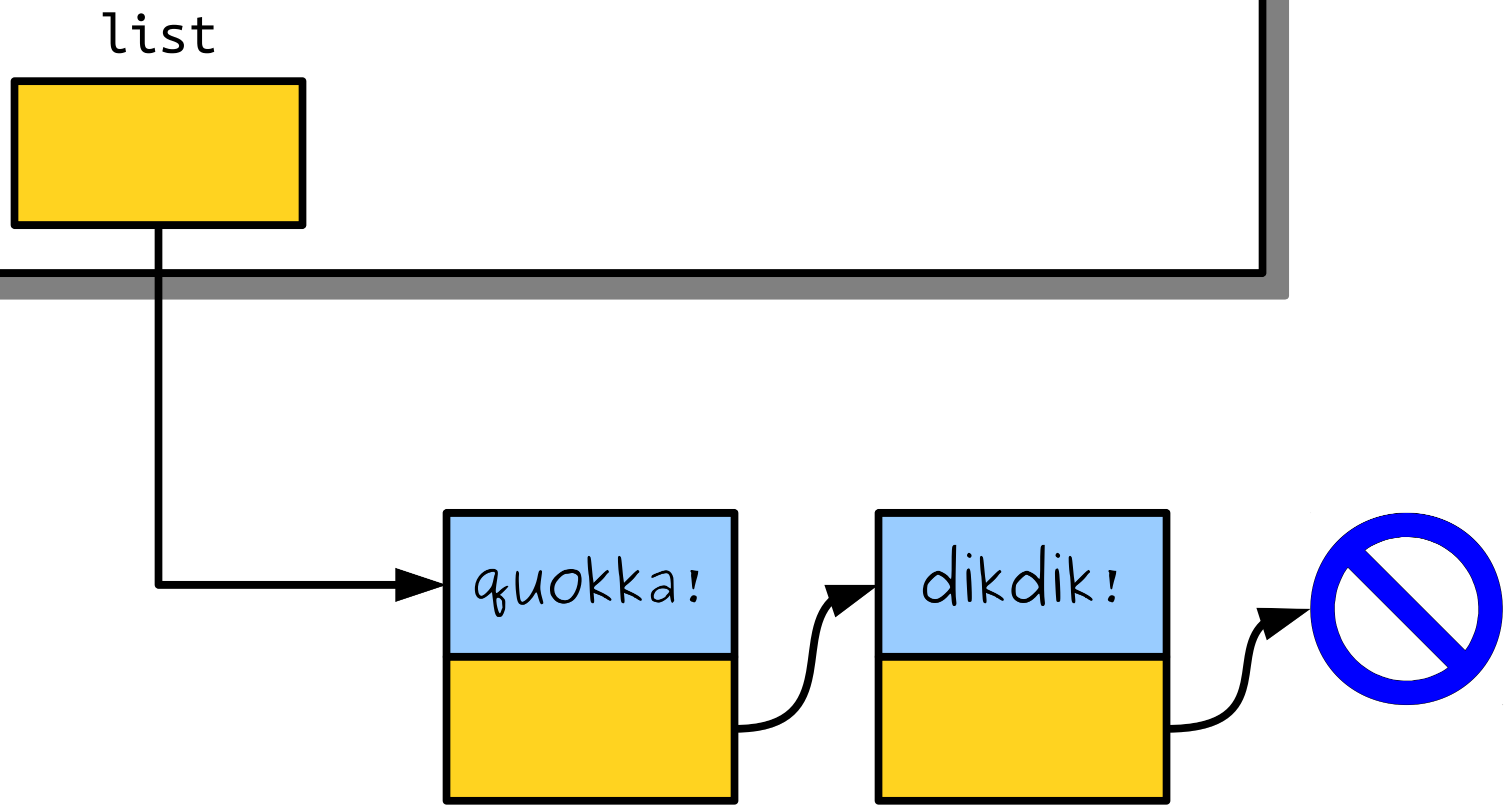
```
void deleteList(Cell* list) {
    while (list != nullptr) {
        Cell* next = list->next;
        delete list;
        list = next;
    }
}
```



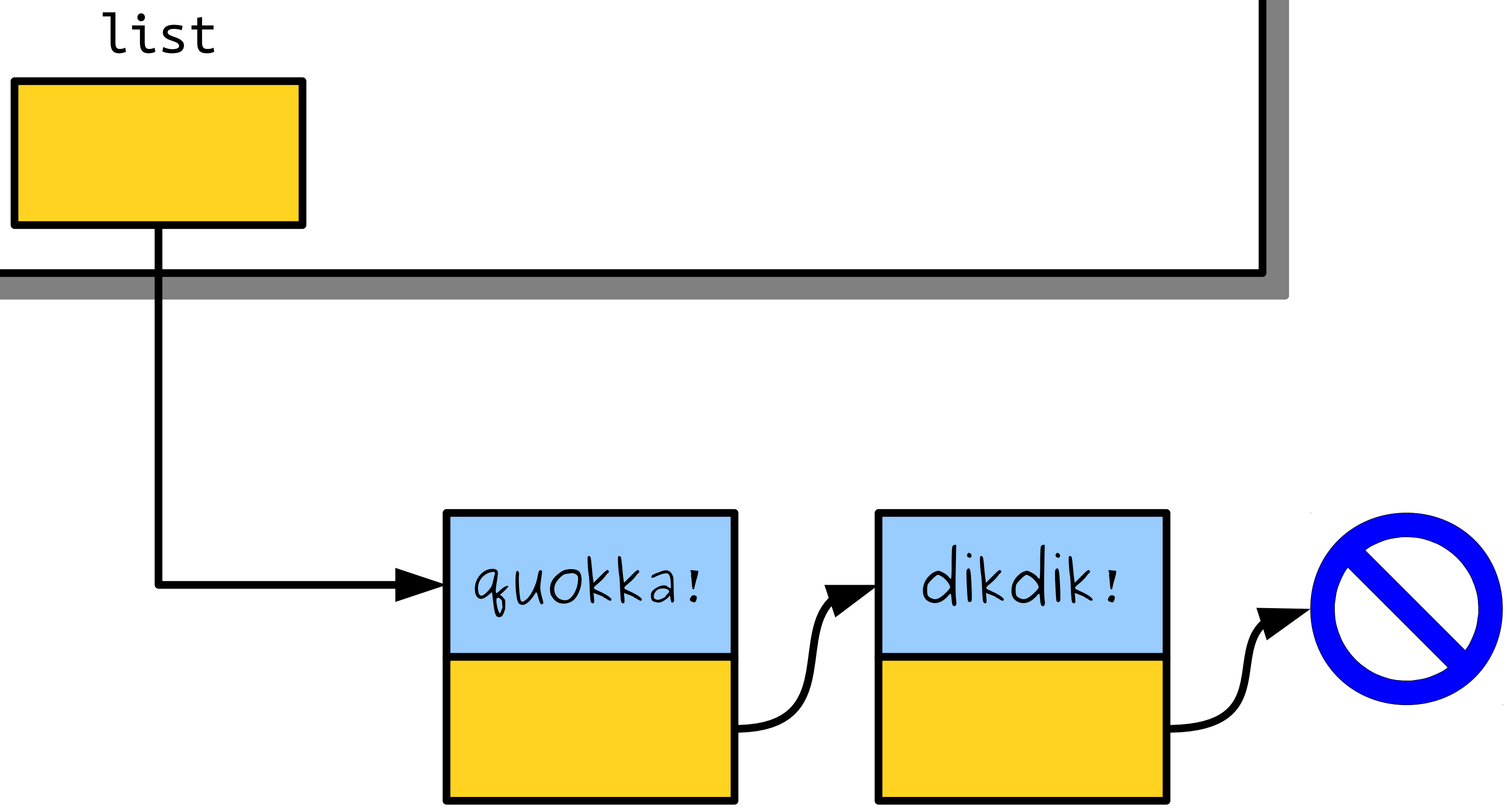
```
void deleteList(Cell* list) {
   while (list != nullptr) {
      Cell* next = list->next;
      delete list;
      list = next;
   }
}
```



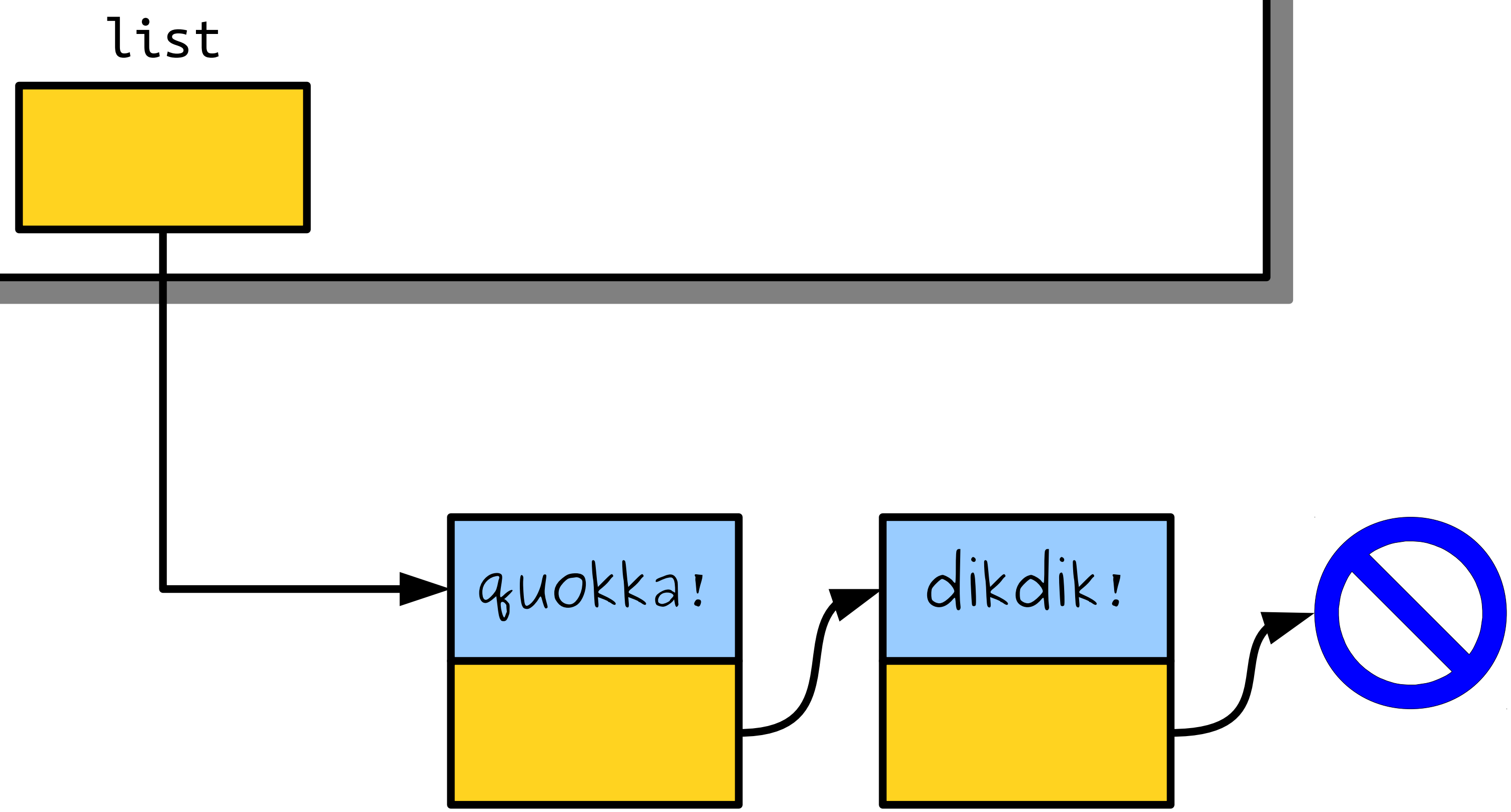
```
void deleteList(Cell* list) {
   while (list != nullptr) {
      Cell* next = list->next;
      delete list;
      list = next;
   }
}
```



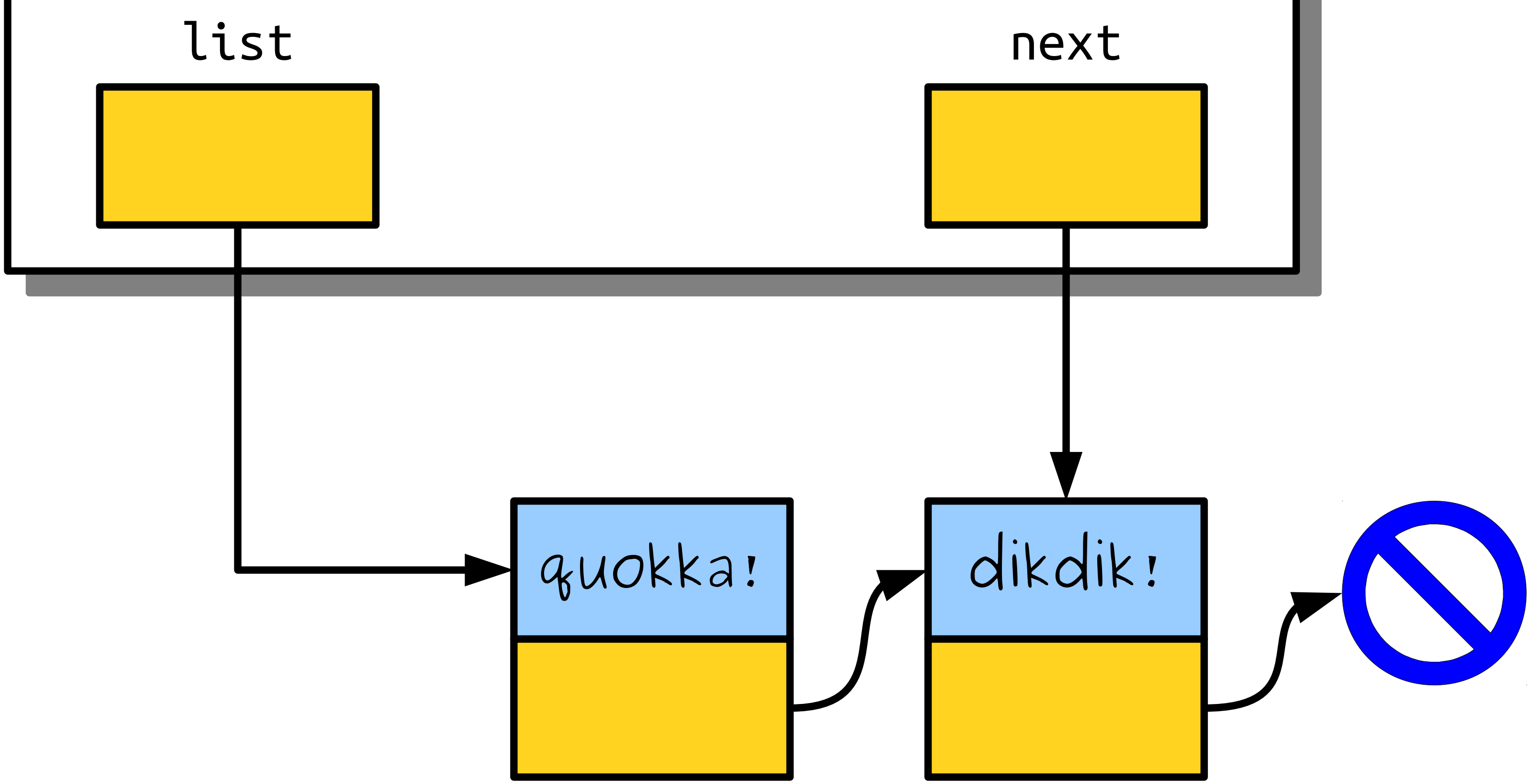
```
void deleteList(Cell* list) {
    while (list != nullptr) {
        Cell* next = list->next;
        delete list;
        list = next;
    }
}
```

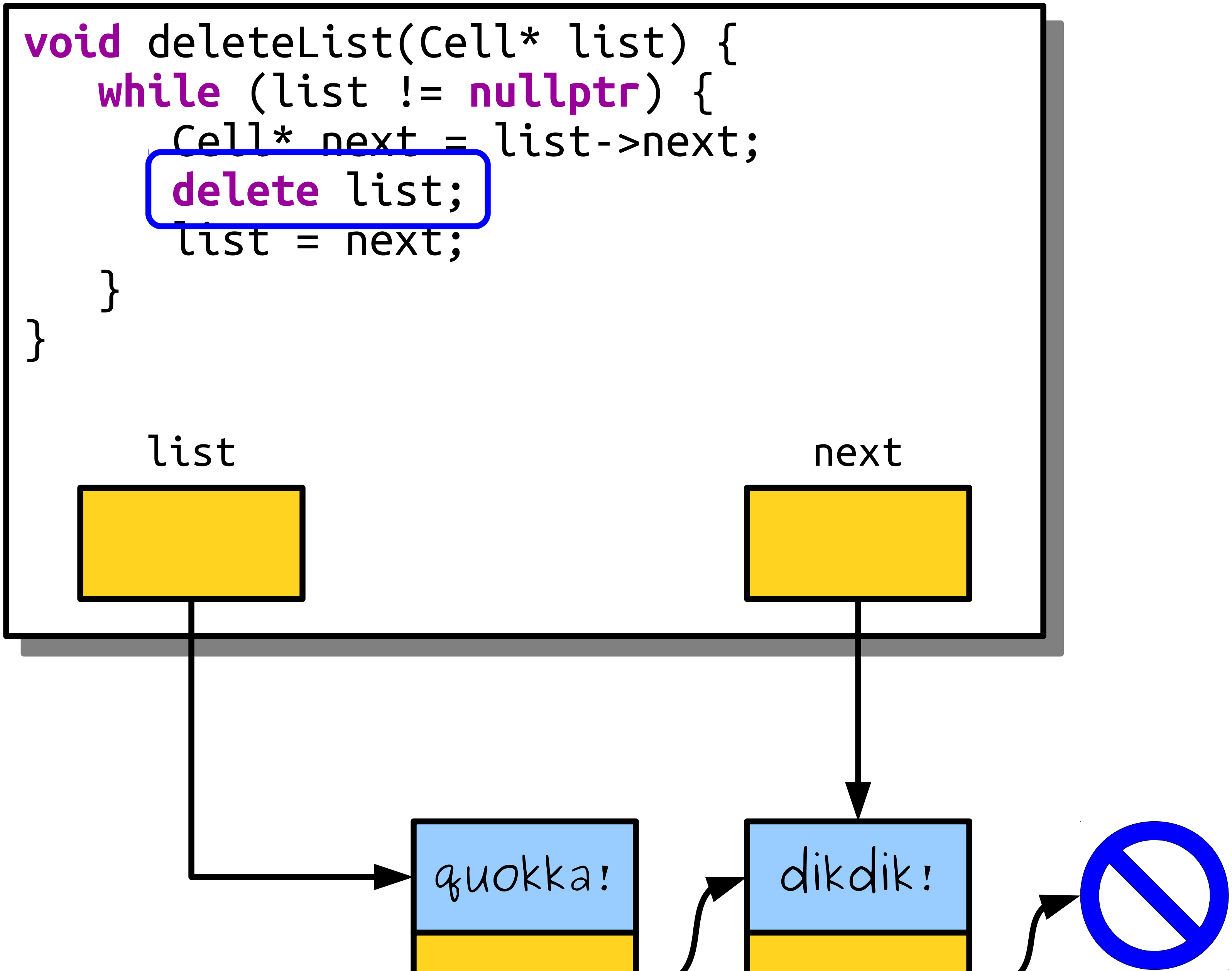


```
void deleteList(Cell* list) {
    while (list != nullptr) {
        Cell* next = list->next;
        delete list;
        list = next;
    }
}
```

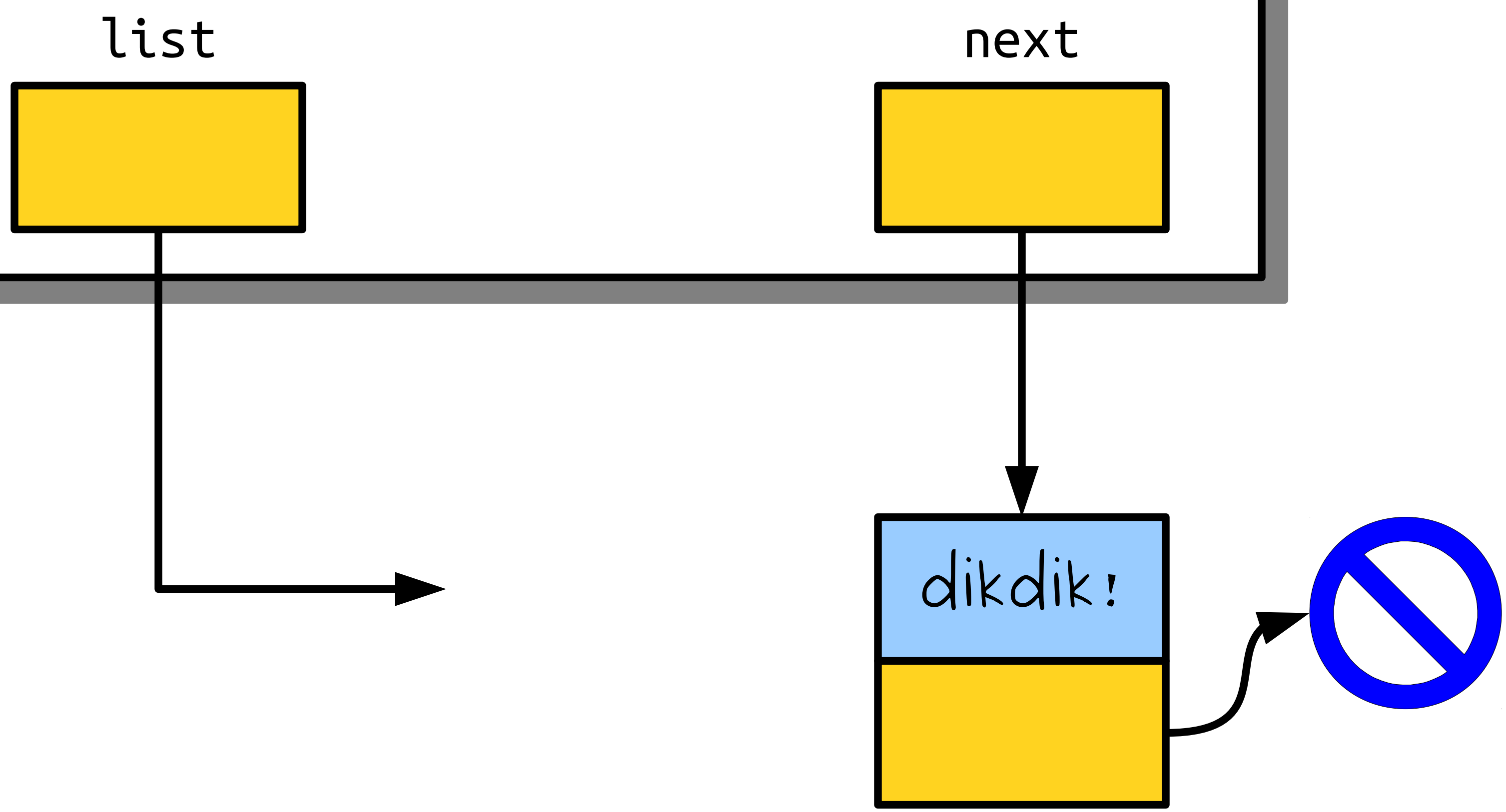


```
void deleteList(Cell* list) {
    while (list != nullptr) {
        Cell* next = list->next;
        delete list;
        list = next;
    }
}
```

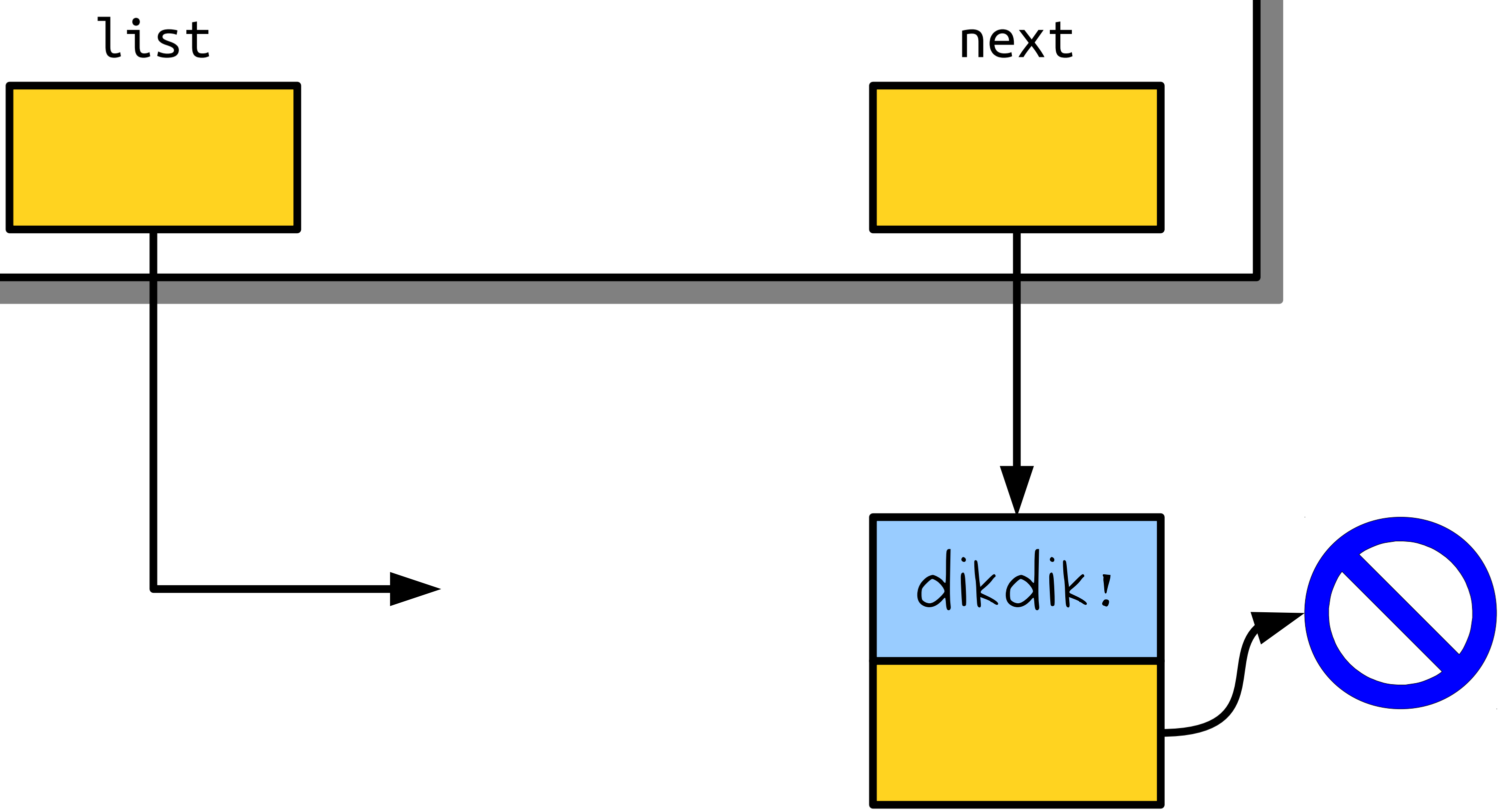



void deleteList(Cell* list) {
while (list != nullptr) {
Cell* next = list->next;
delete list;
list = next;
}
}
list
next
quokka!
dikdik!

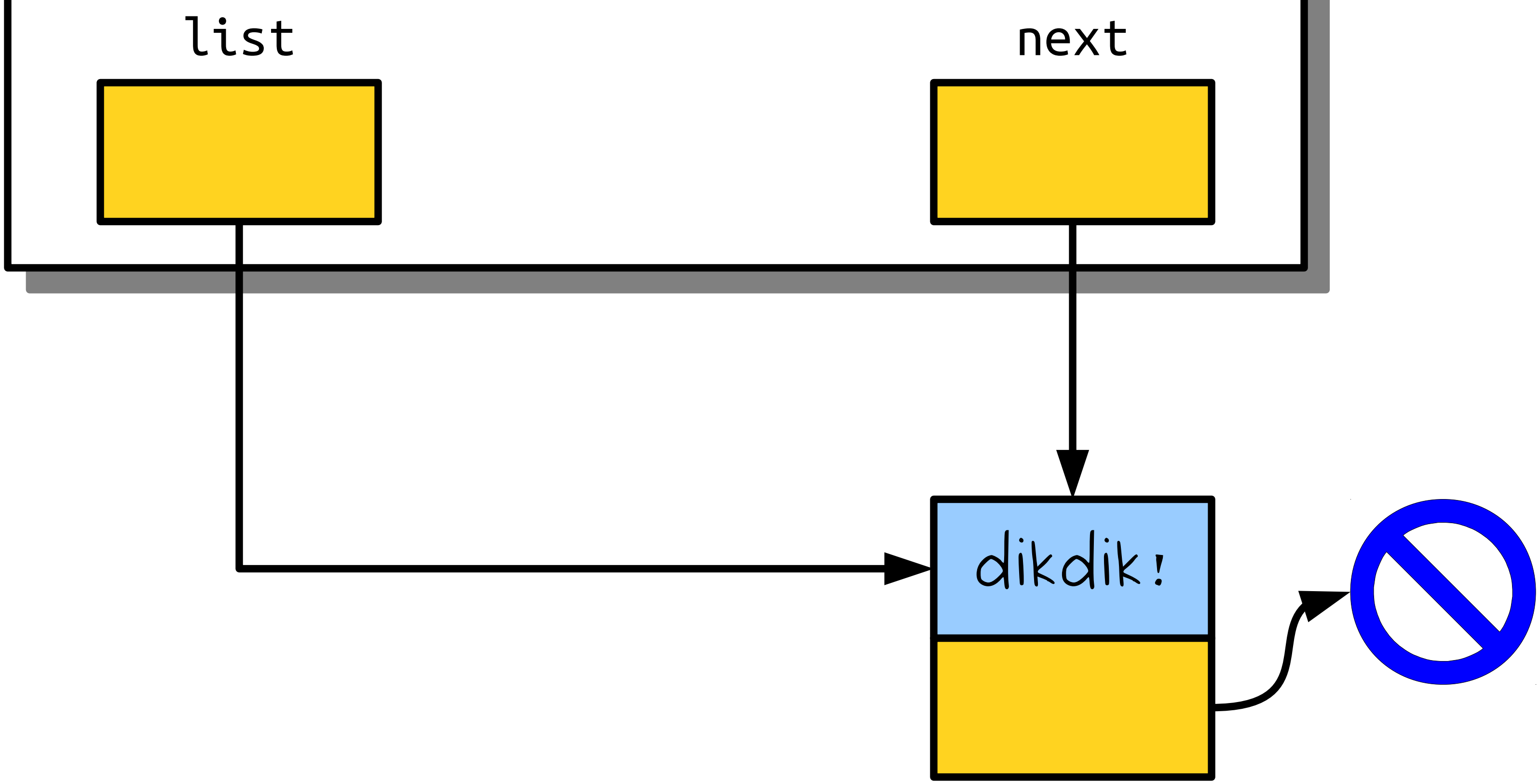
```
void deleteList(Cell* list) {
    while (list != nullptr) {
        Cell* next = list->next;
        delete list;
        list = next;
    }
}
```



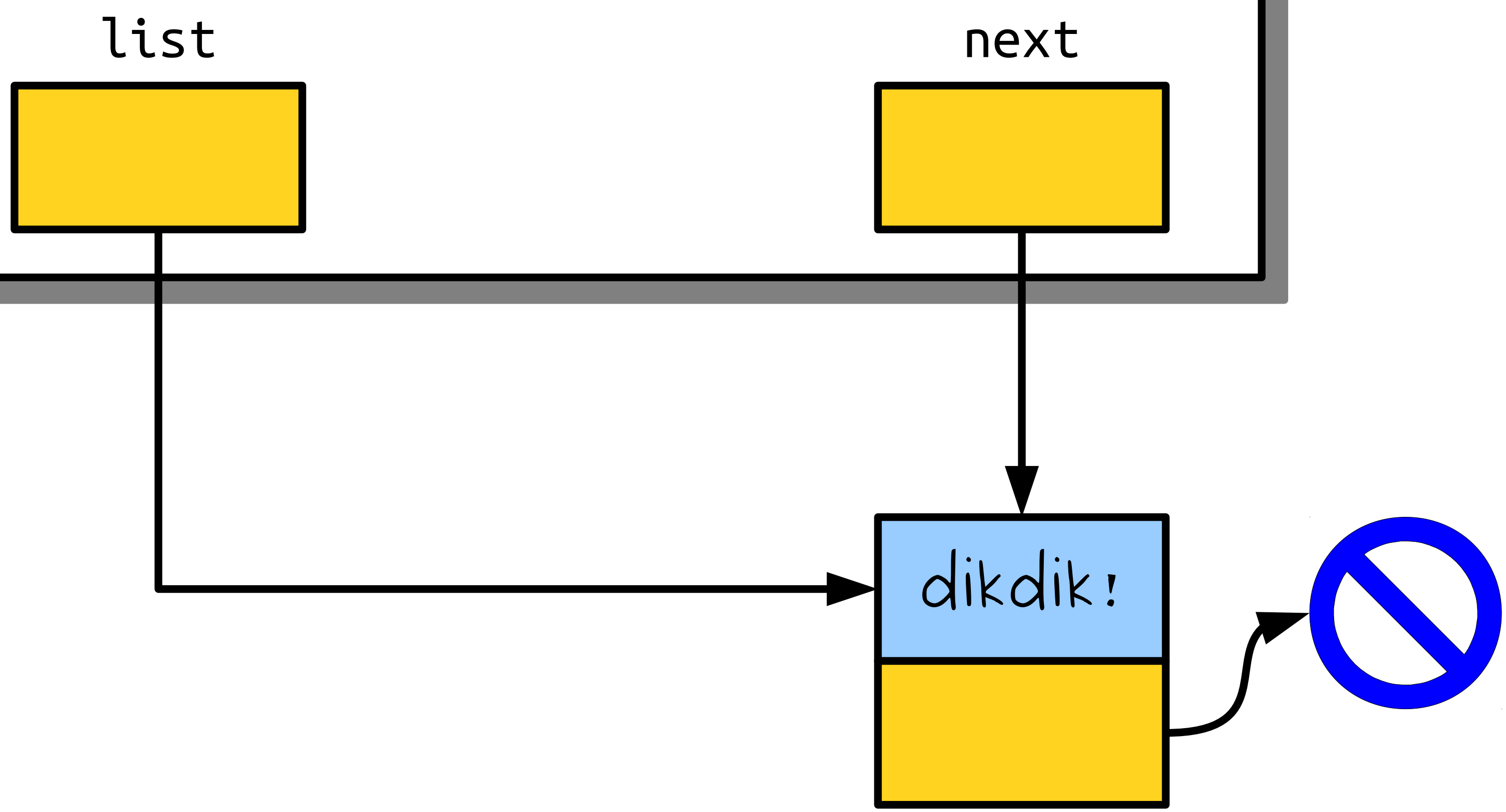
```
void deleteList(Cell* list) {
    while (list != nullptr) {
        Cell* next = list->next;
        delete list;
        list = next;
    }
}
```



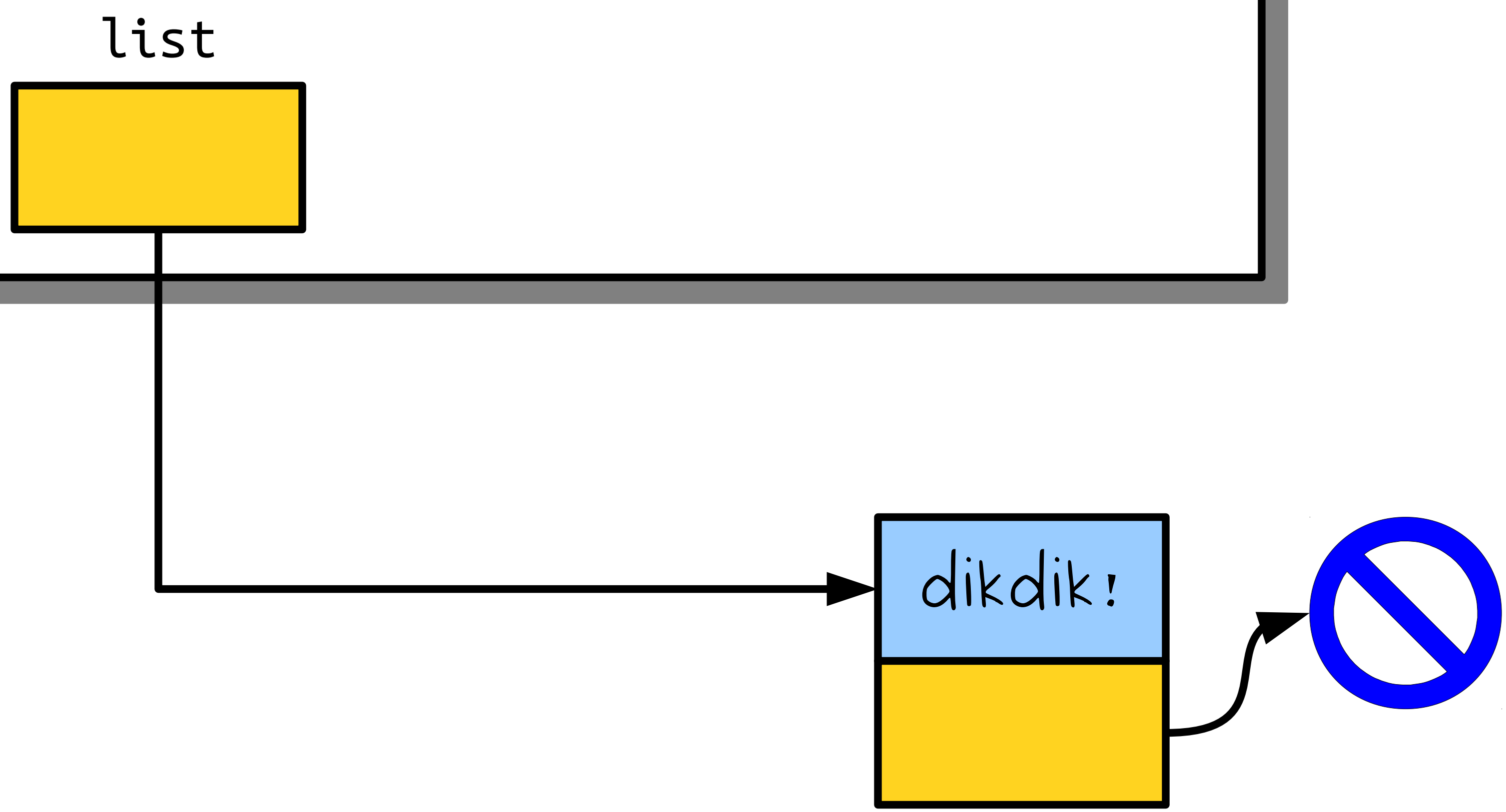
```
void deleteList(Cell* list) {
    while (list != nullptr) {
        Cell* next = list->next;
        delete list;
        list = next;
    }
}
```



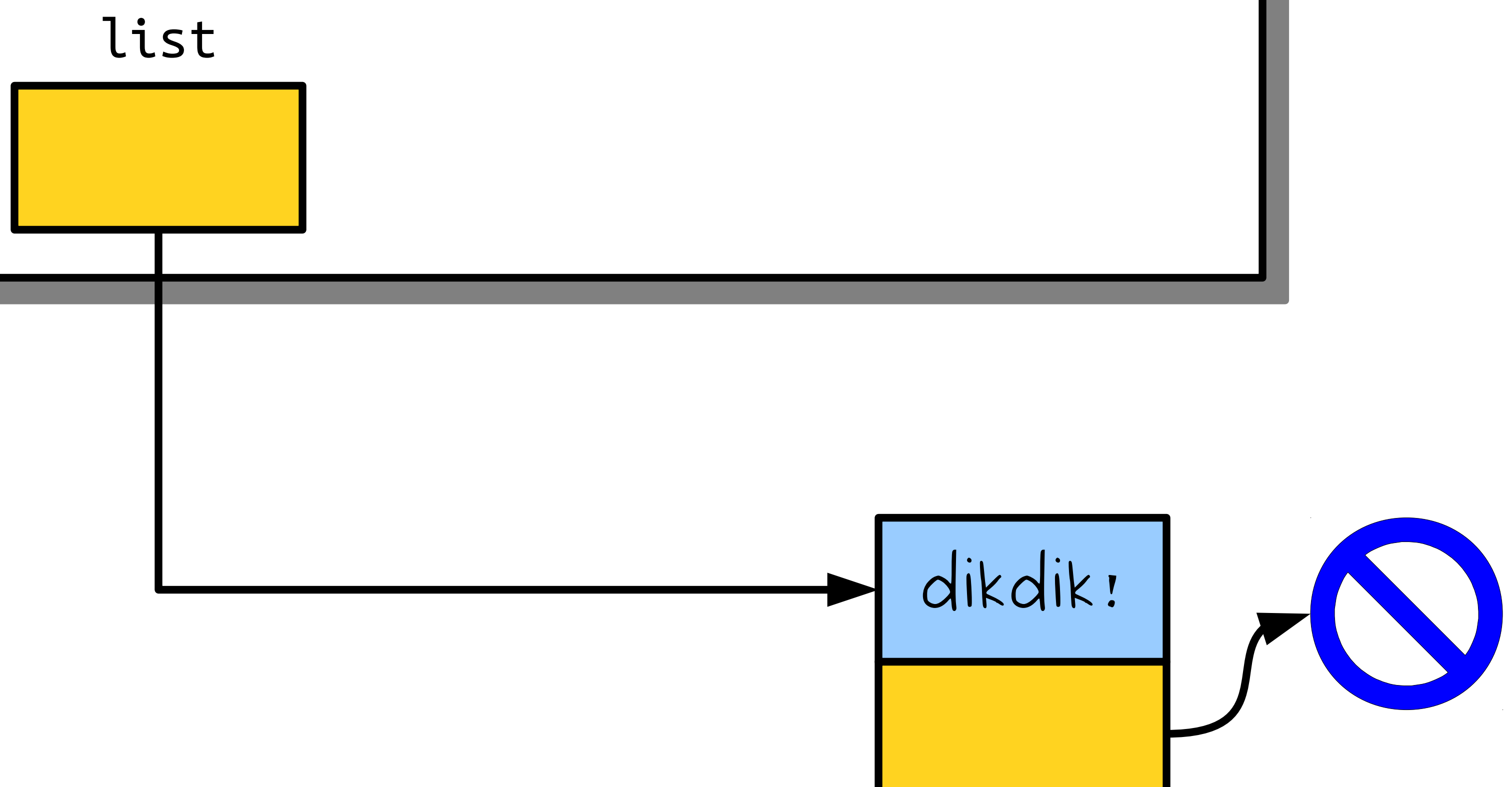
```
void deleteList(Cell* list) {
    while (list != nullptr) {
        Cell* next = list->next;
        delete list;
        list = next;
    }
}
```



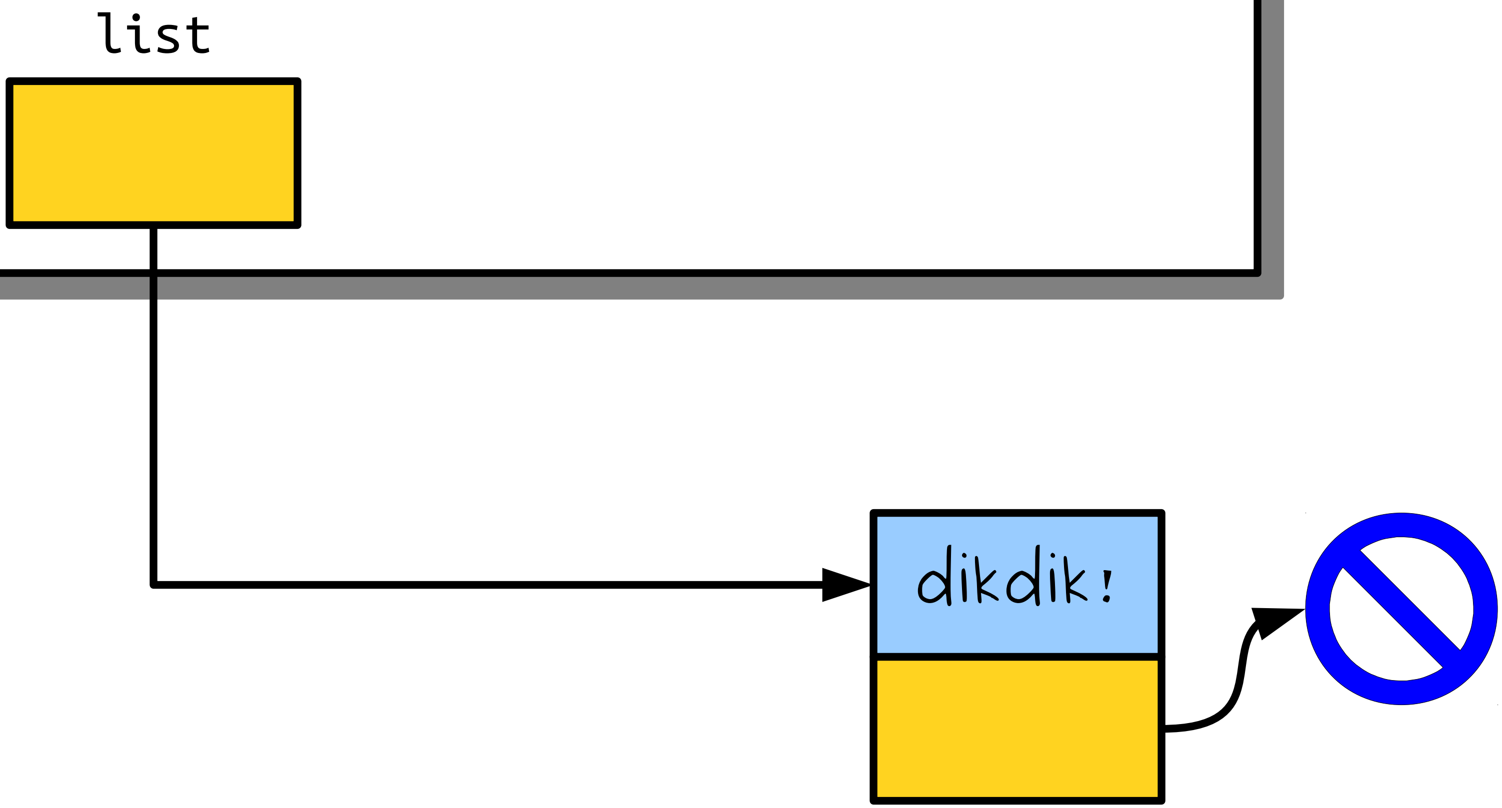
```
void deleteList(Cell* list) {
   while (list != nullptr) {
      Cell* next = list->next;
      delete list;
      list = next;
   }
}
```

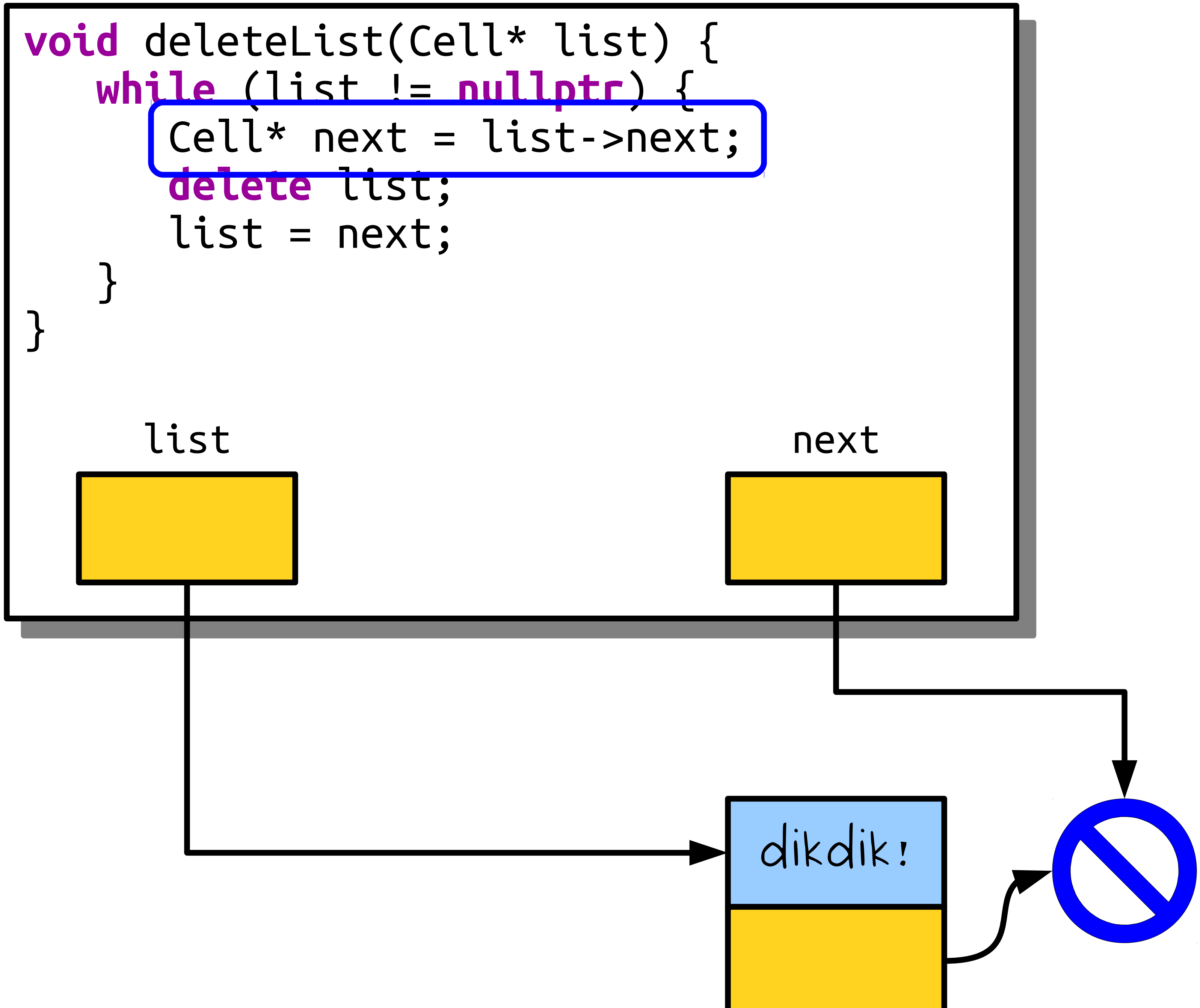


```
void deleteList(Cell* list) {
    while (list != nullptr) {
        Cell* next = list->next;
        delete list;
        list = next;
    }
}
```

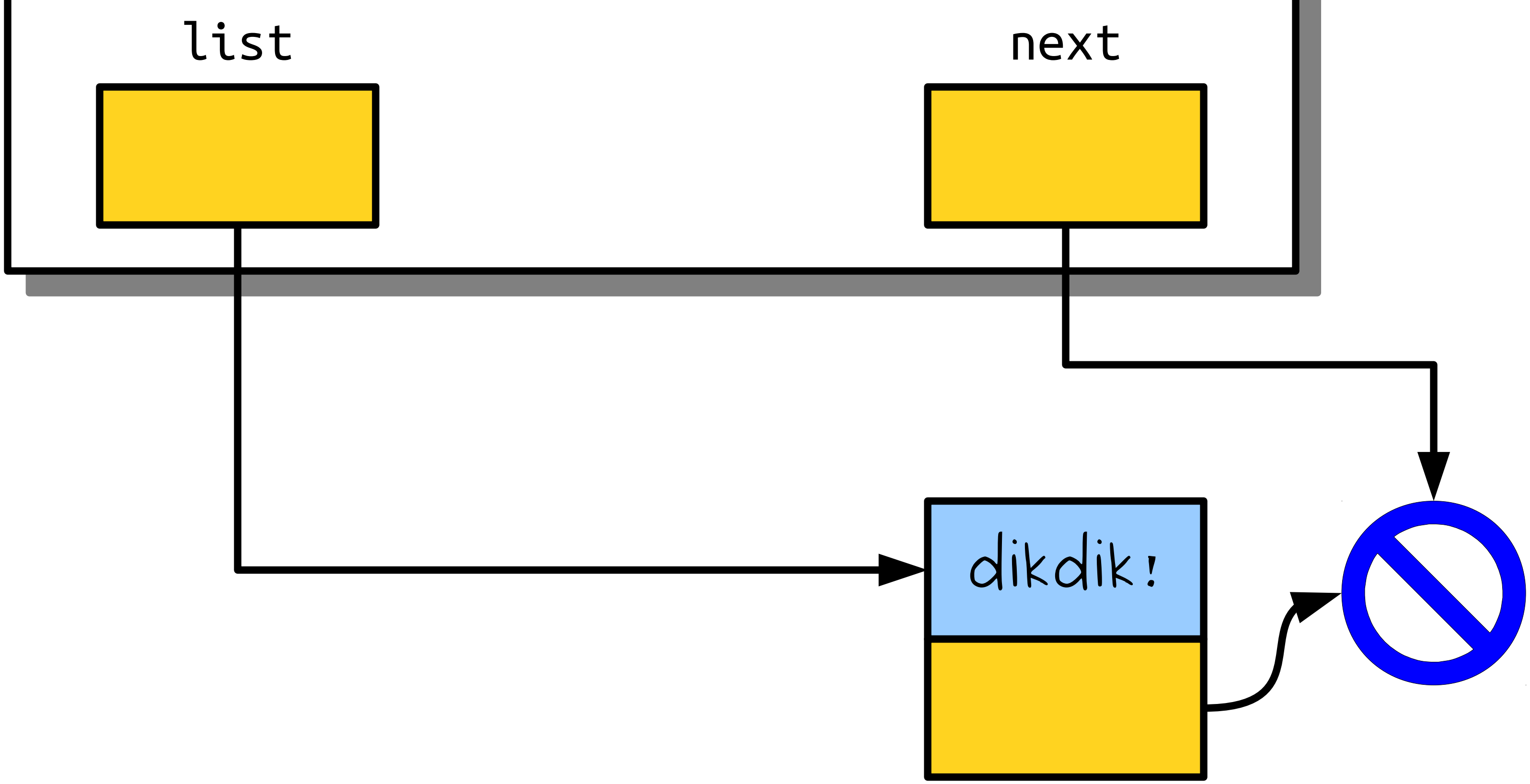


```
void deleteList(Cell* list) {
    while (list != nullptr) {
        Cell* next = list->next;
        delete list;
        list = next;
    }
}
```

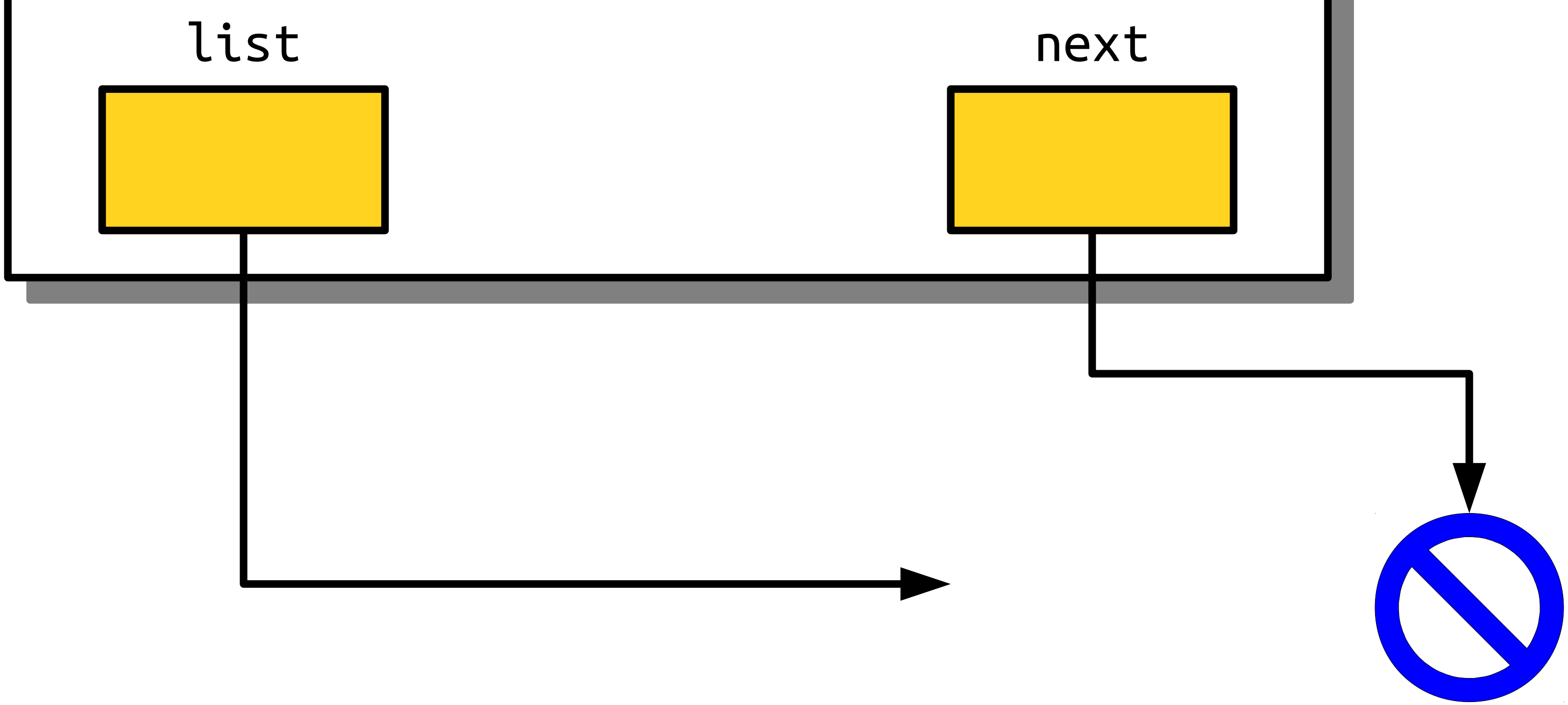



void deleteList(Cell* list) {
while (list != nullptr) {
Cell* next = list->next;
delete list;
list = next;
}
}
list
next
dikdik!

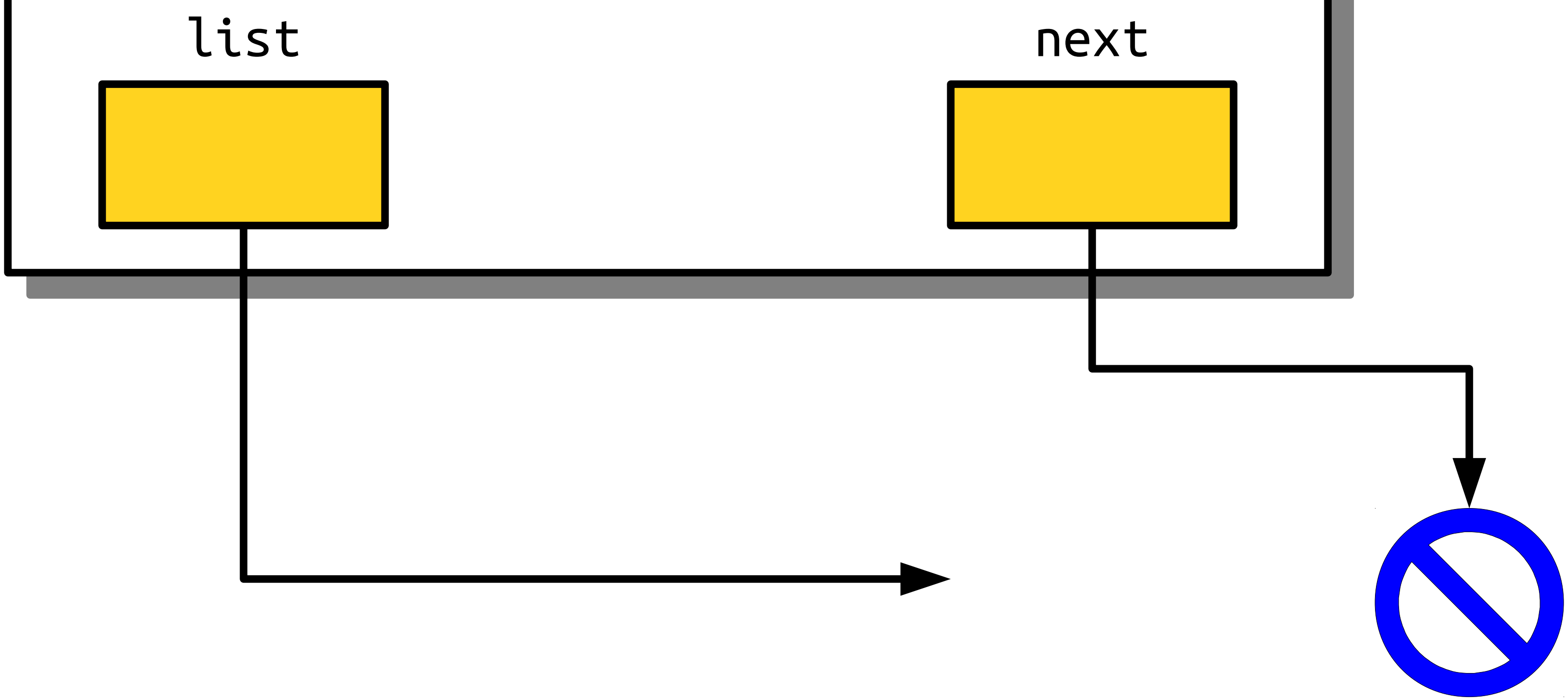
```
void deleteList(Cell* list) {
    while (list != nullptr) {
        Cell* next = list->next;
        delete list;
        list = next;
    }
}
```



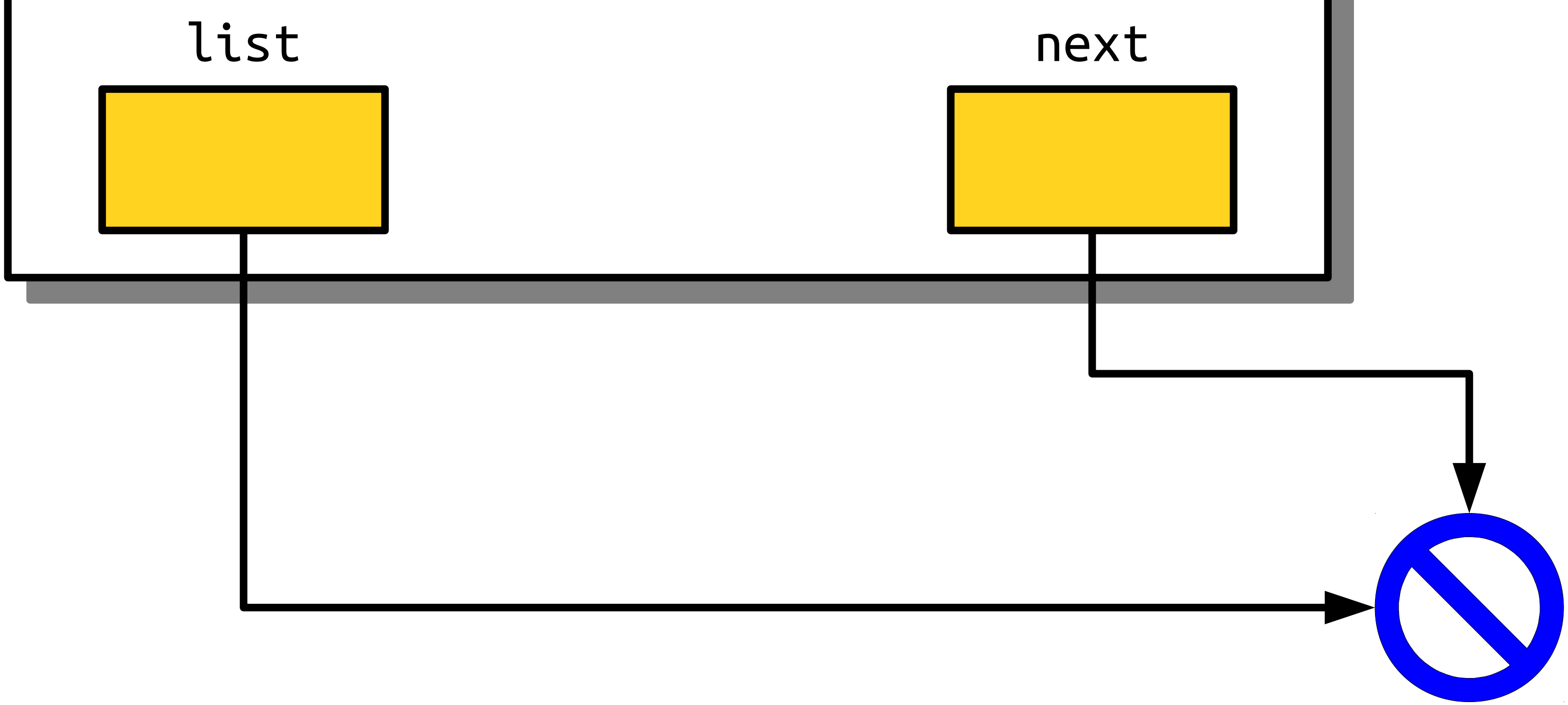
```
void deleteList(Cell* list) {
    while (list != nullptr) {
        Cell* next = list->next;
        delete list;
        list = next;
    }
}
```



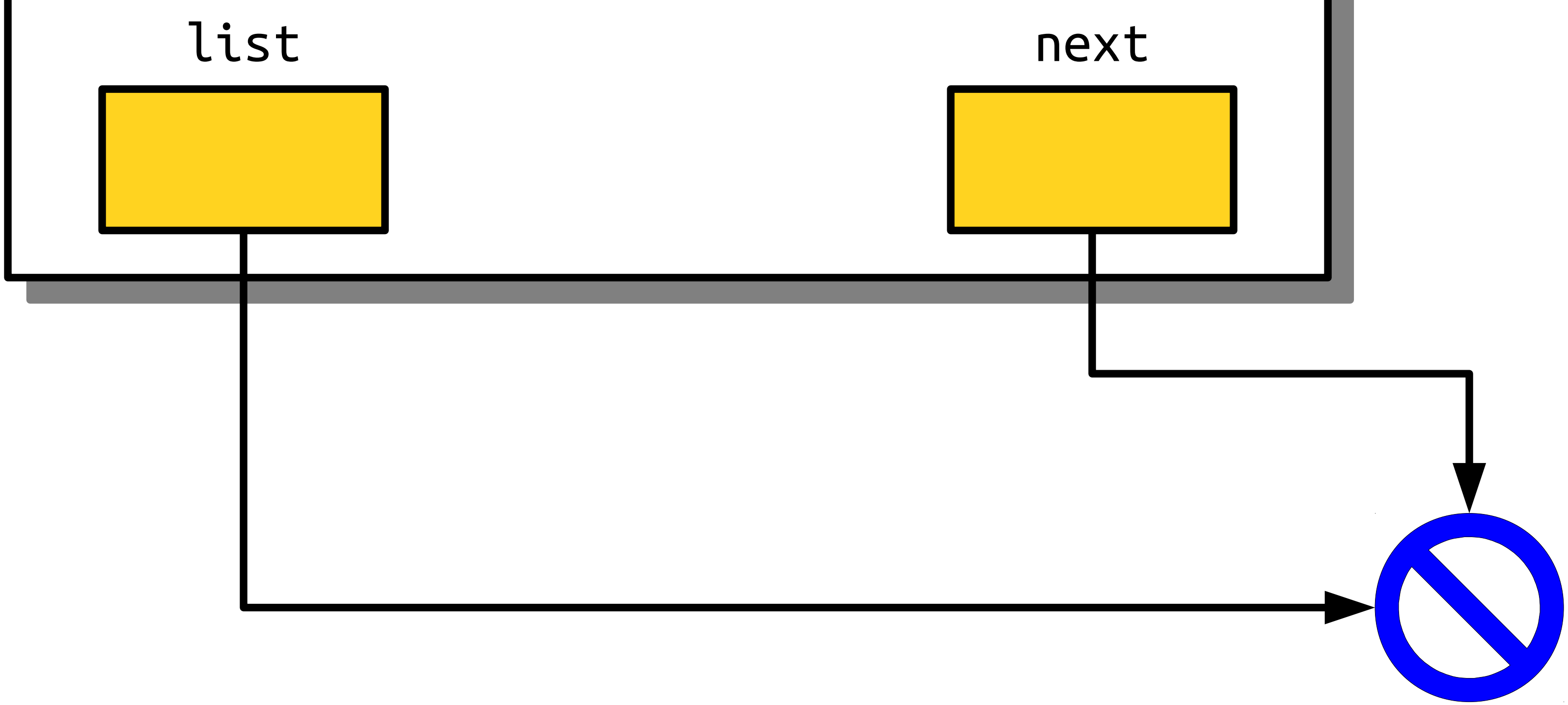
```
void deleteList(Cell* list) {
    while (list != nullptr) {
        Cell* next = list->next;
        delete list;
        list = next;
    }
}
```



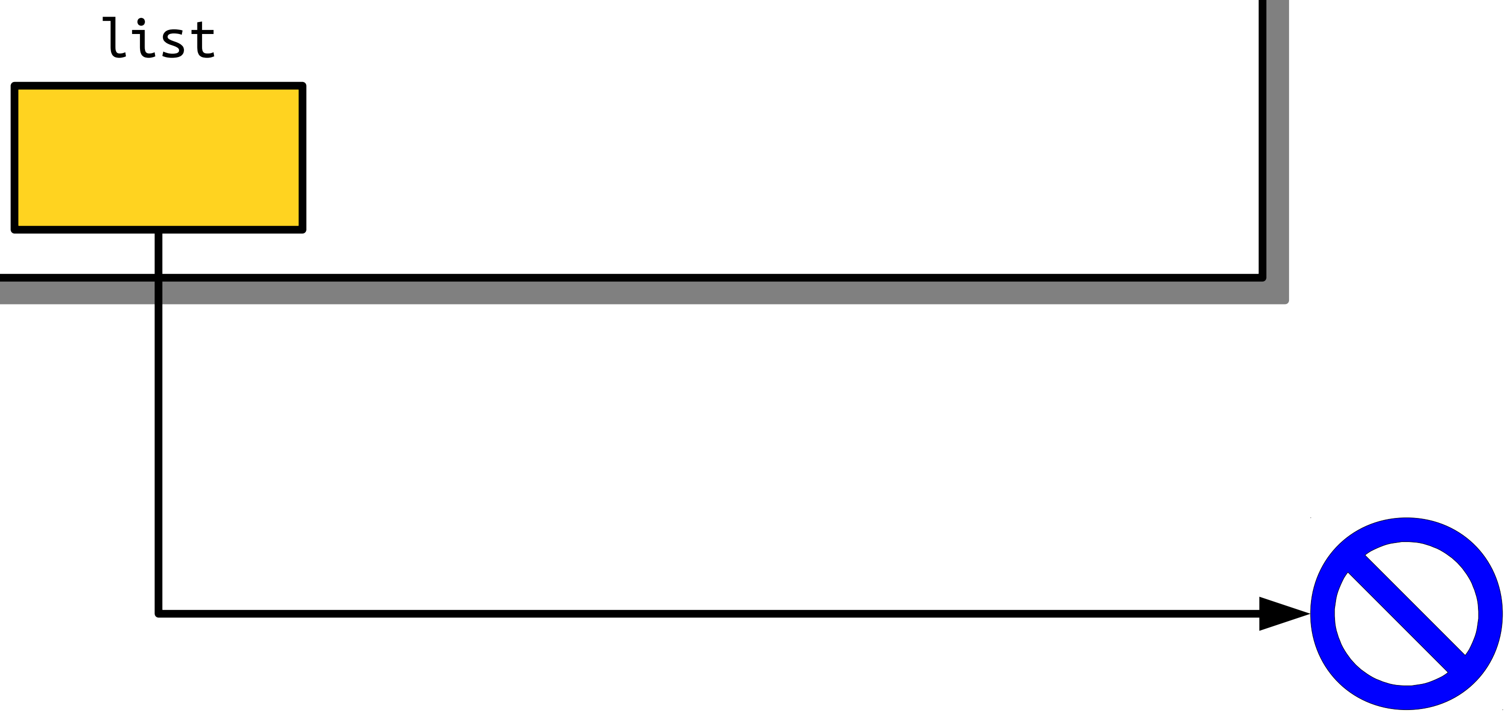
```
void deleteList(Cell* list) {
   while (list != nullptr) {
      Cell* next = list->next;
      delete list;
      list = next;
   }
}
```



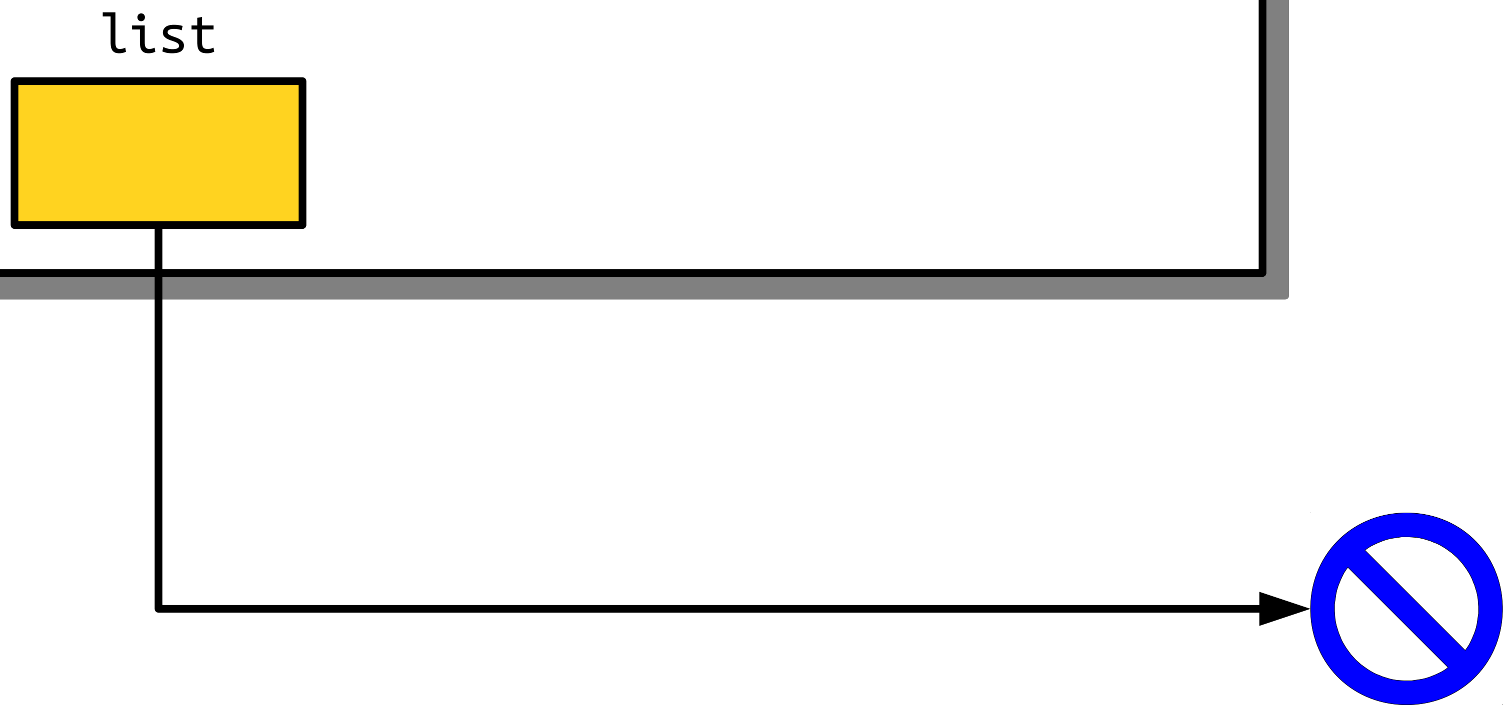
```
void deleteList(Cell* list) {
   while (list != nullptr) {
      Cell* next = list->next;
      delete list;
      list = next;
   }
}
```



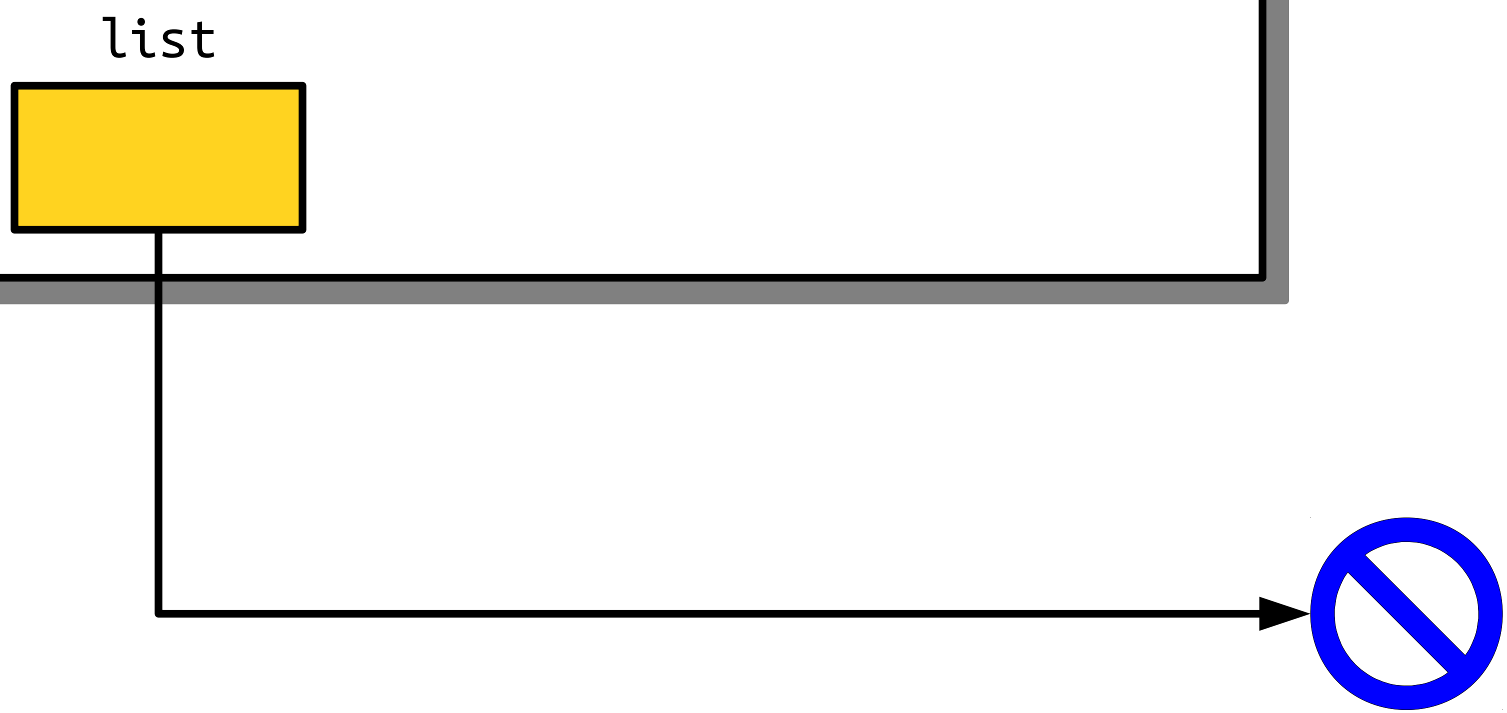
```
void deleteList(Cell* list) {
   while (list != nullptr) {
      Cell* next = list->next;
      delete list;
      list = next;
   }
}
```



```
void deleteList(Cell* list) {
    while (list != nullptr) {
        Cell* next = list->next;
        delete list;
        list = next;
    }
}
```

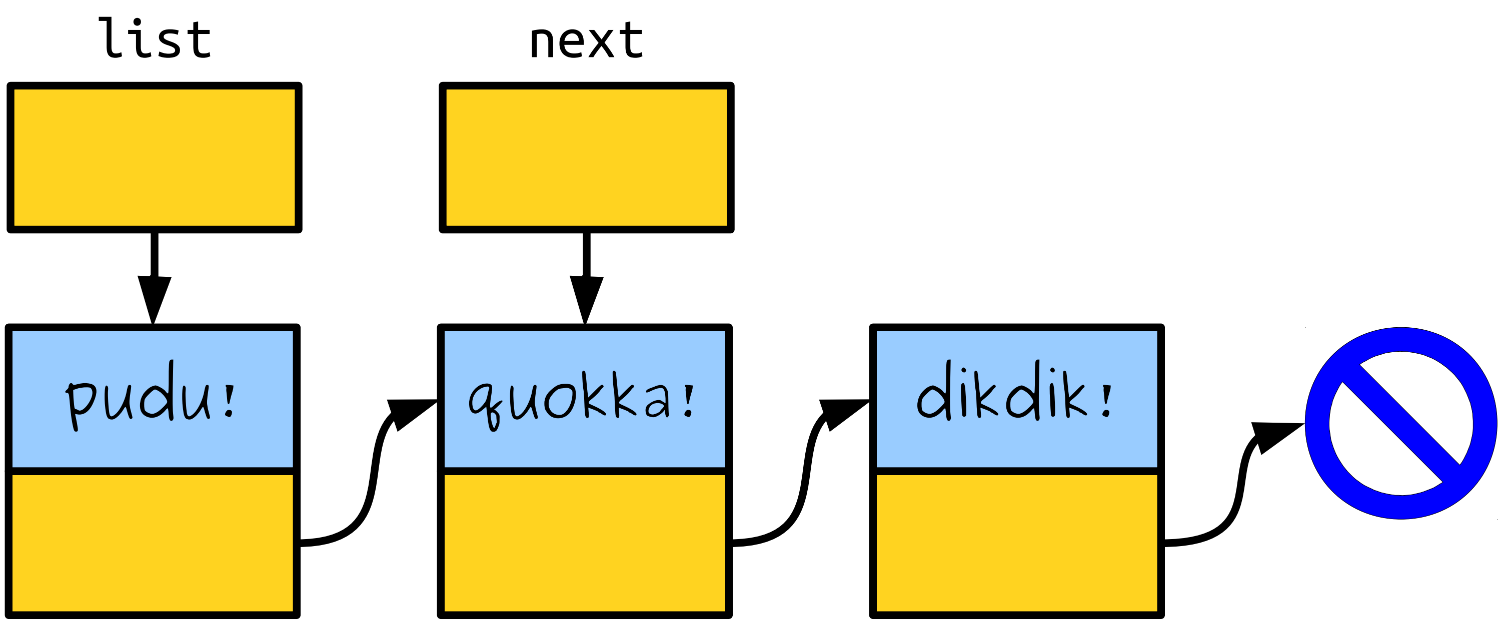


```
void deleteList(Cell* list) {
    while (list != nullptr) {
        Cell* next = list->next;
        delete list;
        list = next;
    }
}
```



Pointers Into Lists

- When processing linked lists iteratively, it's common to introduce pointers that point to cells in multiple spots in the list.
- This is particularly useful if we're destroying or rewiring existing lists.

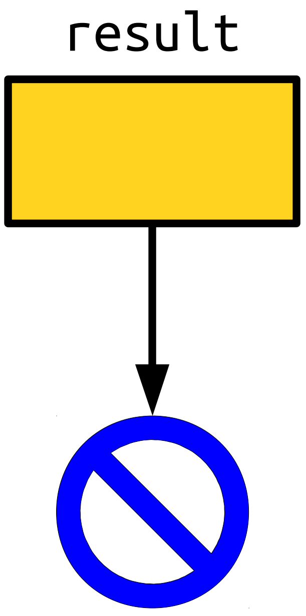


Building a Linked List

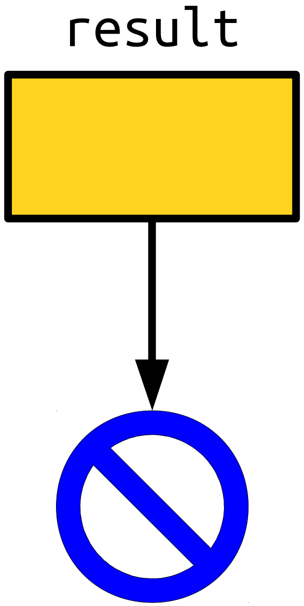
```
Cell* result = nullptr;
while (true) {









}
return result;
```

```
Cell* result = nullptr;
while (true) {







}
return result;
```

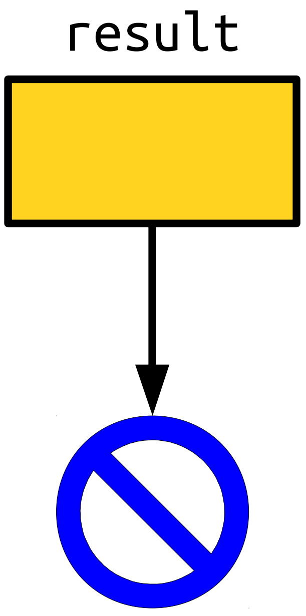
```
Cell* result = nullptr;
while (true) {






}
return result;
```



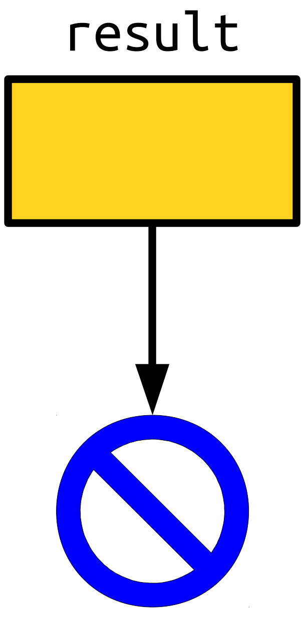
```
Cell* result = nullptr;
while (true) {



}
return result;
```



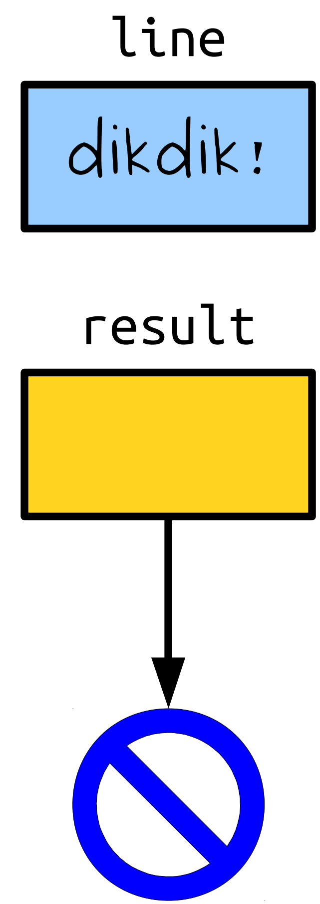
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;









}
return result;
```



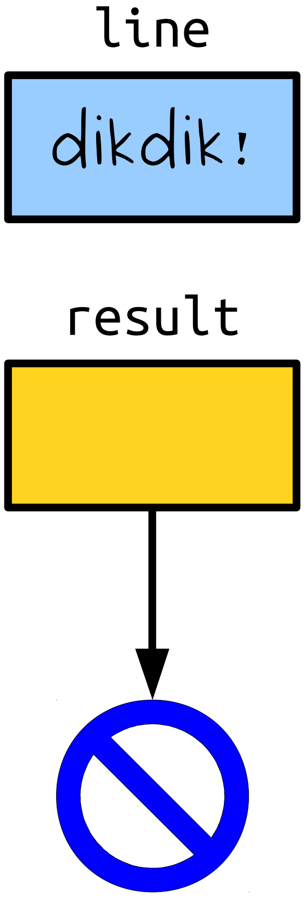
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;






}
return result;
```



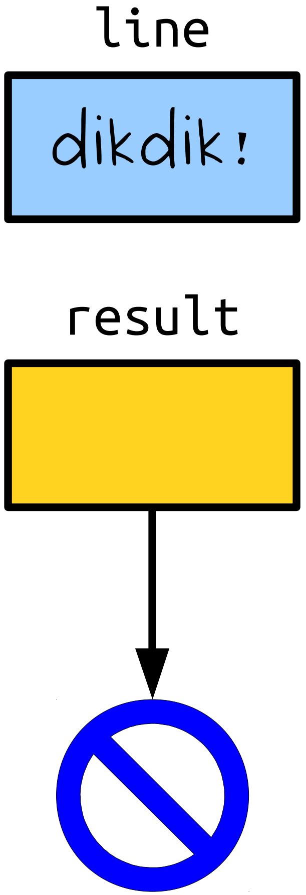
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;




}
return result;
```



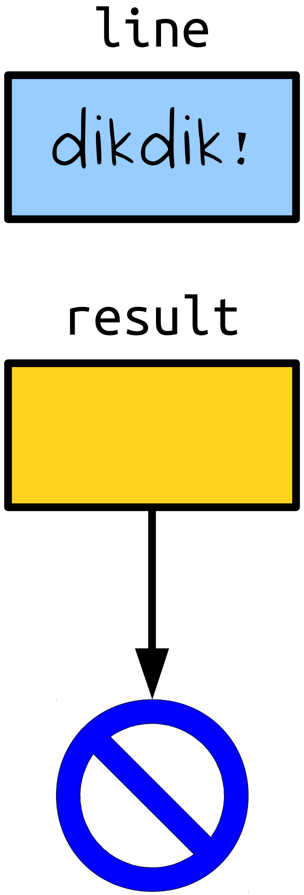
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;




}
return result;
```



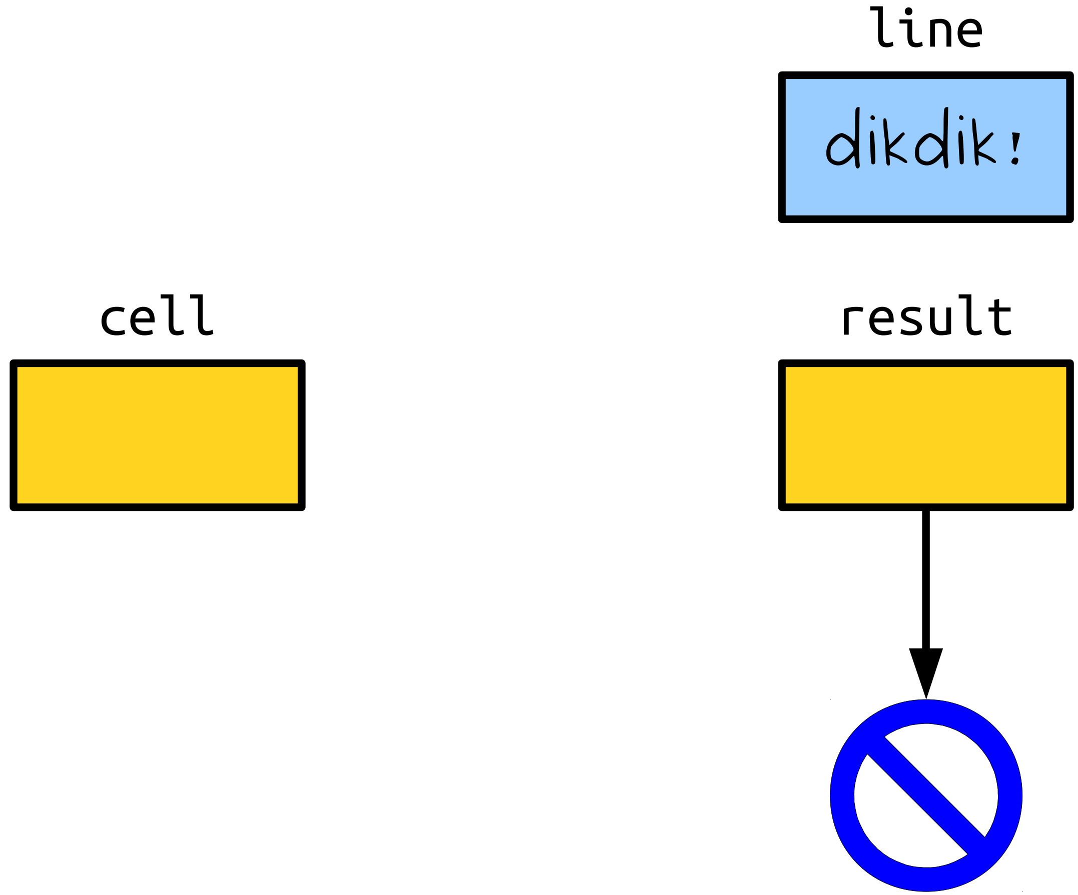
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;




}
return result;
```



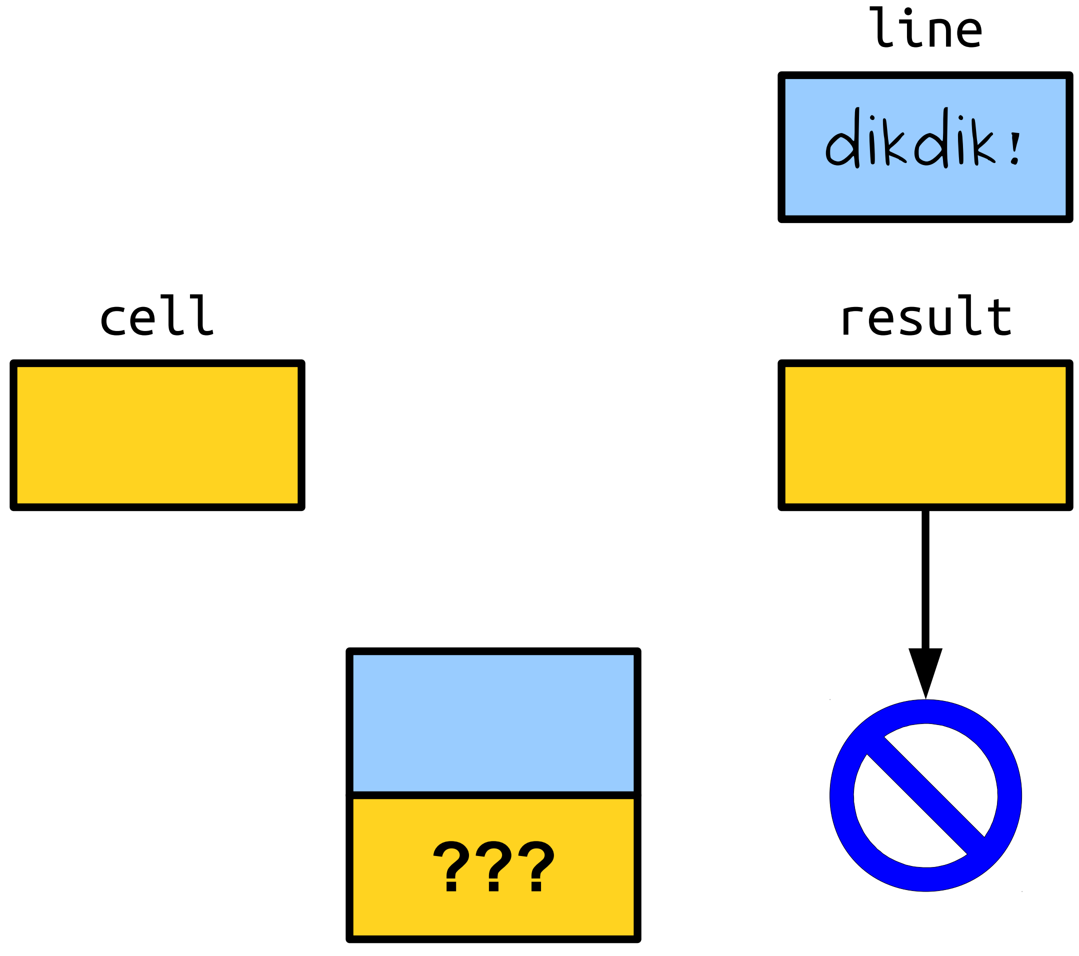
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;




}
return result;
```



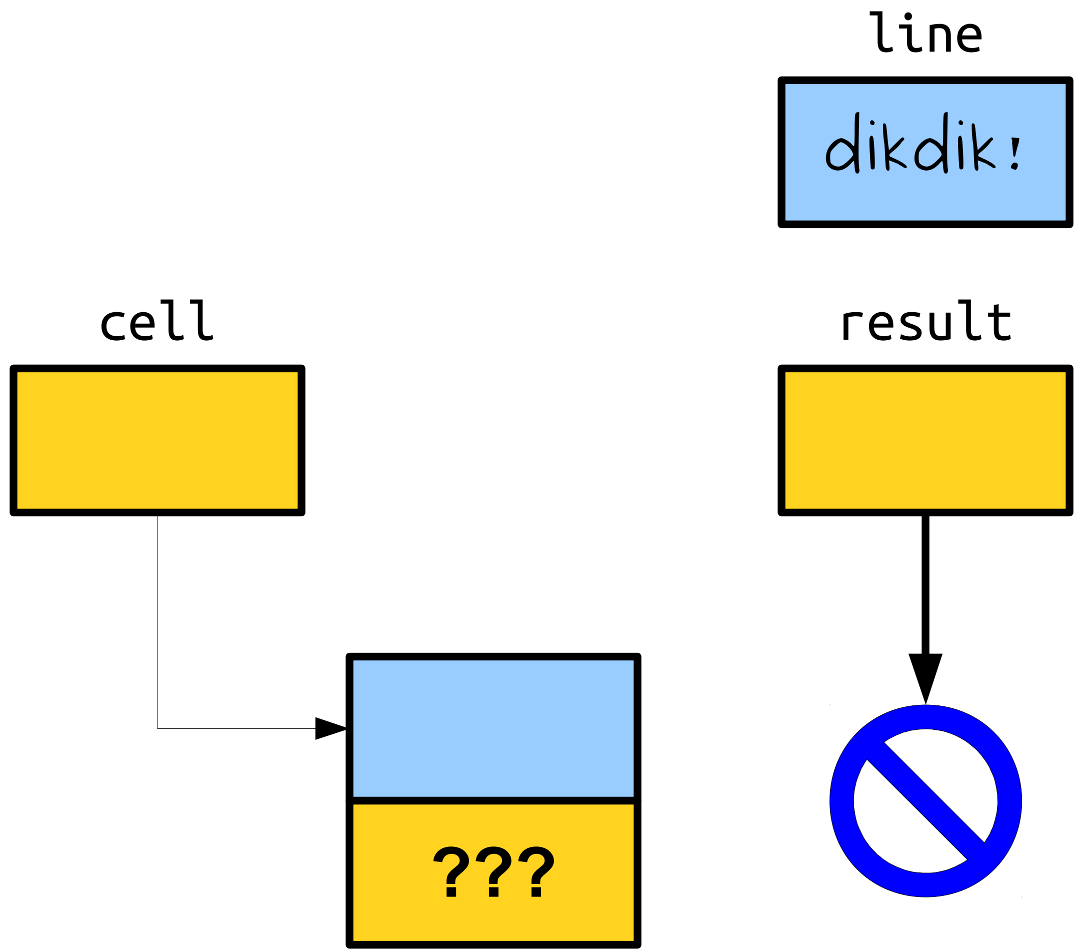
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;



}
return result;
```



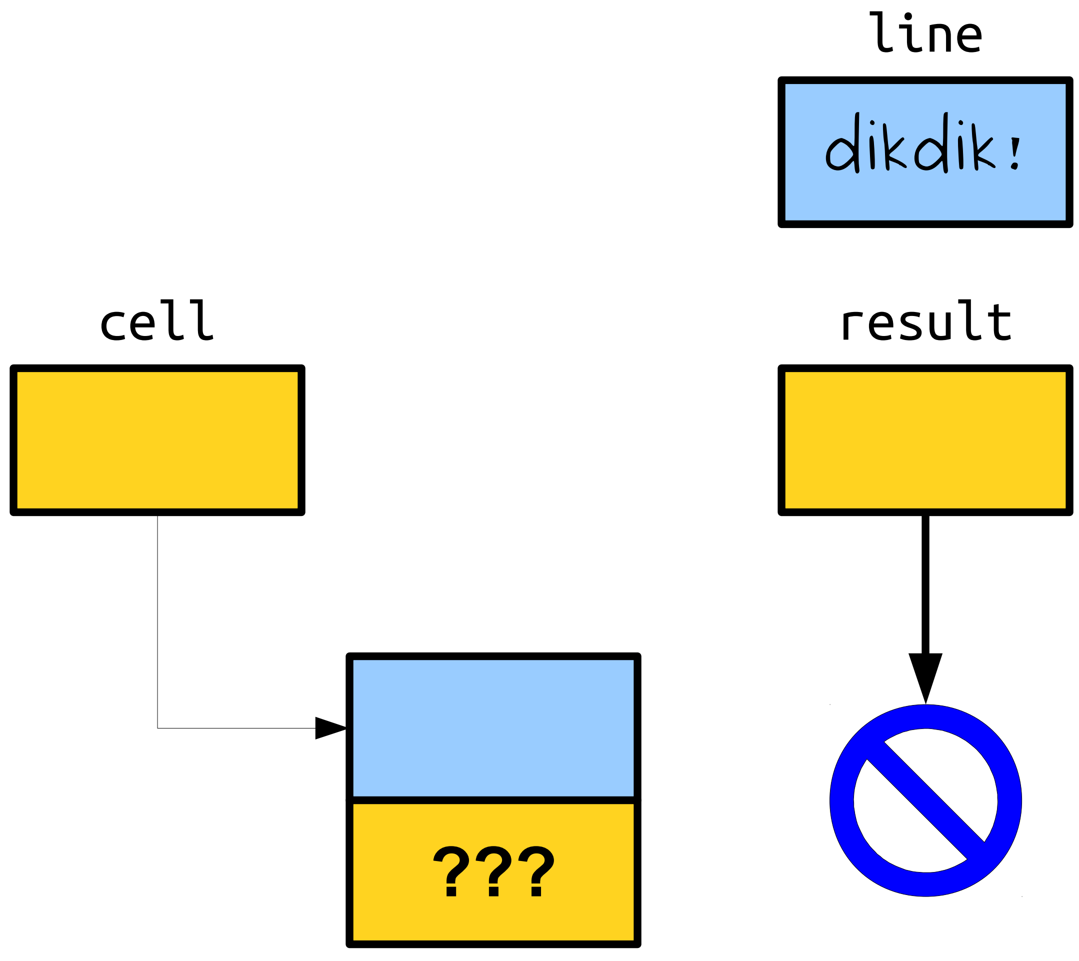
```
Cell* result = nullptr;
while (true) {
   string line = getLine("Next entry? ");
   if (line == "") break;

   Cell* cell = new Cell;



}
return result;
```



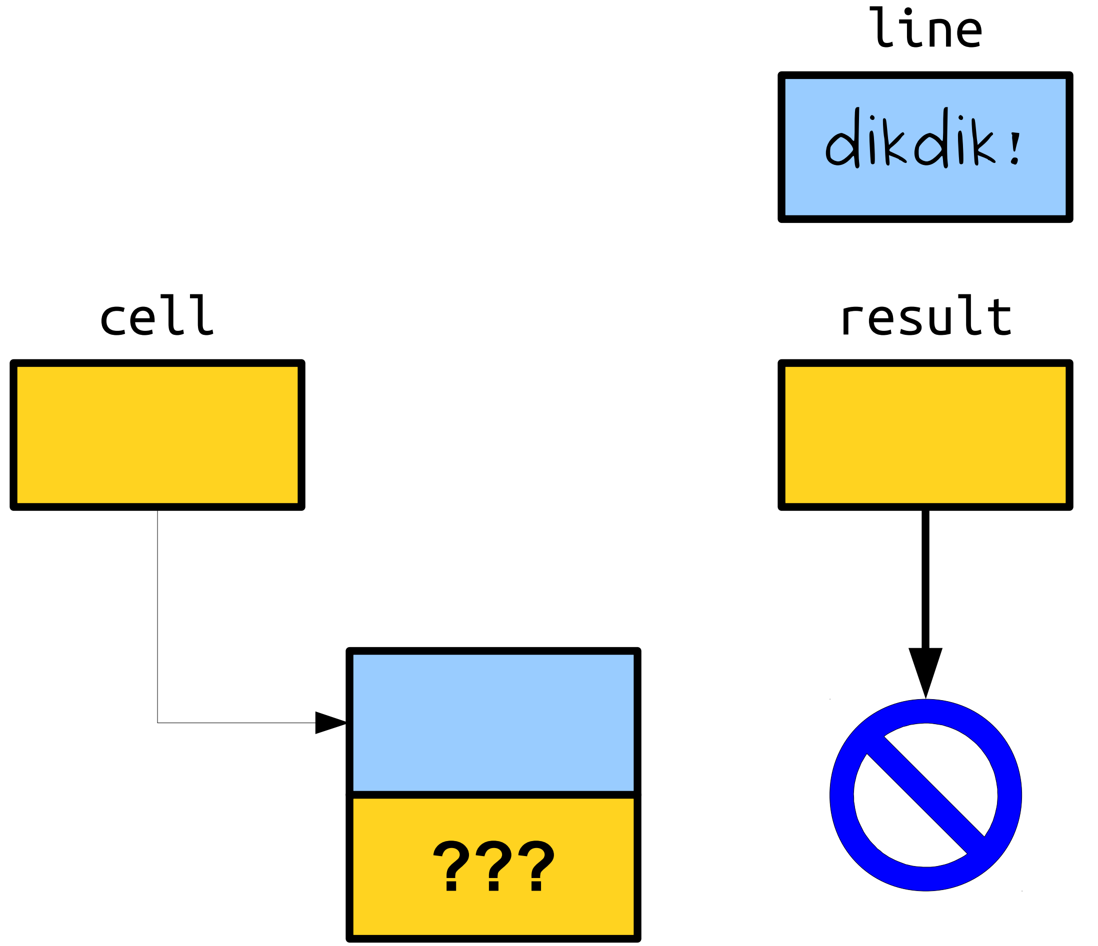
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;



}
return result;
```



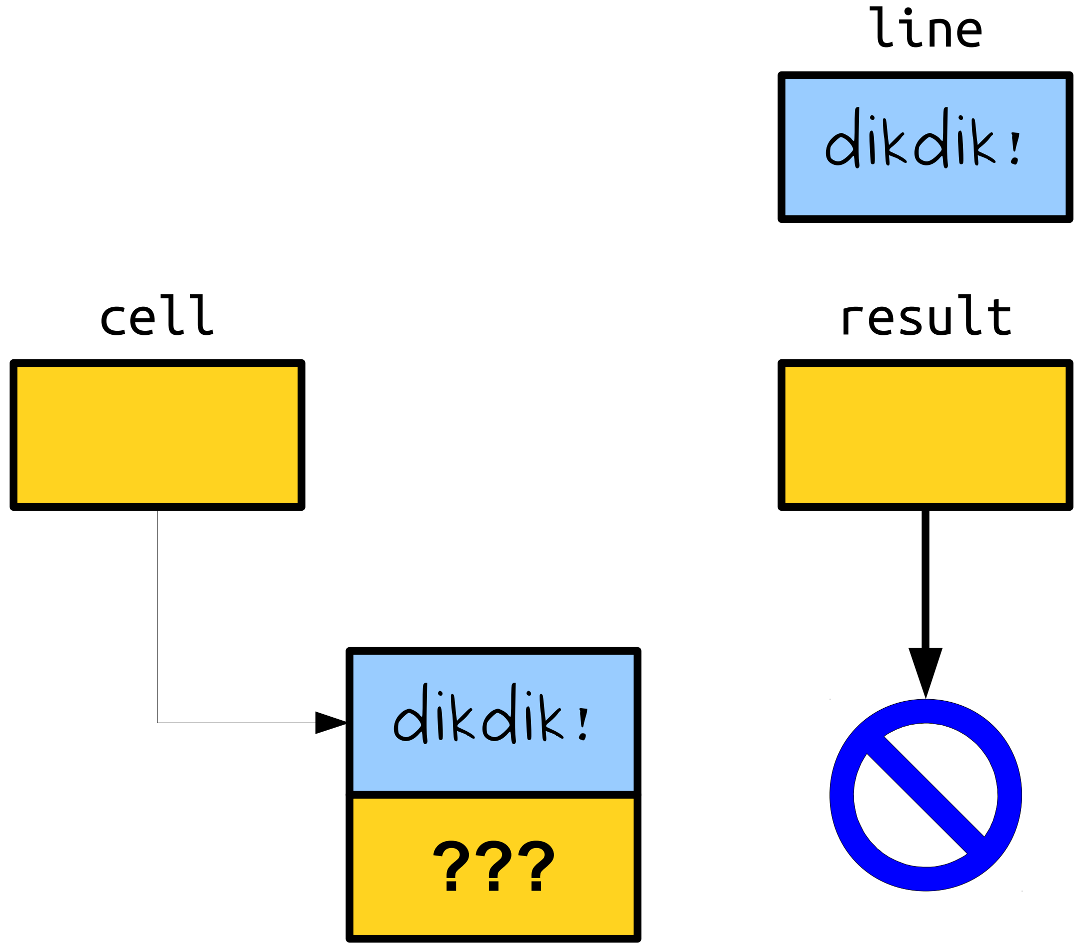
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;



}
return result;
```



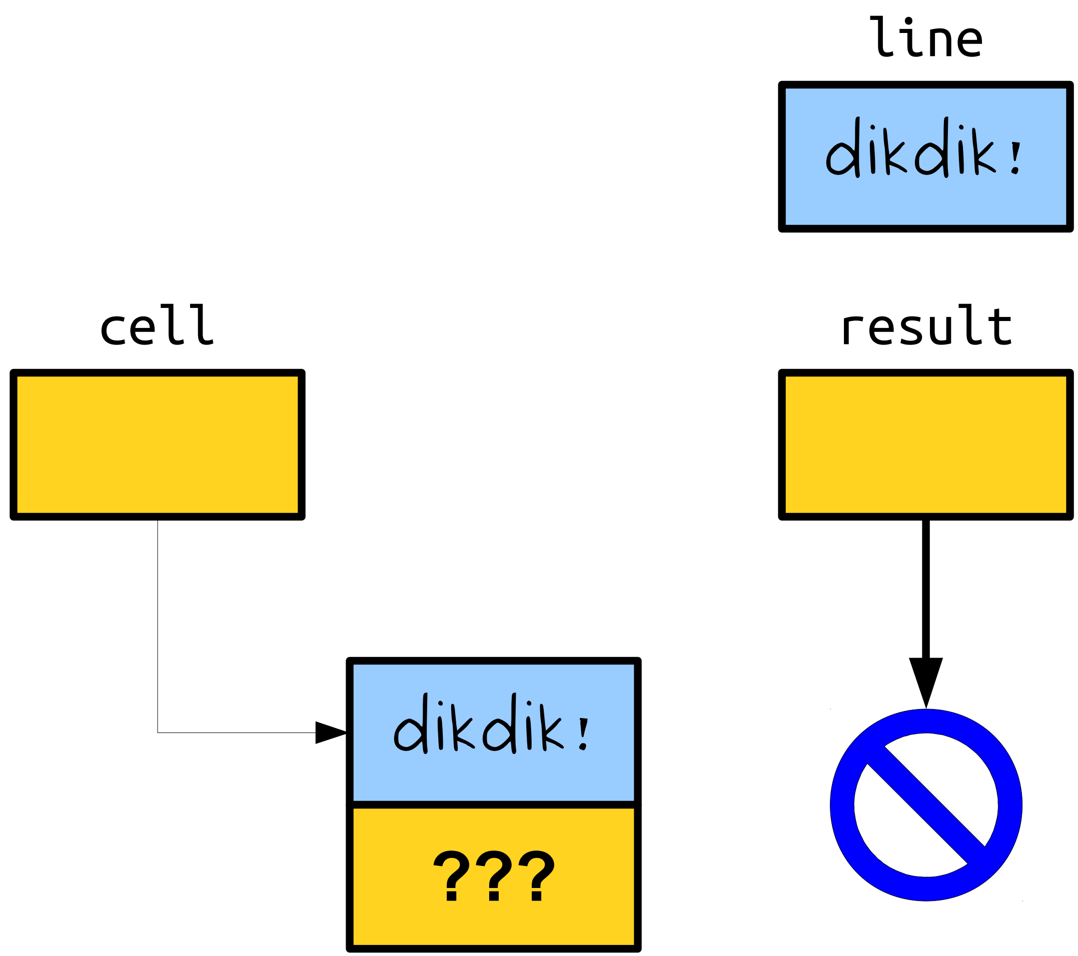
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;


}
return result;
```



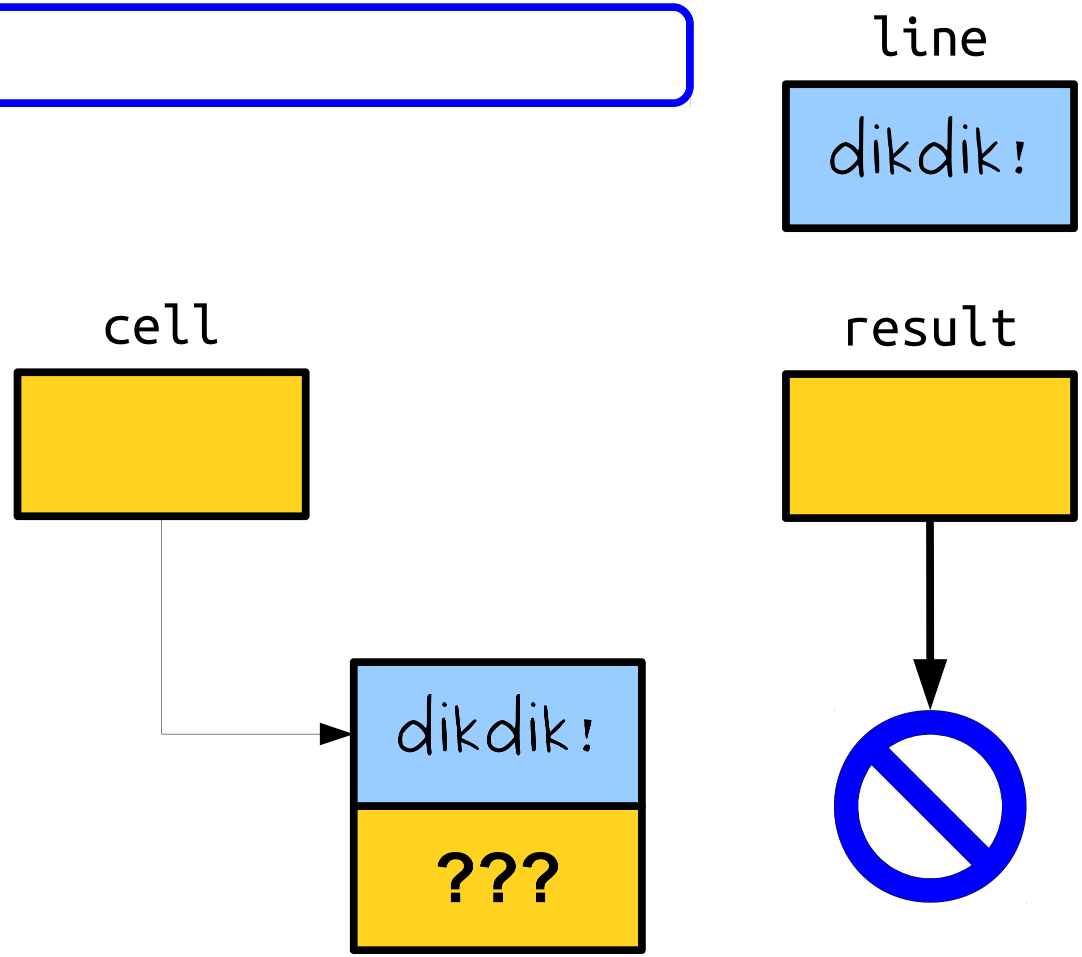
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;


}
return result;
```



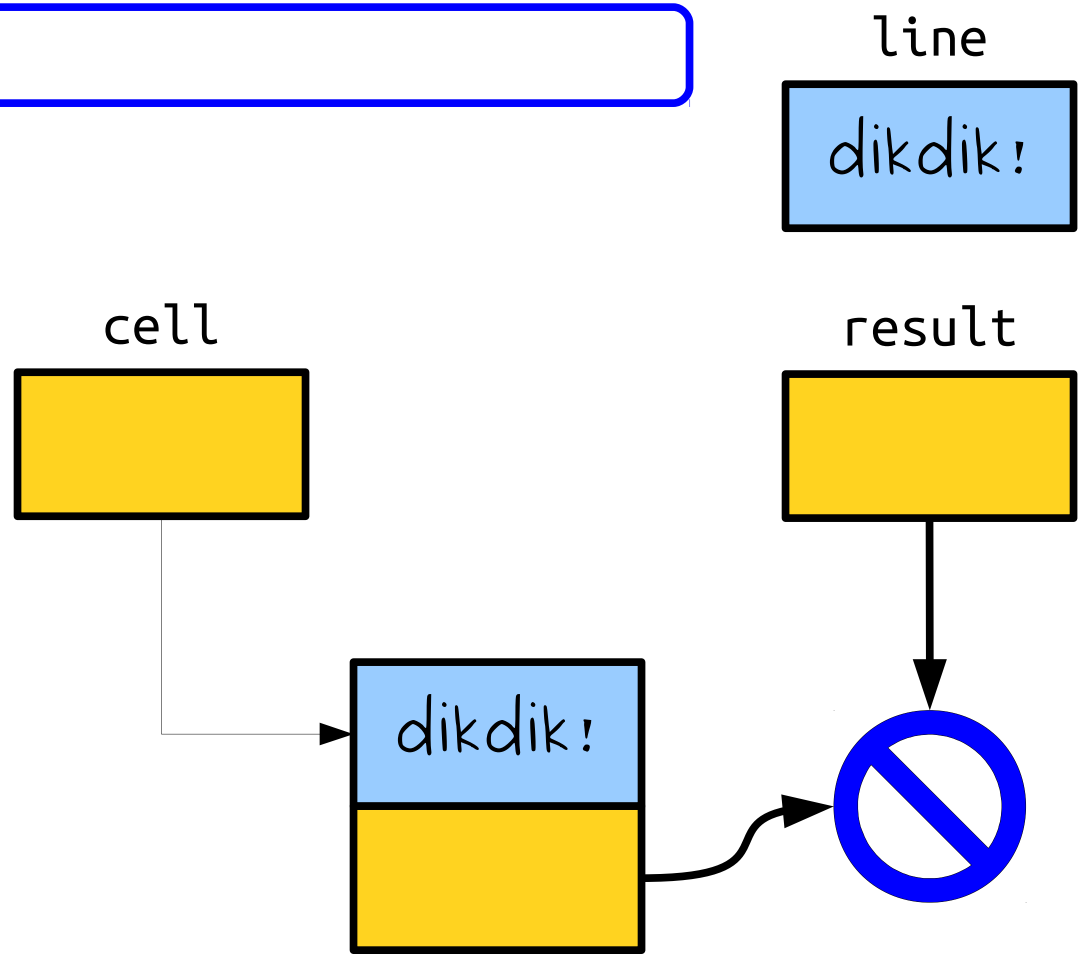
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;

}
return result;
```



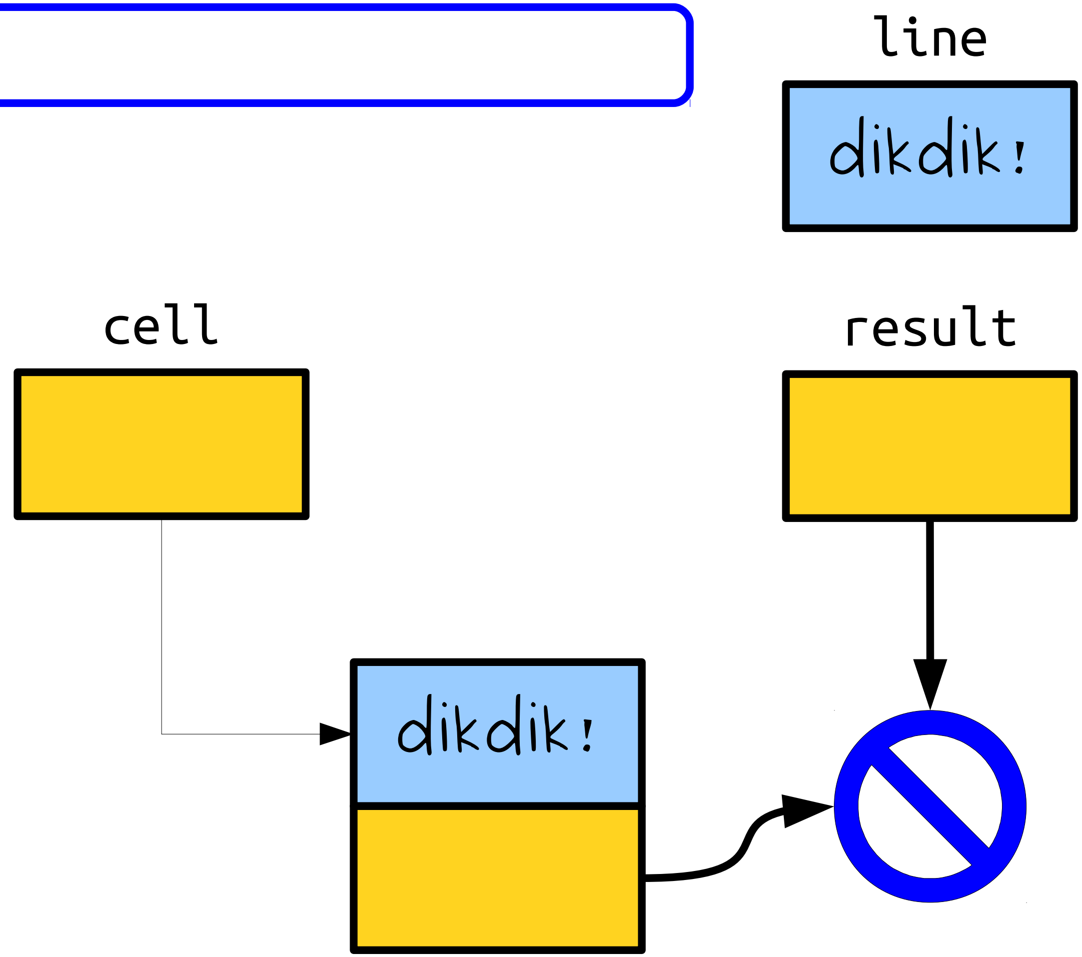
```
Cell* result = nullptr;
while (true) {
   string line = getLine("Next entry? ");
   if (line == "") break;

   Cell* cell = new Cell;
   cell->value = line;

   cell->next = result;

}
return result;
```



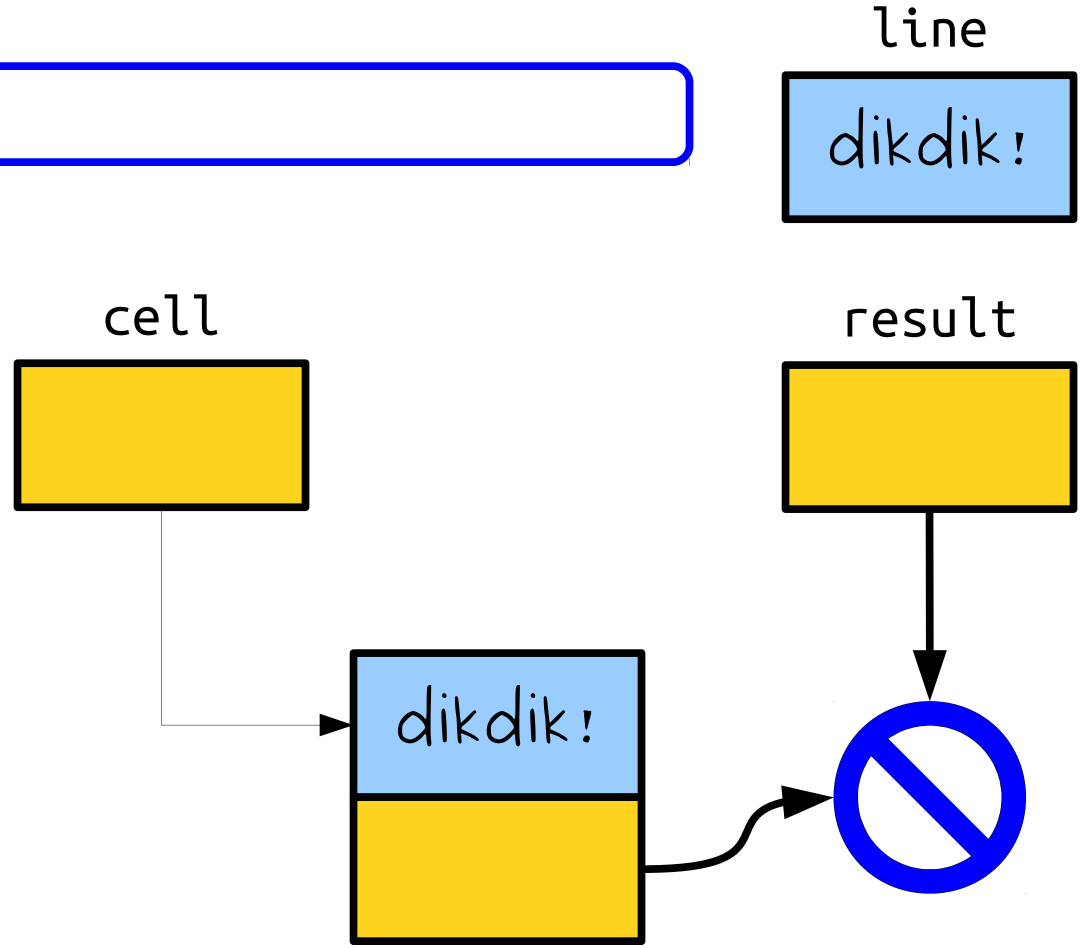
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;

}
return result;
```



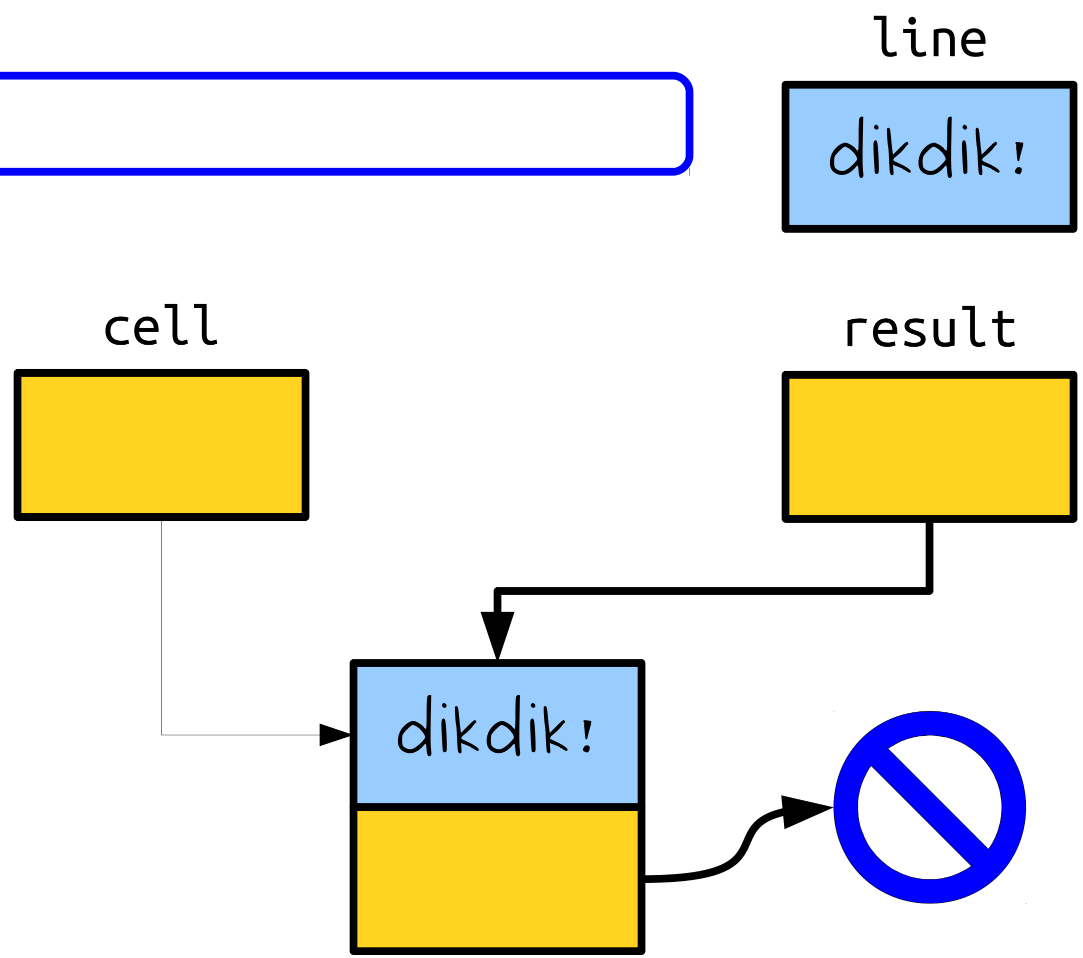
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;
    result = cell;
}
return result;
```



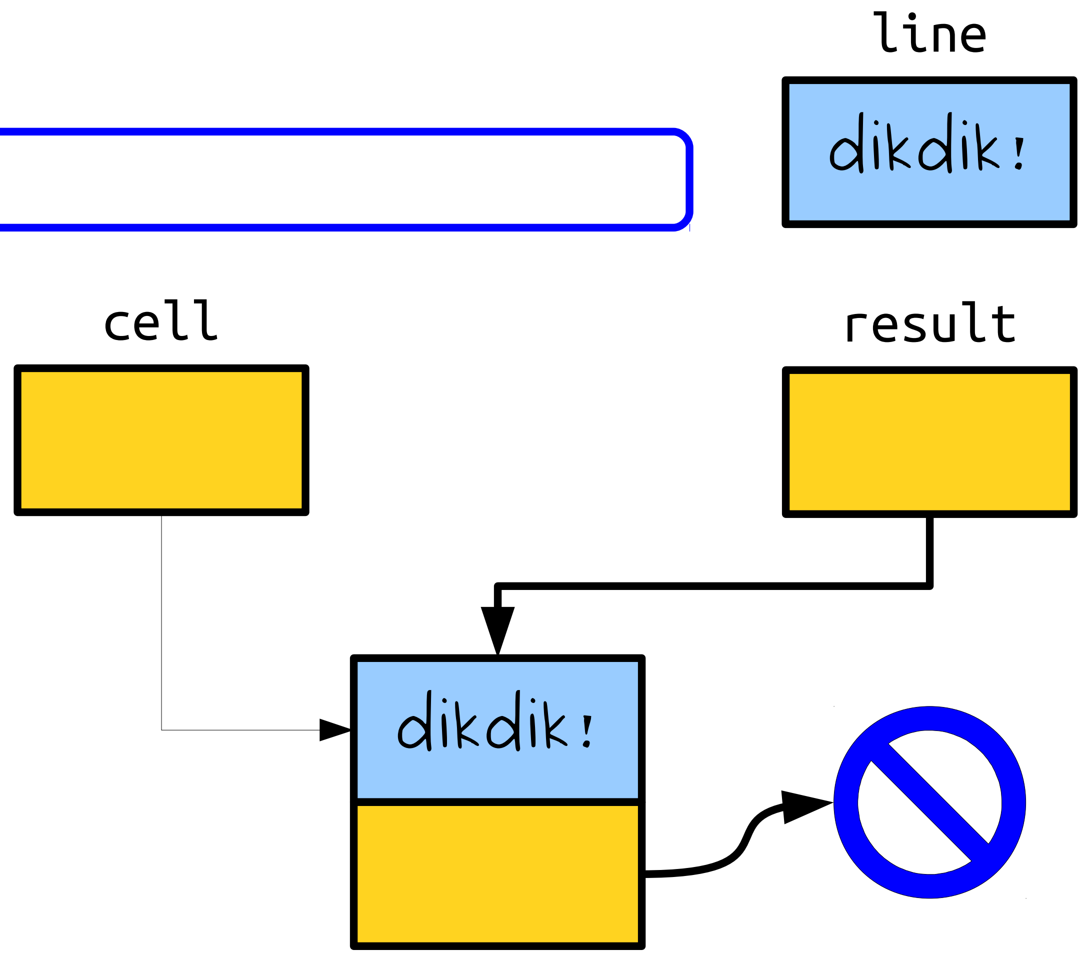
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;
    result = cell;
}
return result;
```



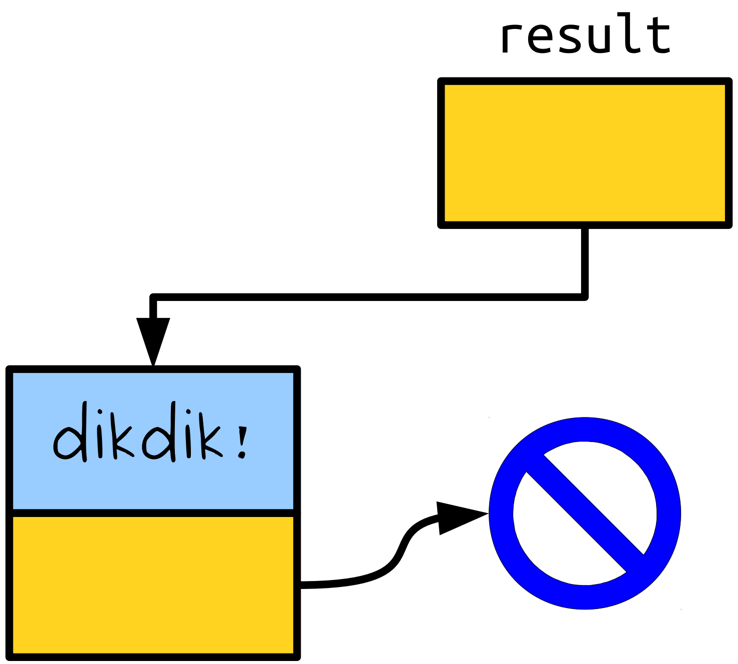
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;
    result = cell;
}
return result;
```



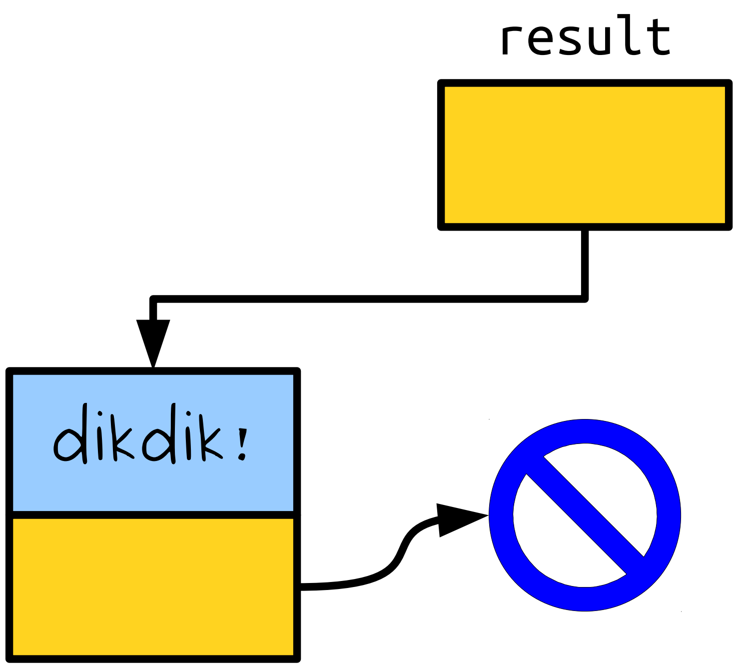
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;
    result = cell;
}
return result;
```



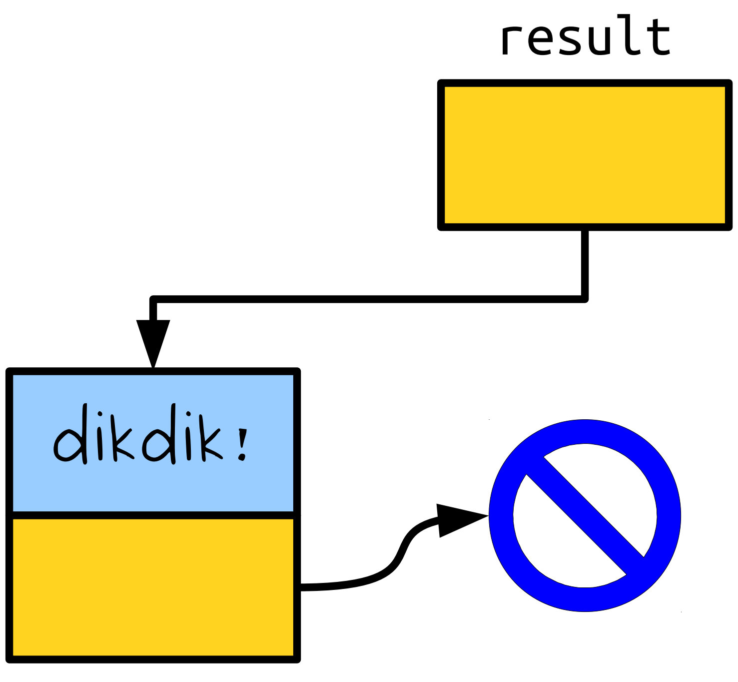
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;
    result = cell;
}
return result;
```



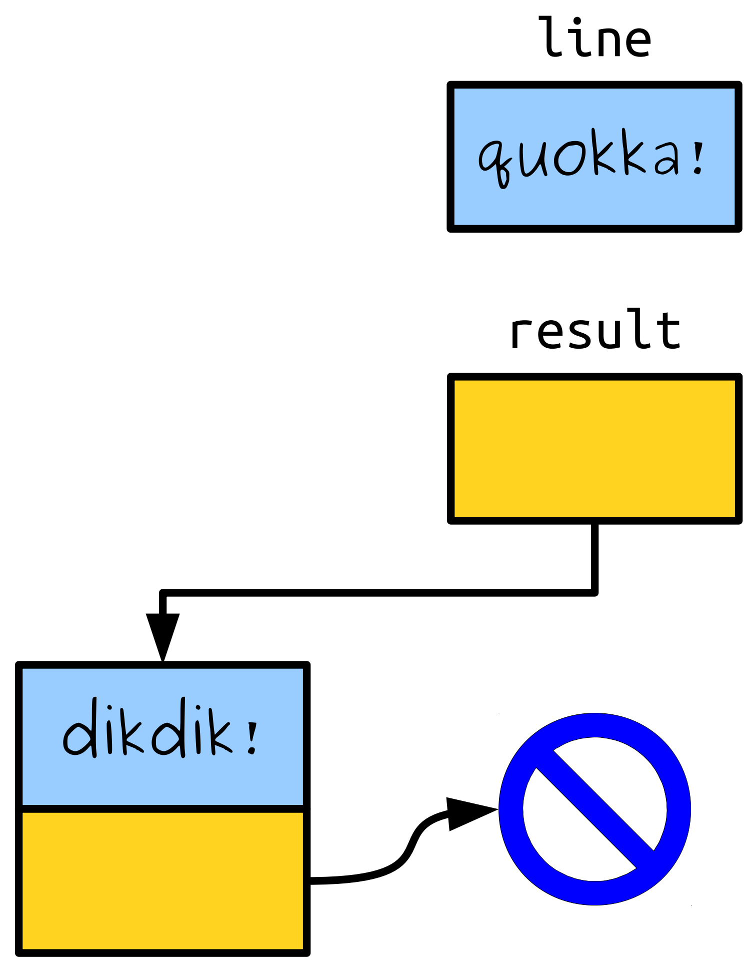
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;
    result = cell;
}
return result;
```



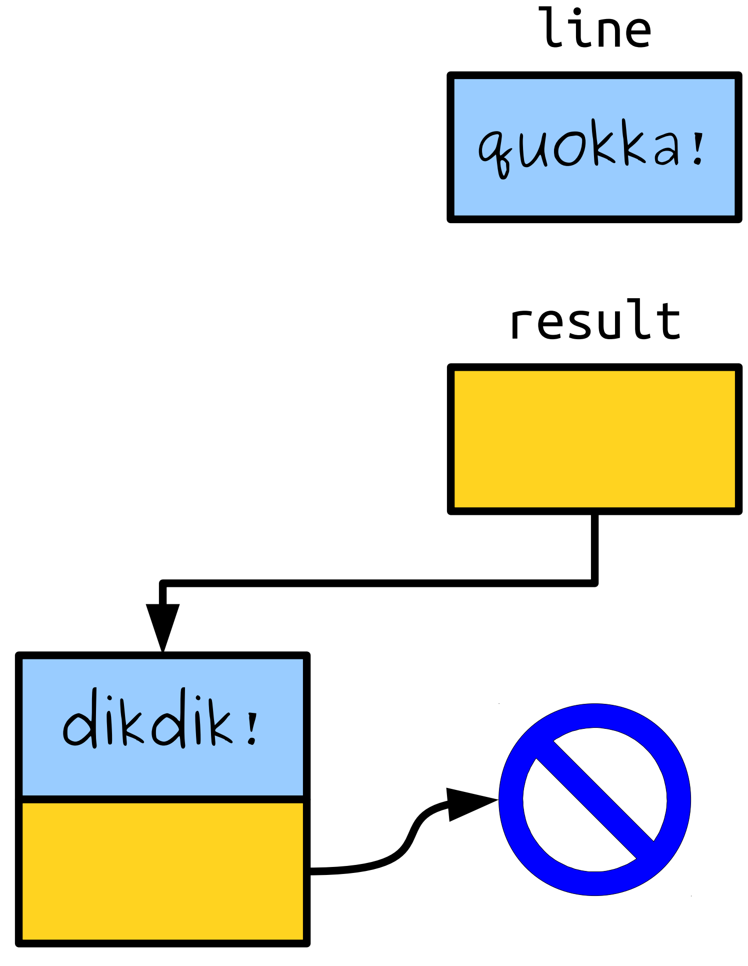
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;
    result = cell;
}
return result;
```



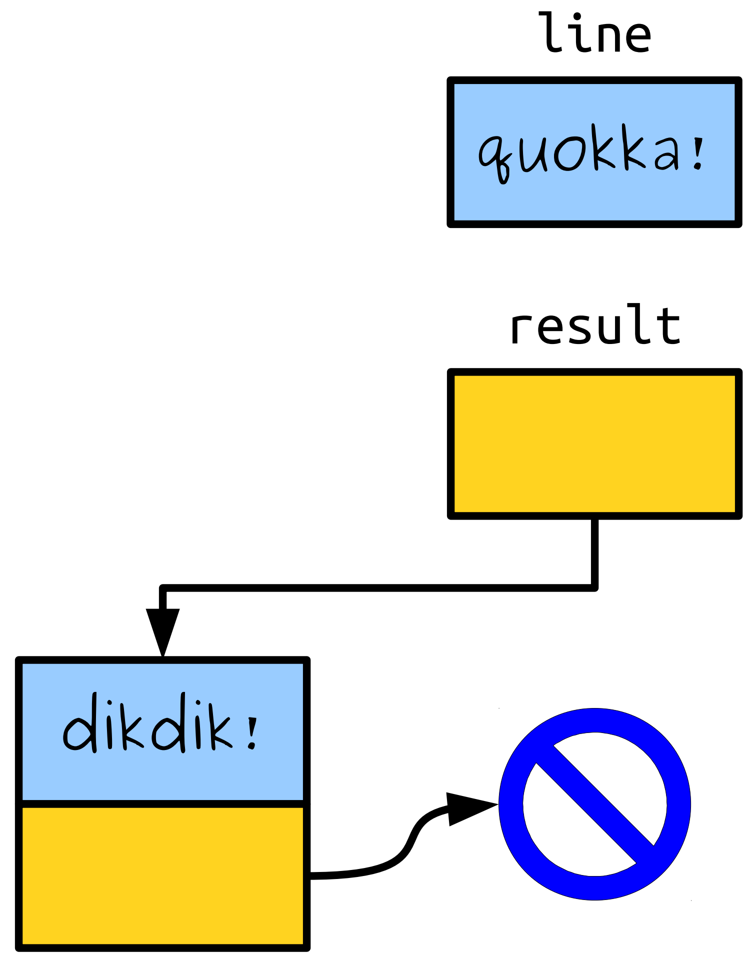
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;
    result = cell;
}
return result;
```



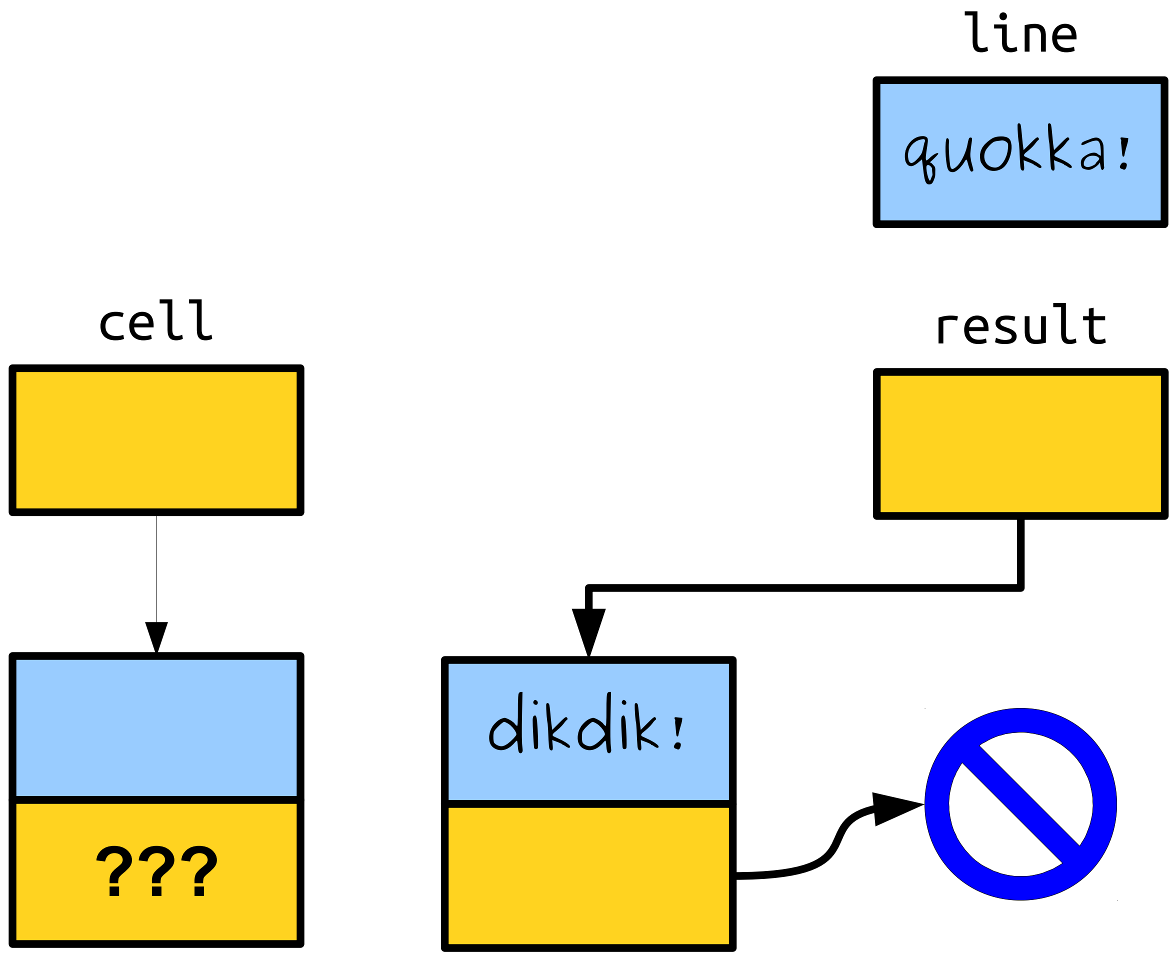
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;
    result = cell;
}
return result;
```



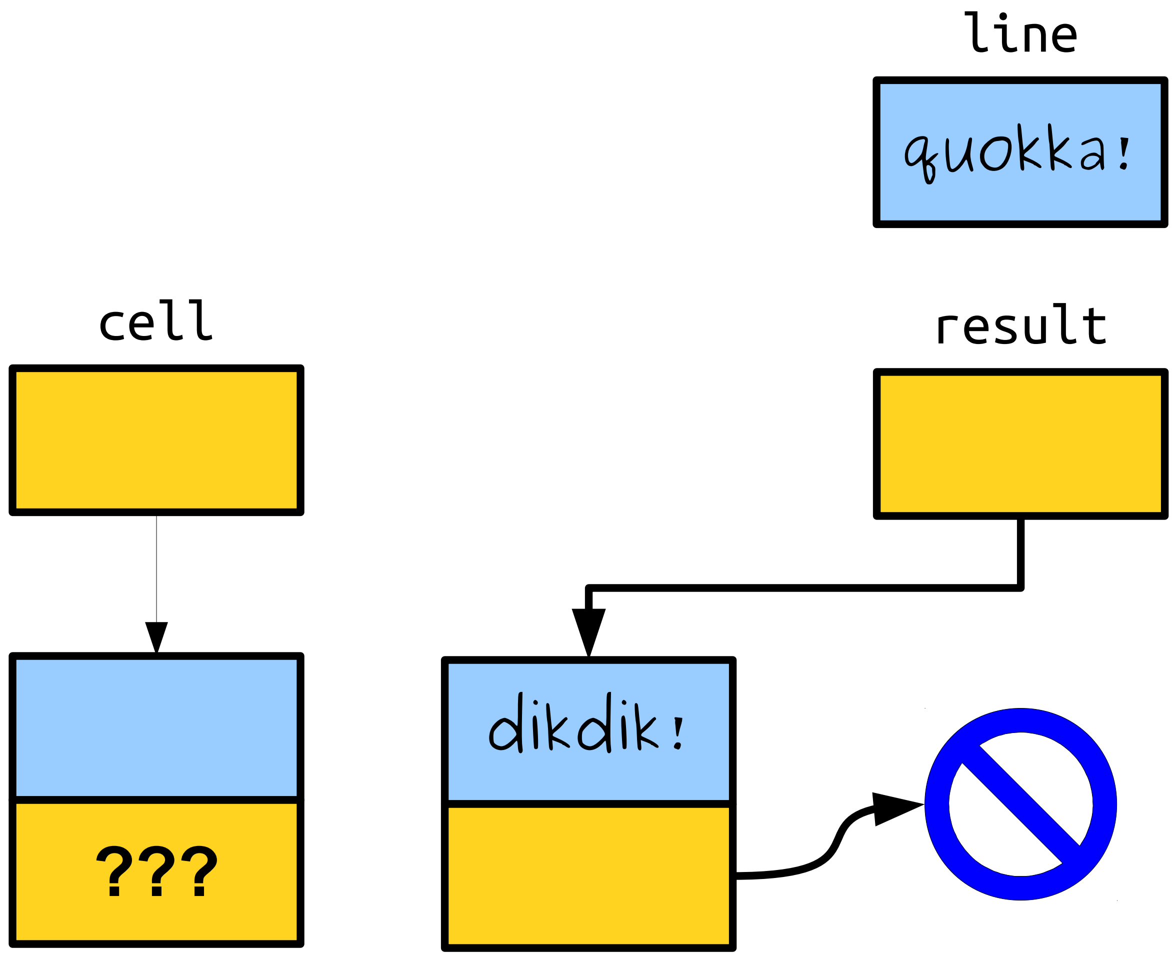
```
Cell* result = nullptr;
while (true) {
   string line = getLine("Next entry? ");
   if (line == "") break;

   Cell* cell = new Cell;
   cell->value = line;

   cell->next = result;
   result = cell;
}
return result;
```



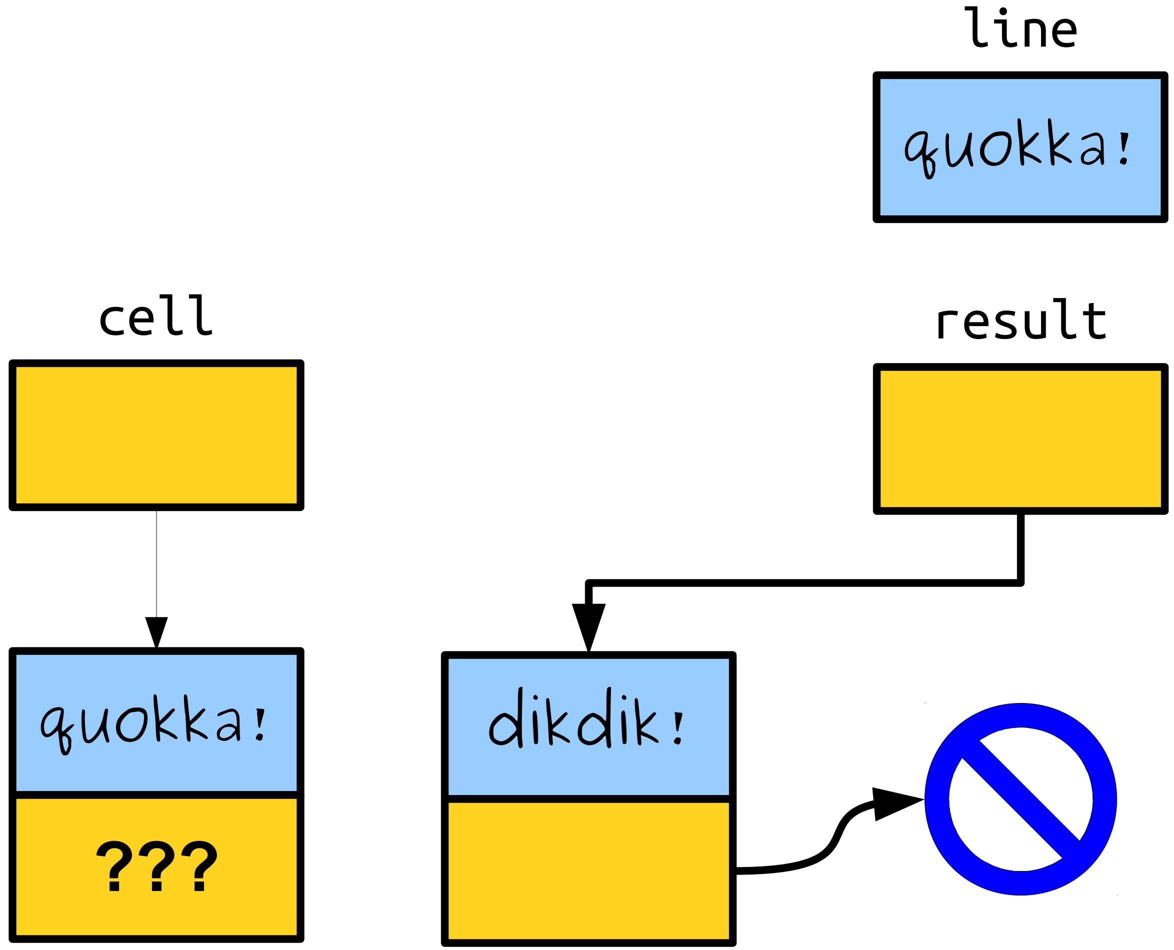
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;
    result = cell;
}
return result;
```



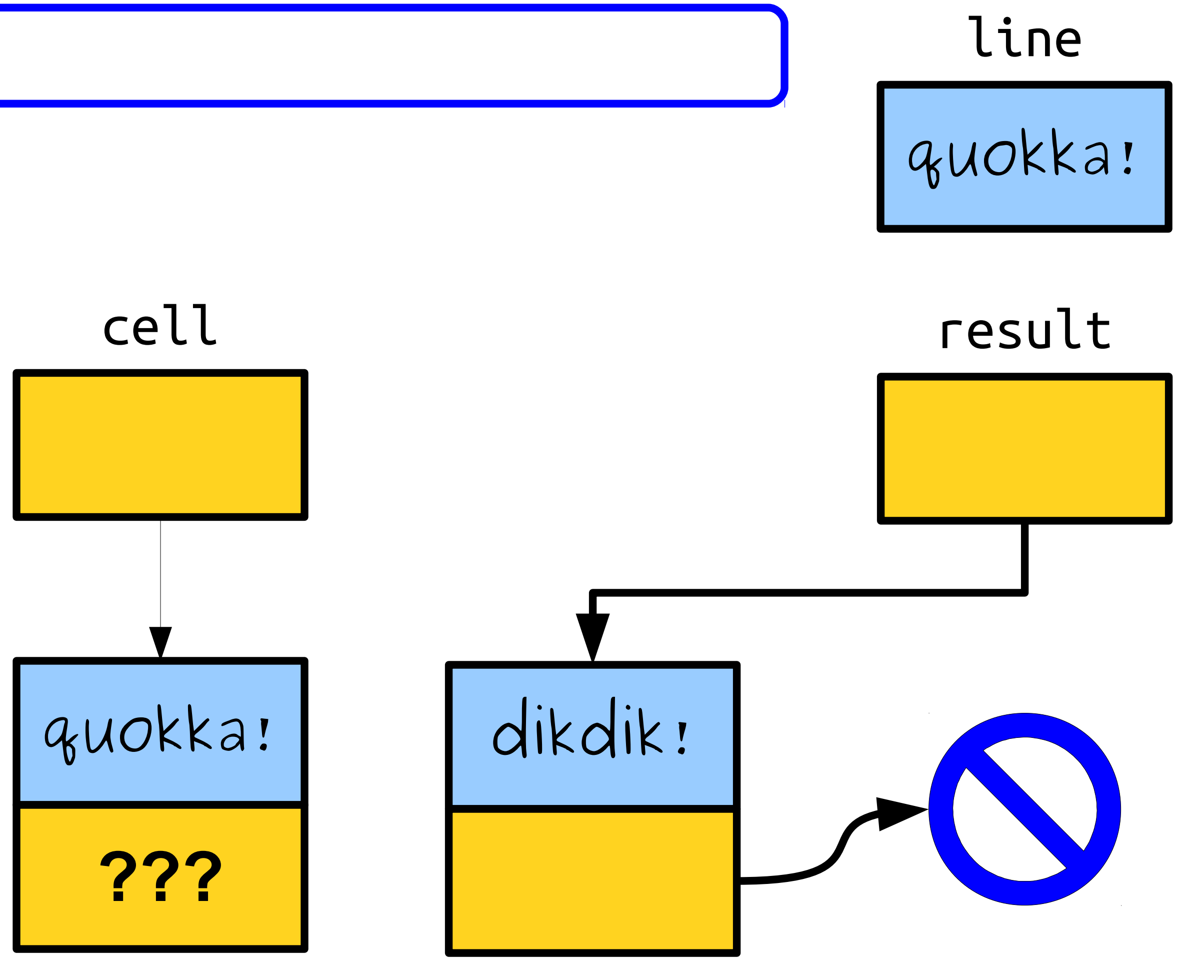
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;
    result = cell;
}
return result;
```



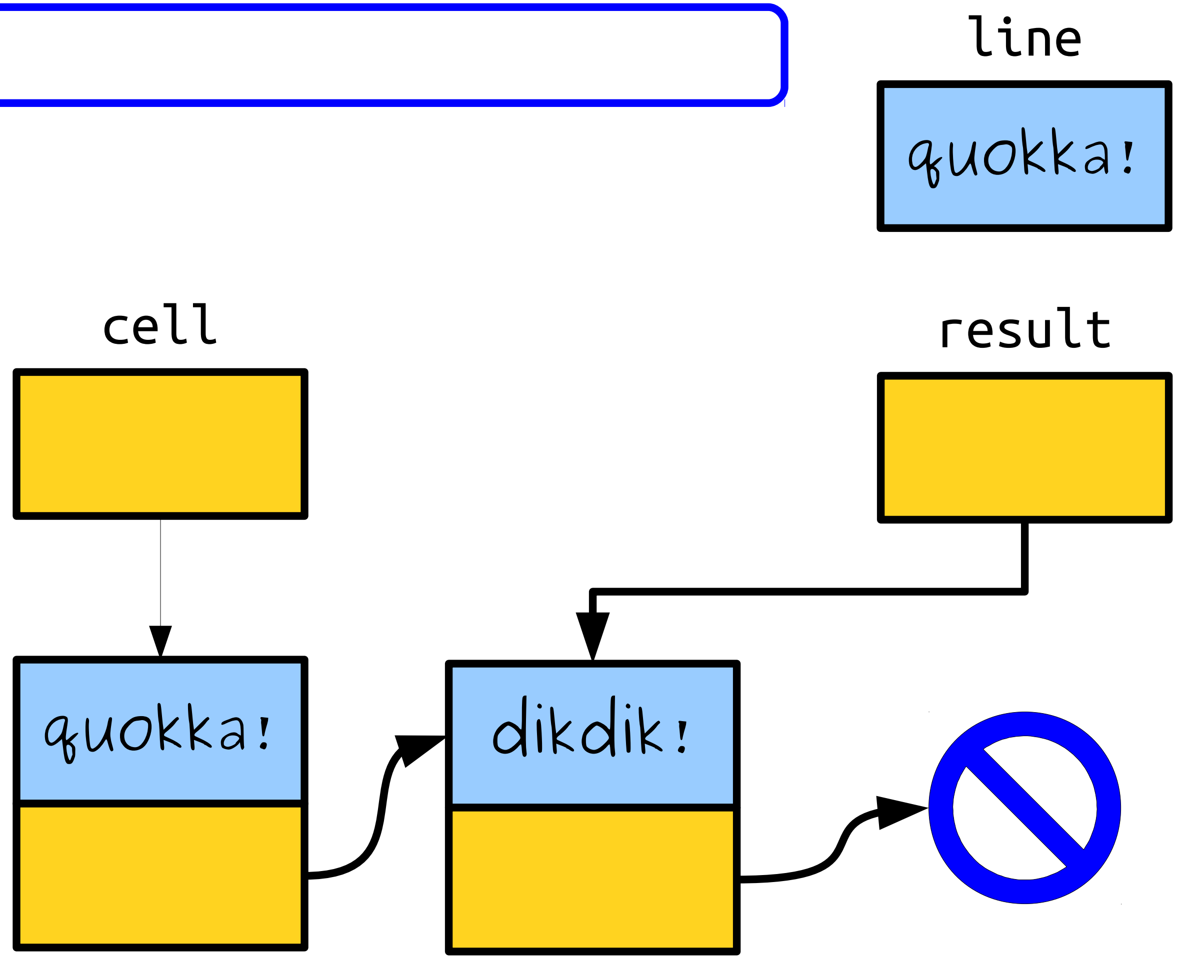
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;
    result = cell;
}
return result;
```



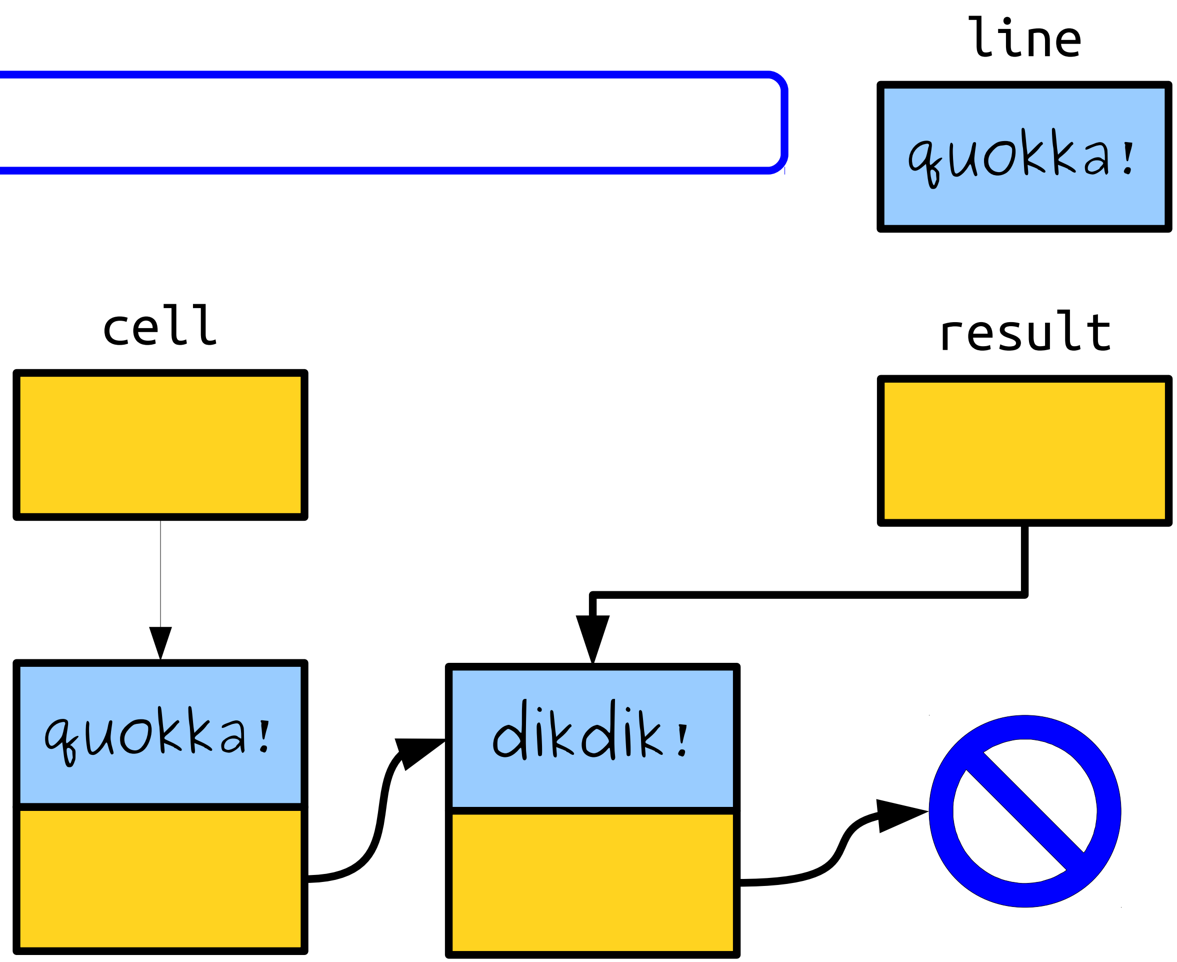
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;
    result = cell;
}
return result;
```



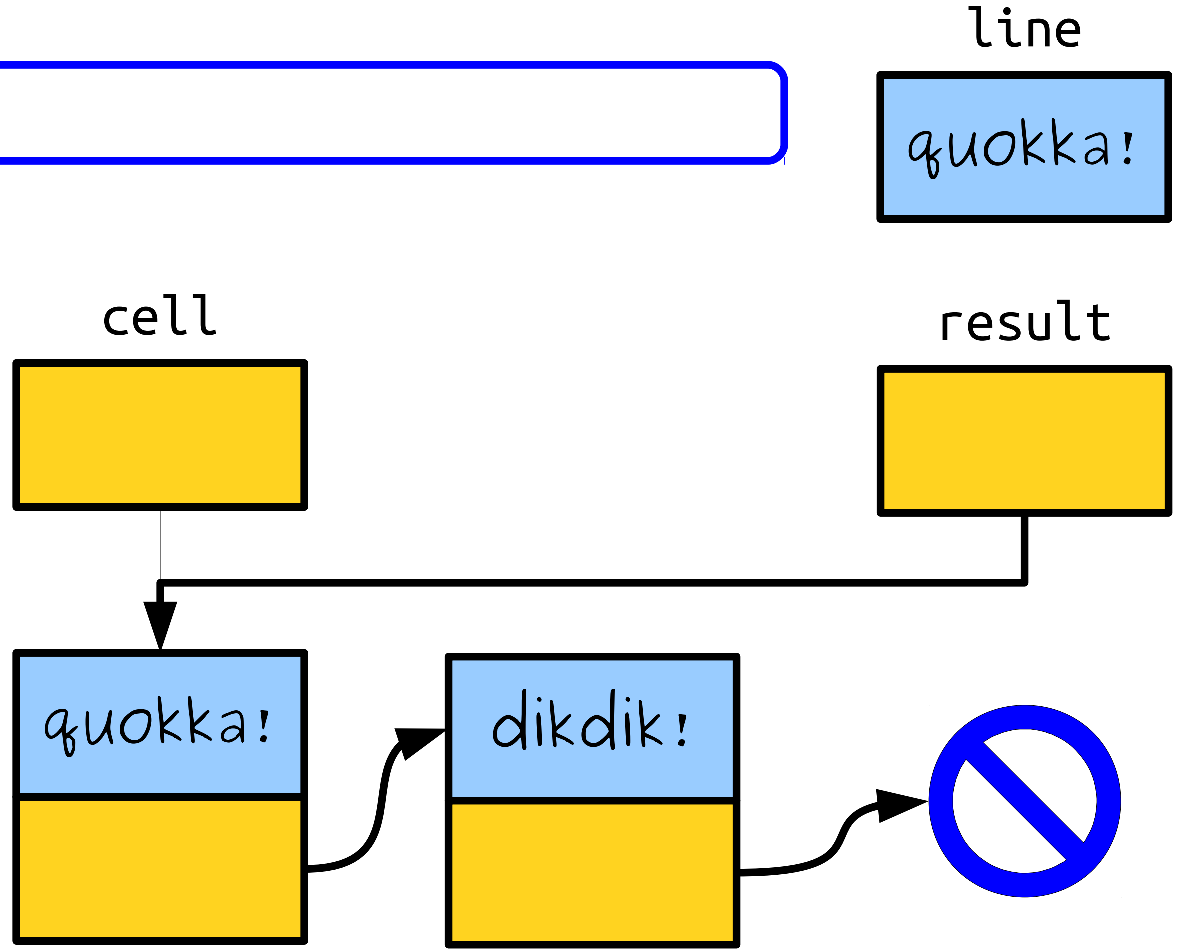
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;
    result = cell;
}
return result;
```



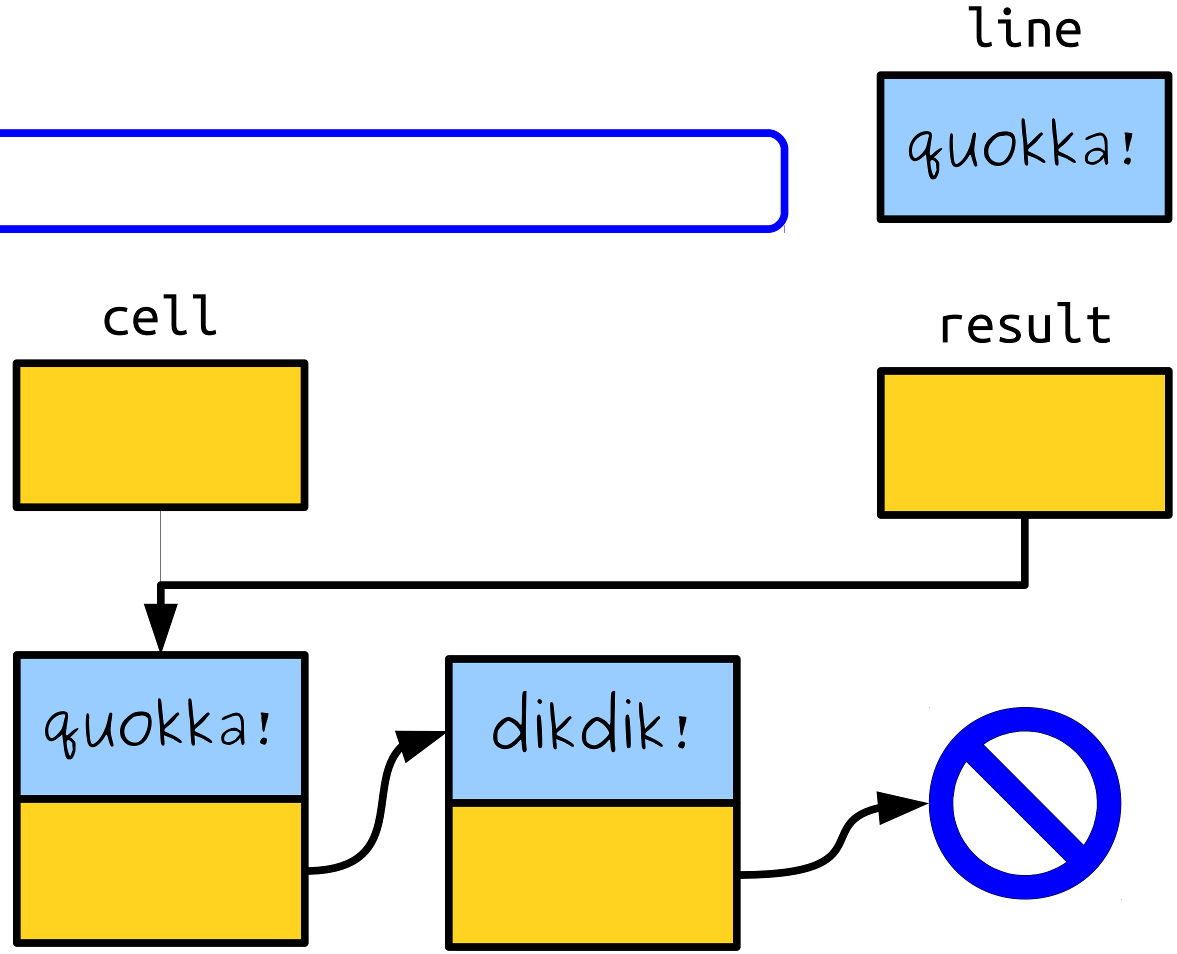
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;
    result = cell;
}
return result;
```



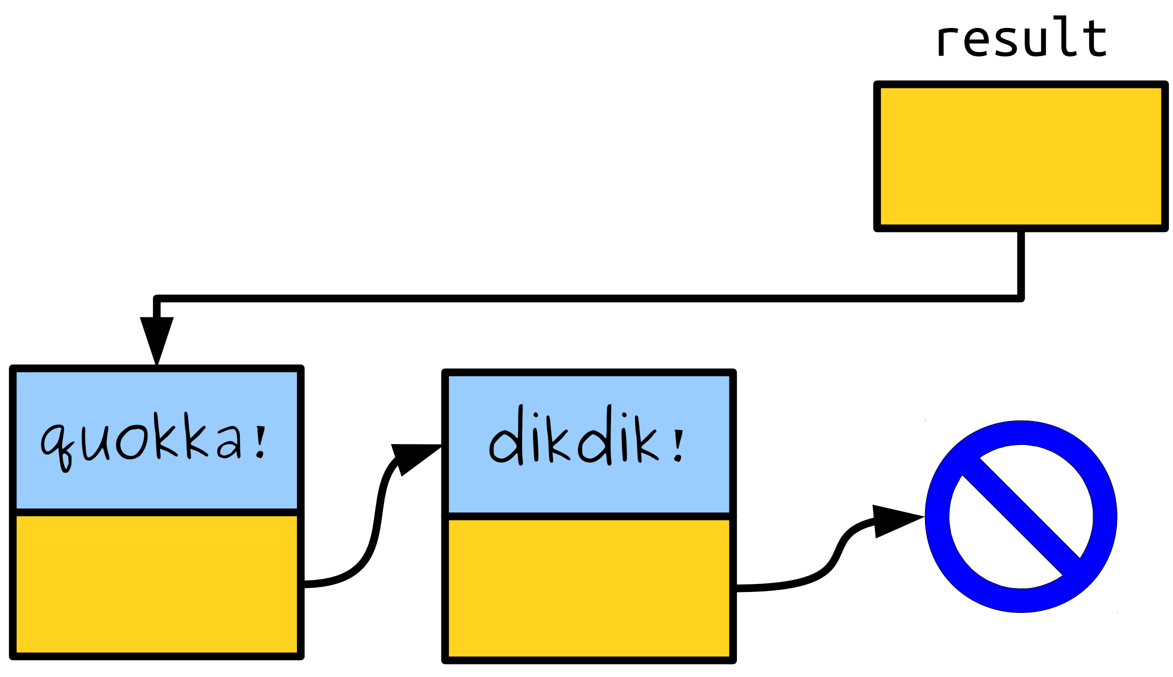
```
Cell* result = nullptr;
while (true) {
   string line = getLine("Next entry? ");
   if (line == "") break;

   Cell* cell = new Cell;
   cell->value = line;

   cell->next = result;
   result = cell;
}
return result;
```



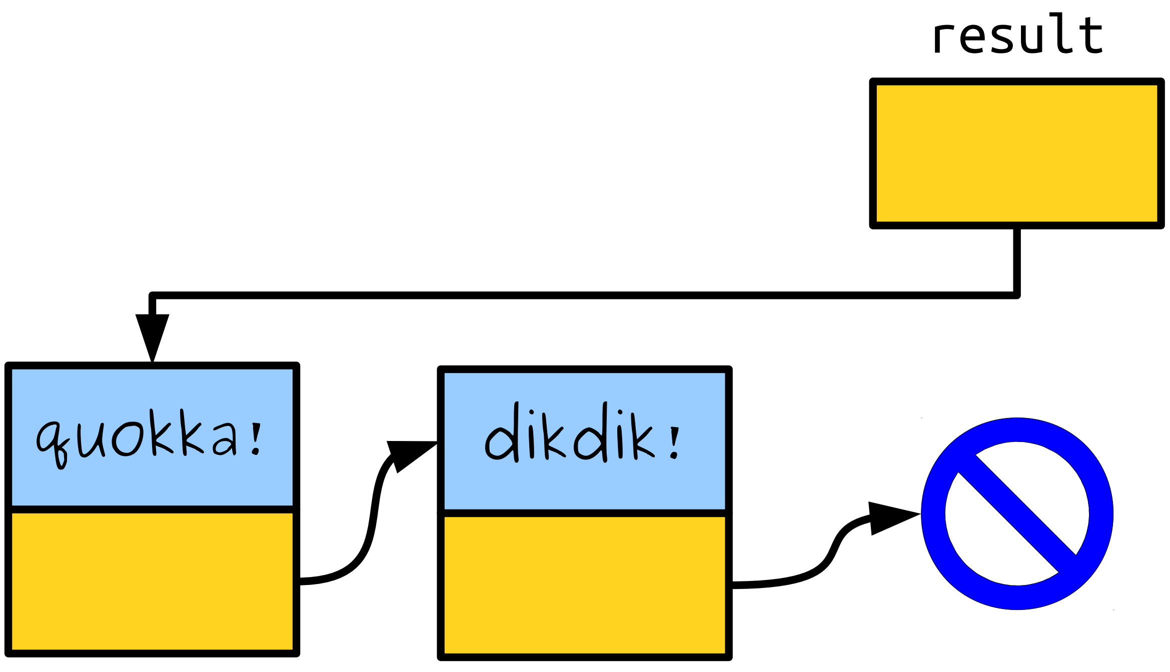
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;
    result = cell;
}
return result;
```



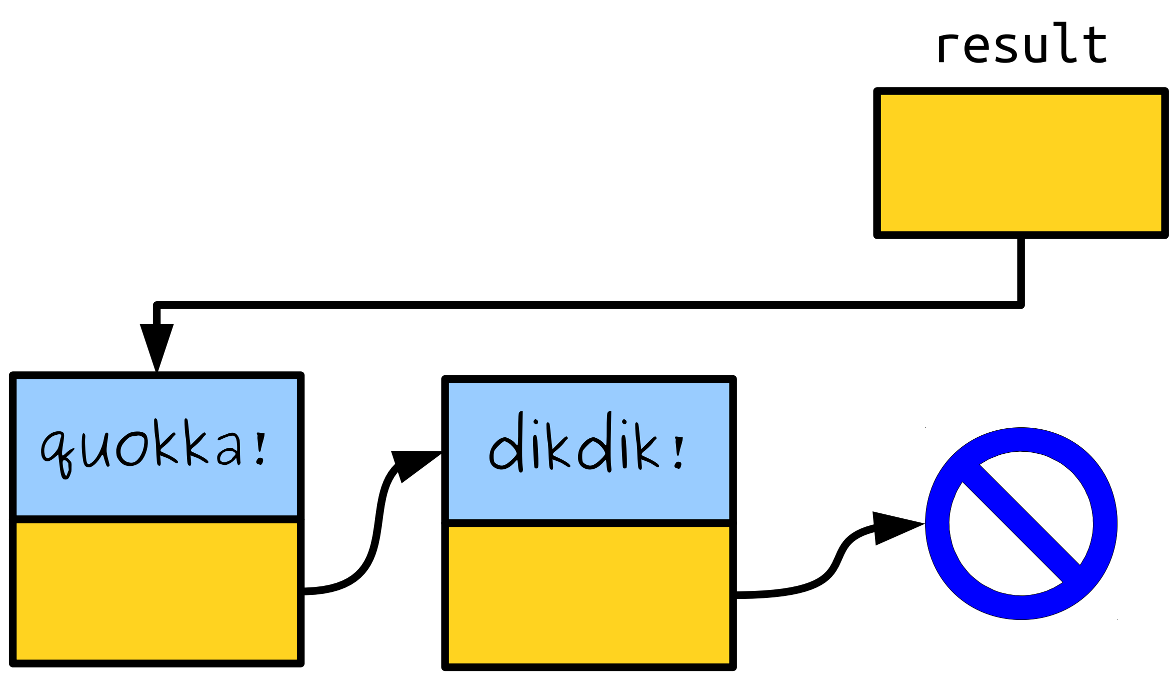
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;
    result = cell;
}
return result;
```



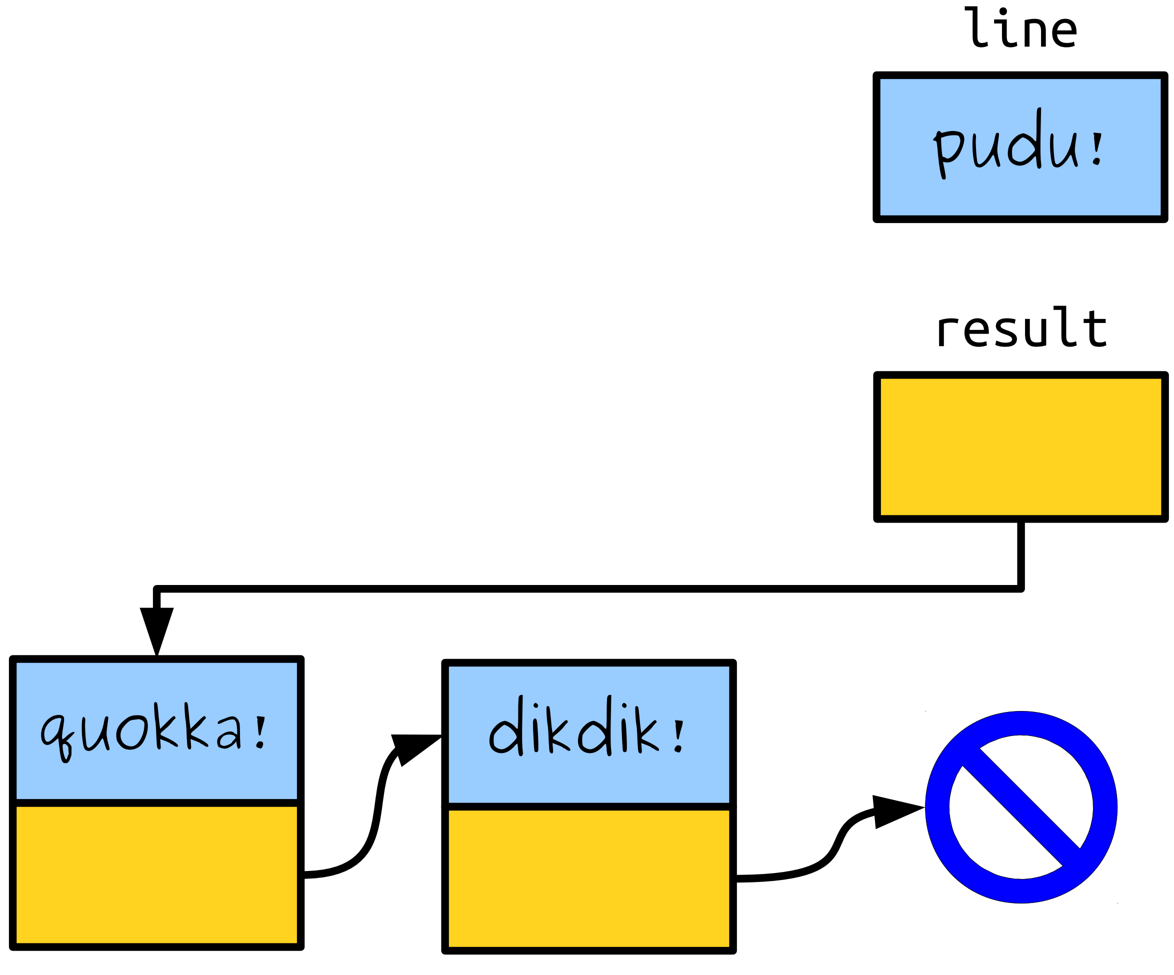
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;
    result = cell;
}
return result;
```



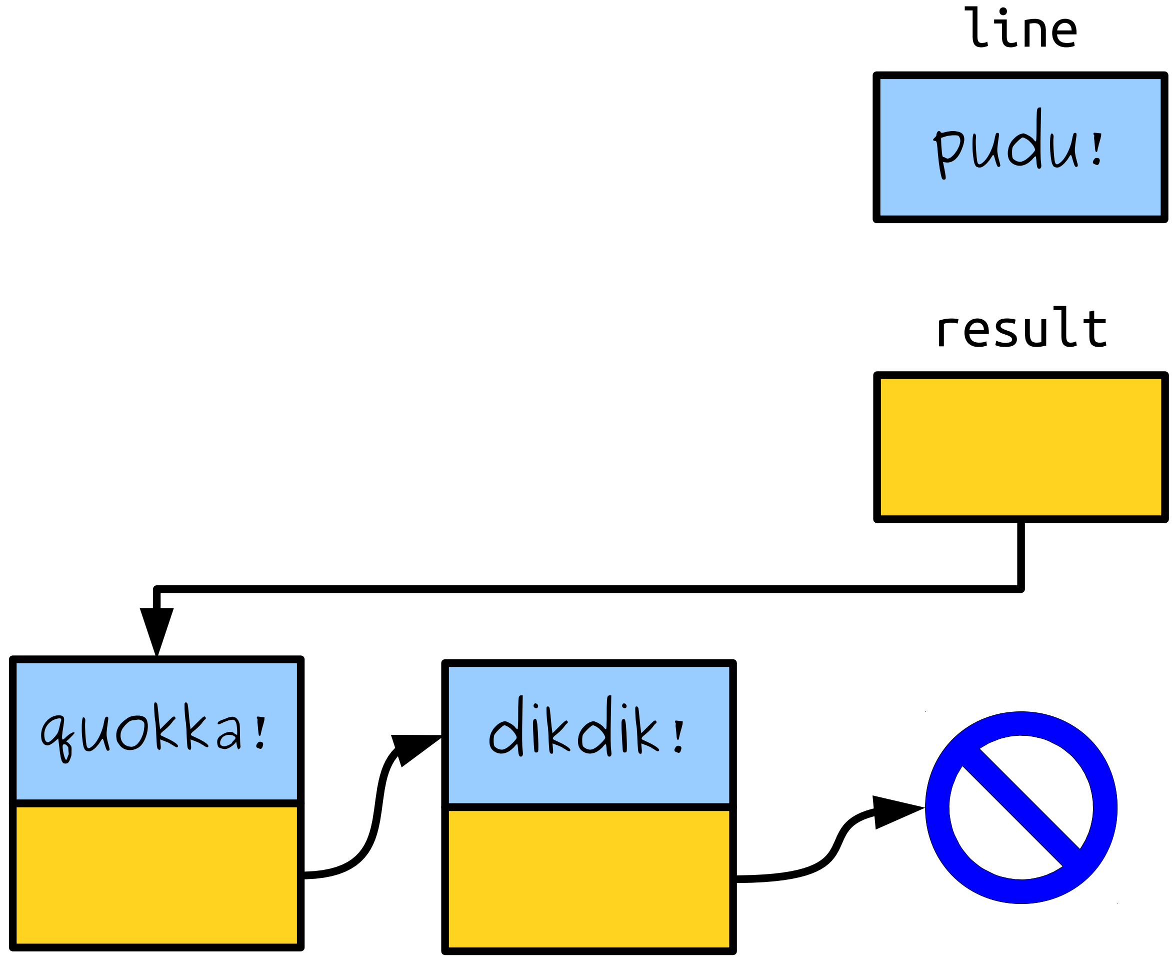
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;
    result = cell;
}
return result;
```



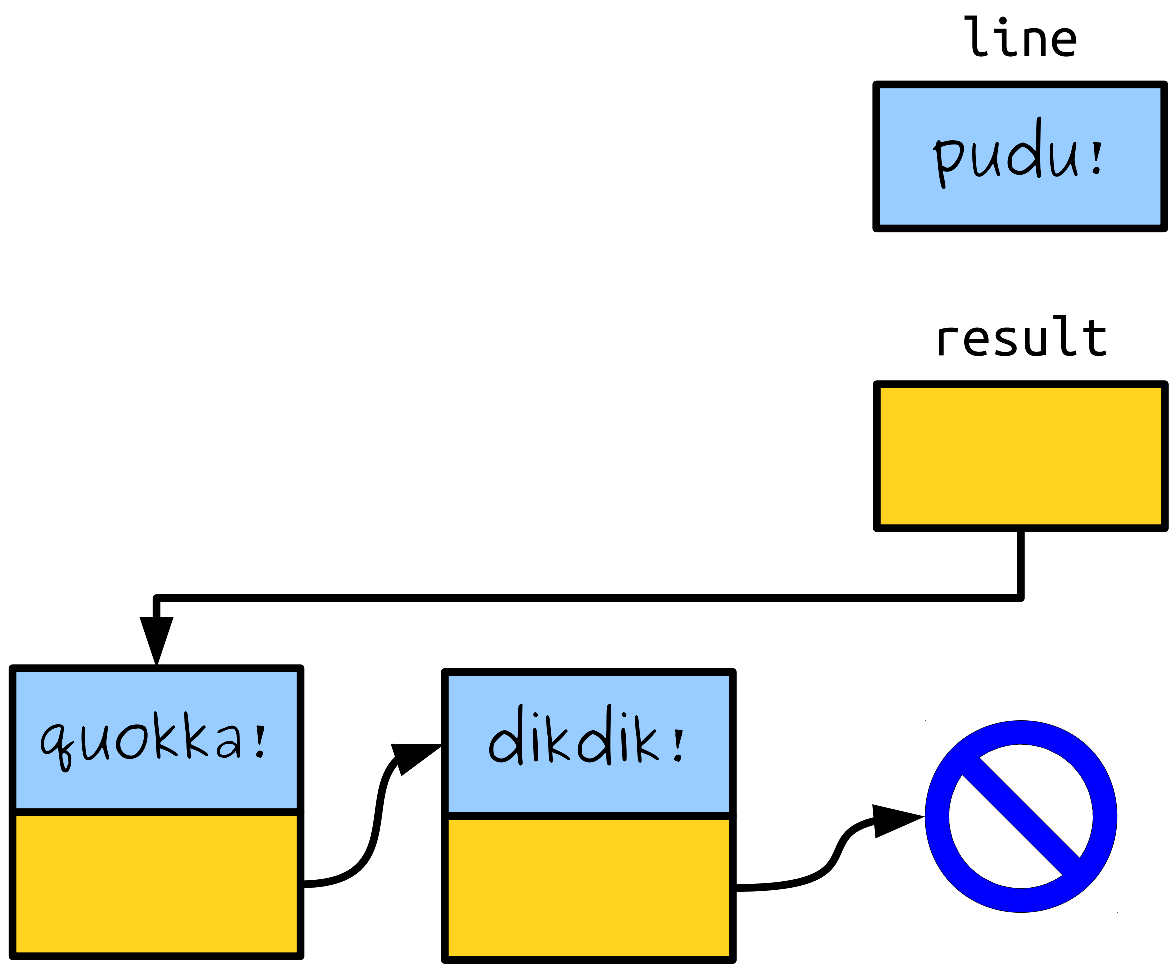
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;
    result = cell;
}
return result;
```



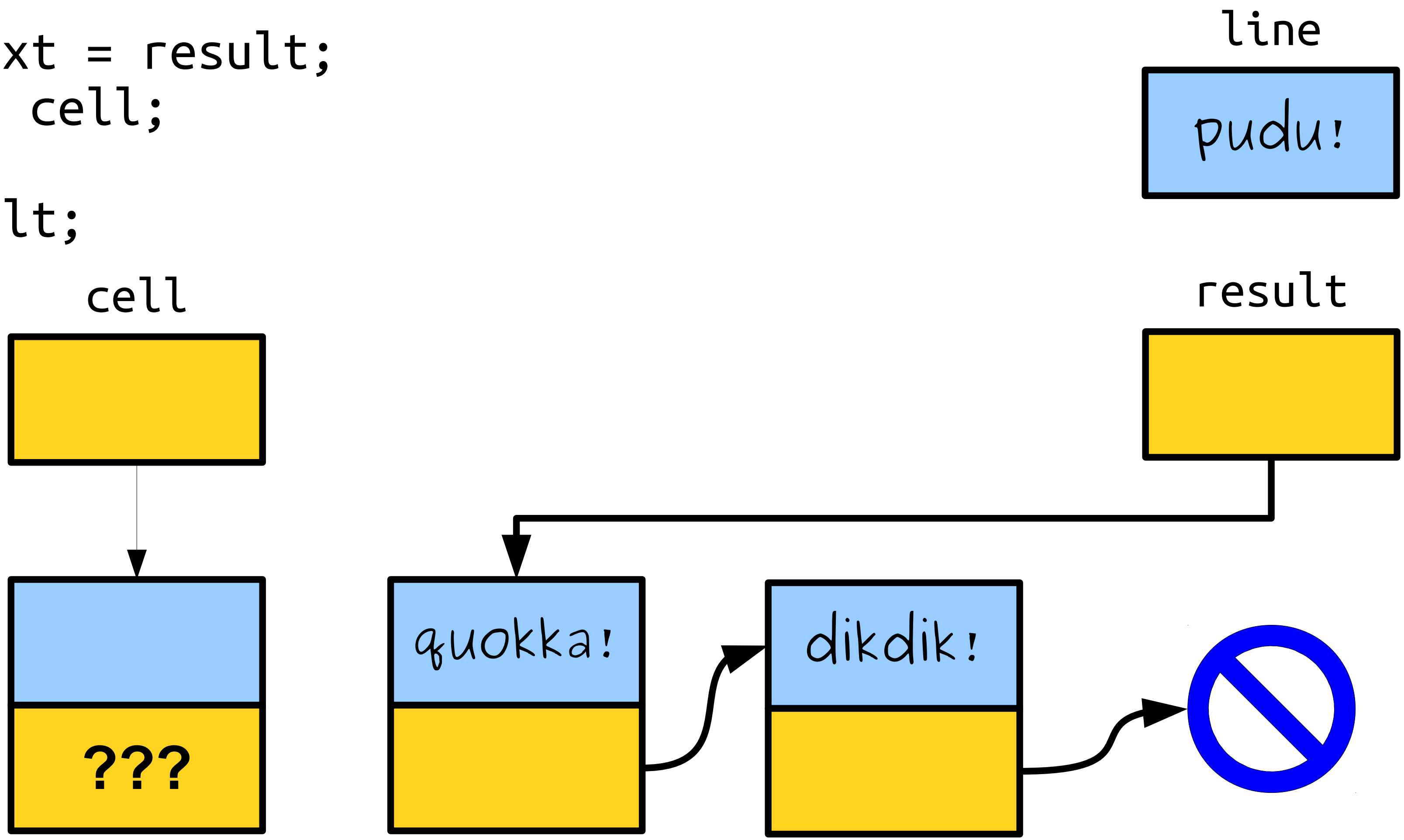
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;
    result = cell;
}
return result;
```



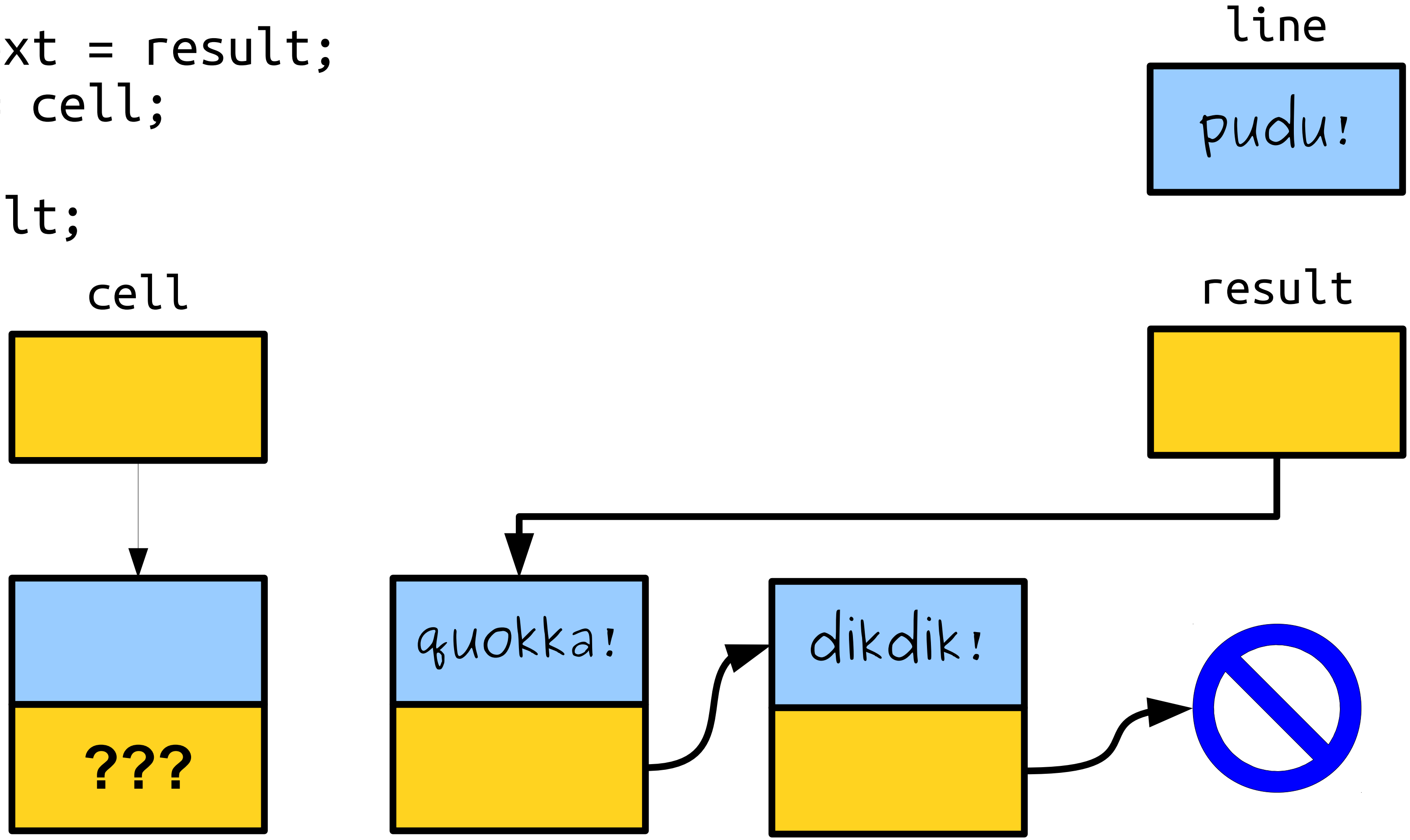
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;
    result = cell;
}
return result;
```



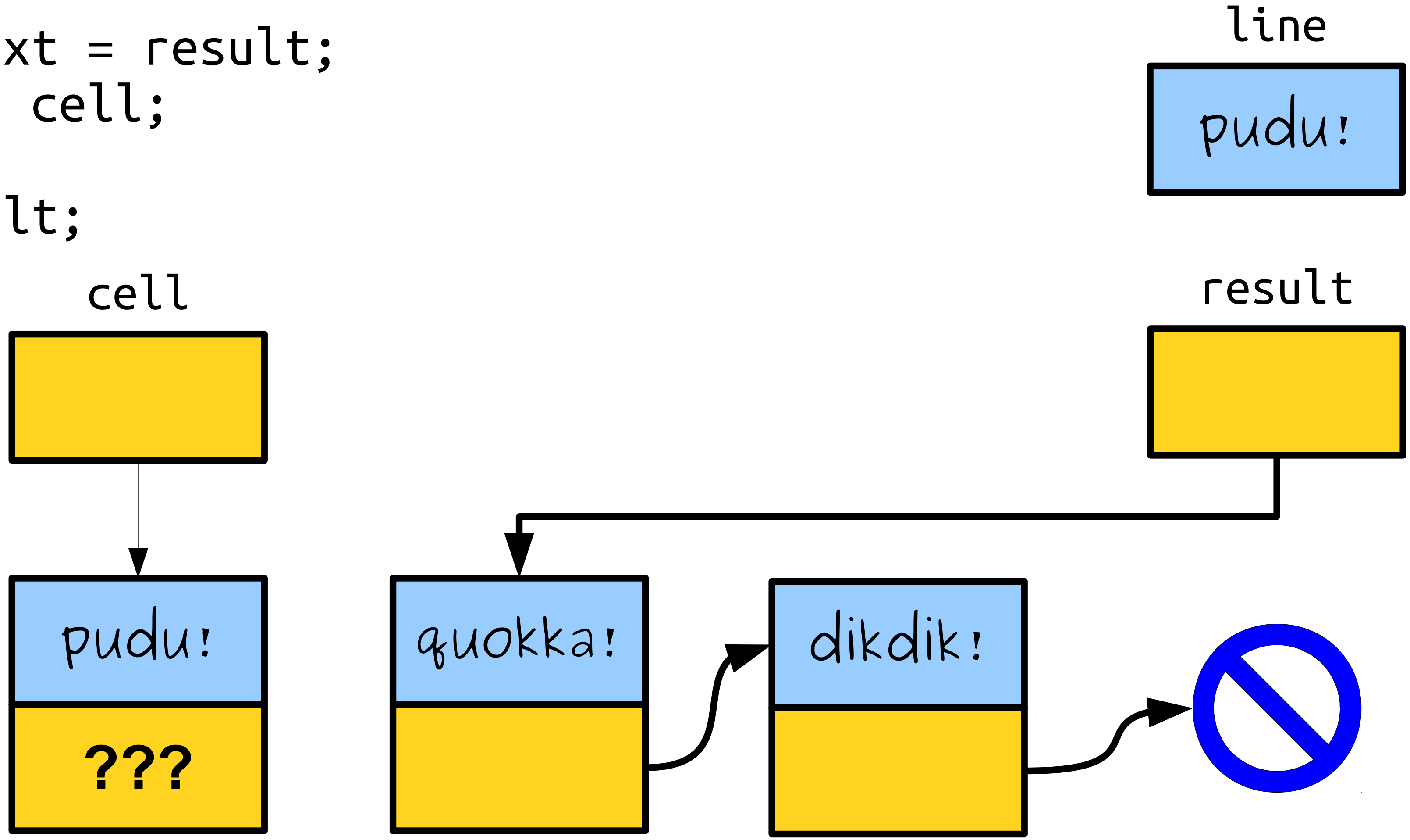
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;
    result = cell;
}
return result;
```



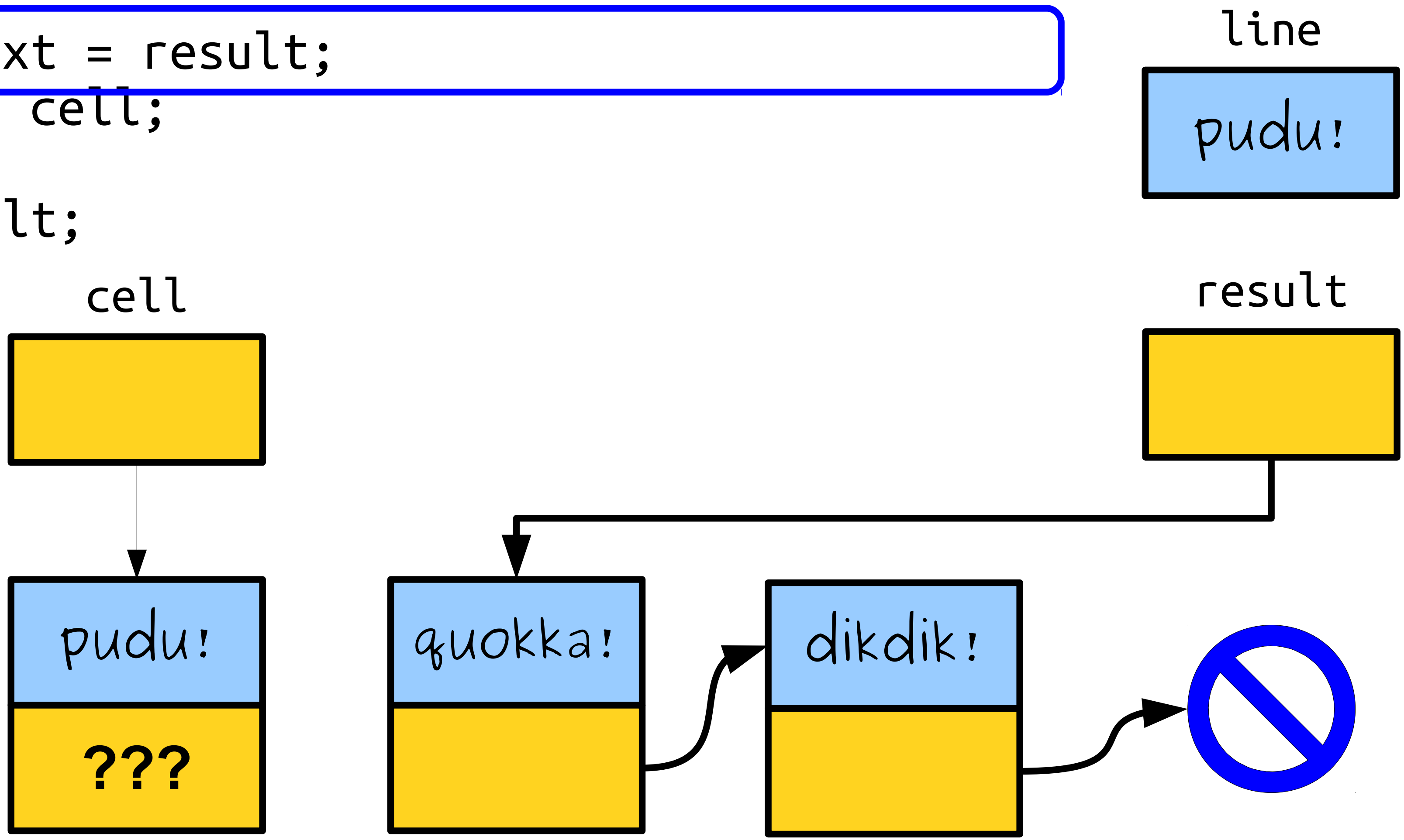
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;
    result = cell;
}
return result;
```



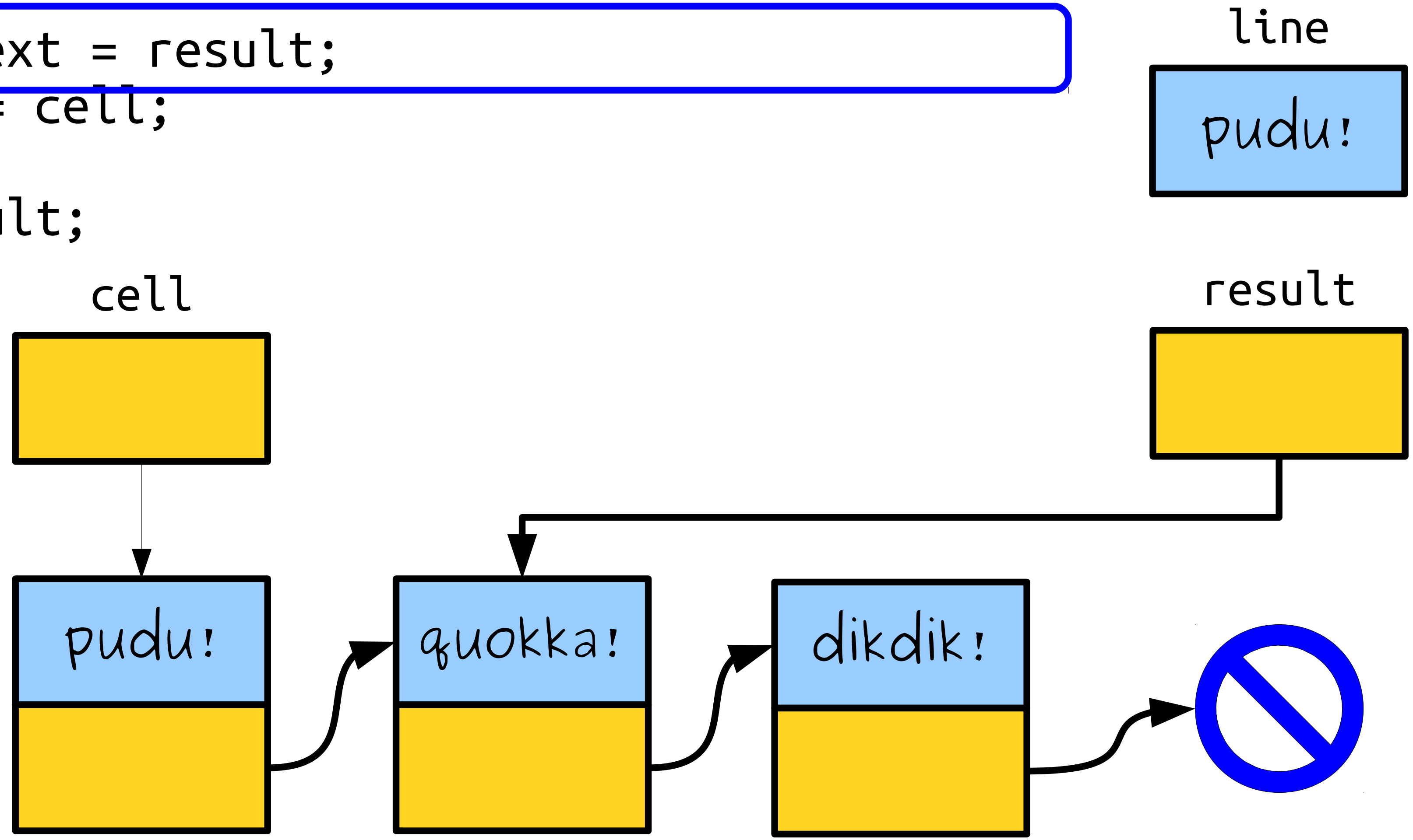
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;
    result = cell;
}
return result;
```



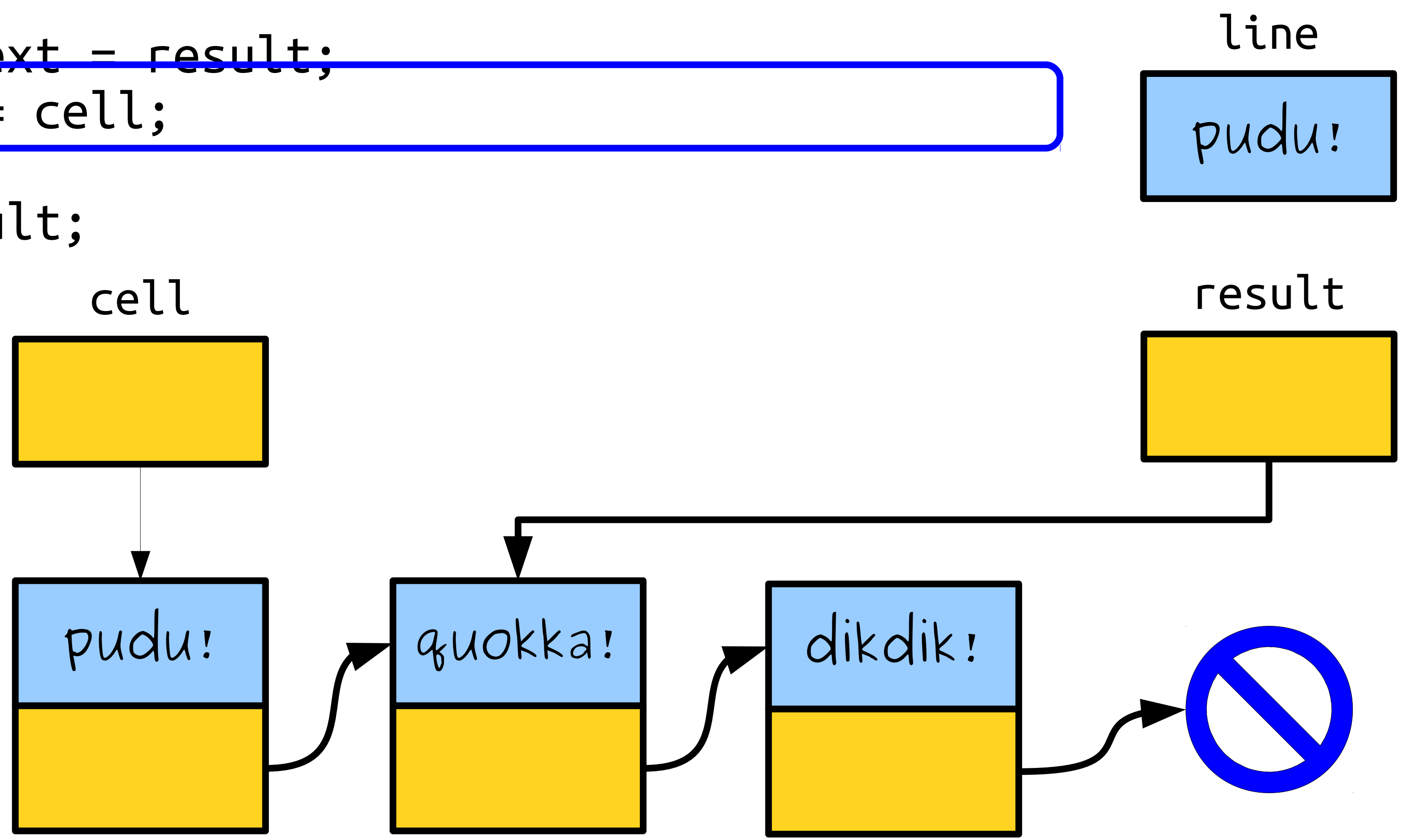
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;
    result = cell;
}
return result;
```



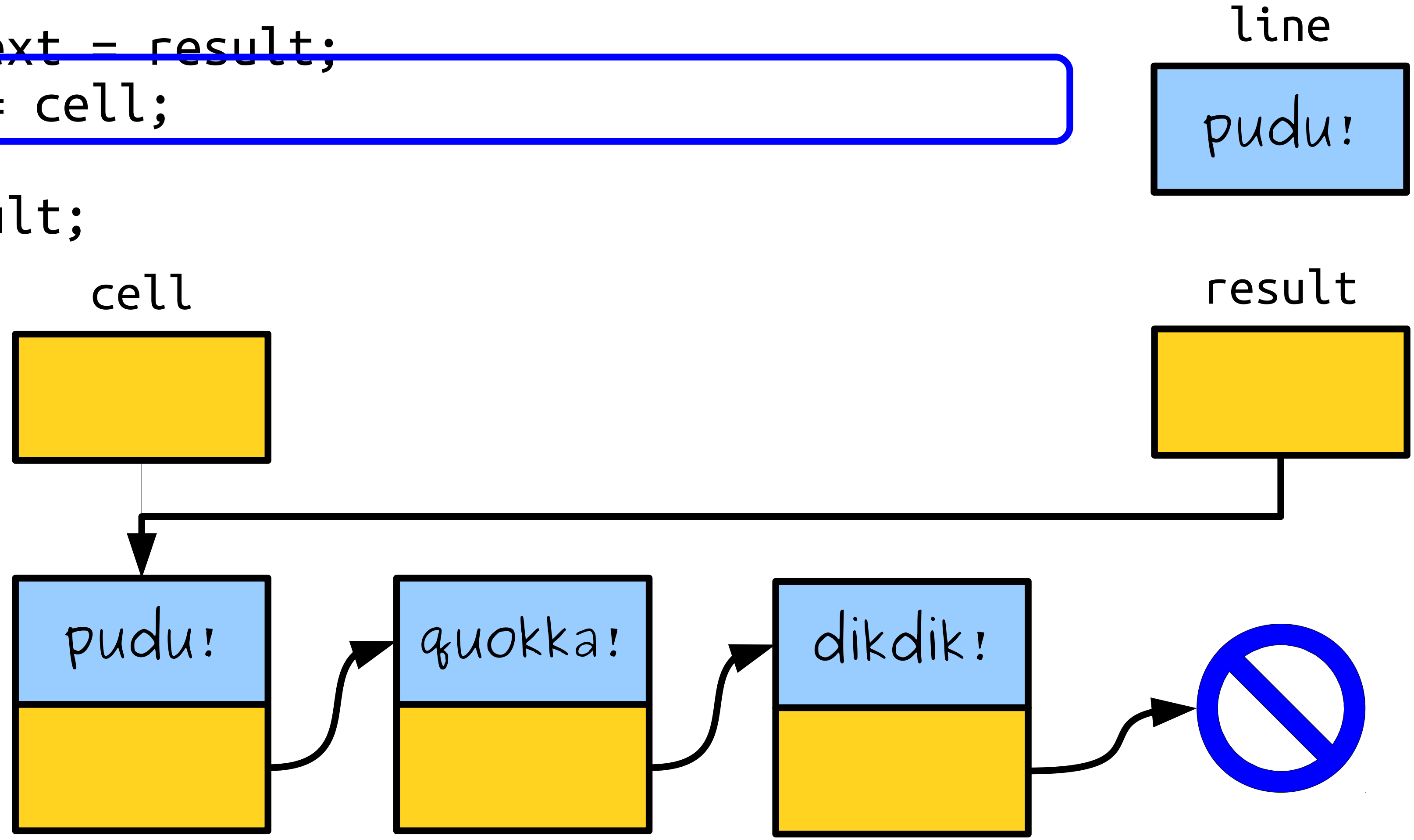
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;
    result = cell;
}
return result;
```



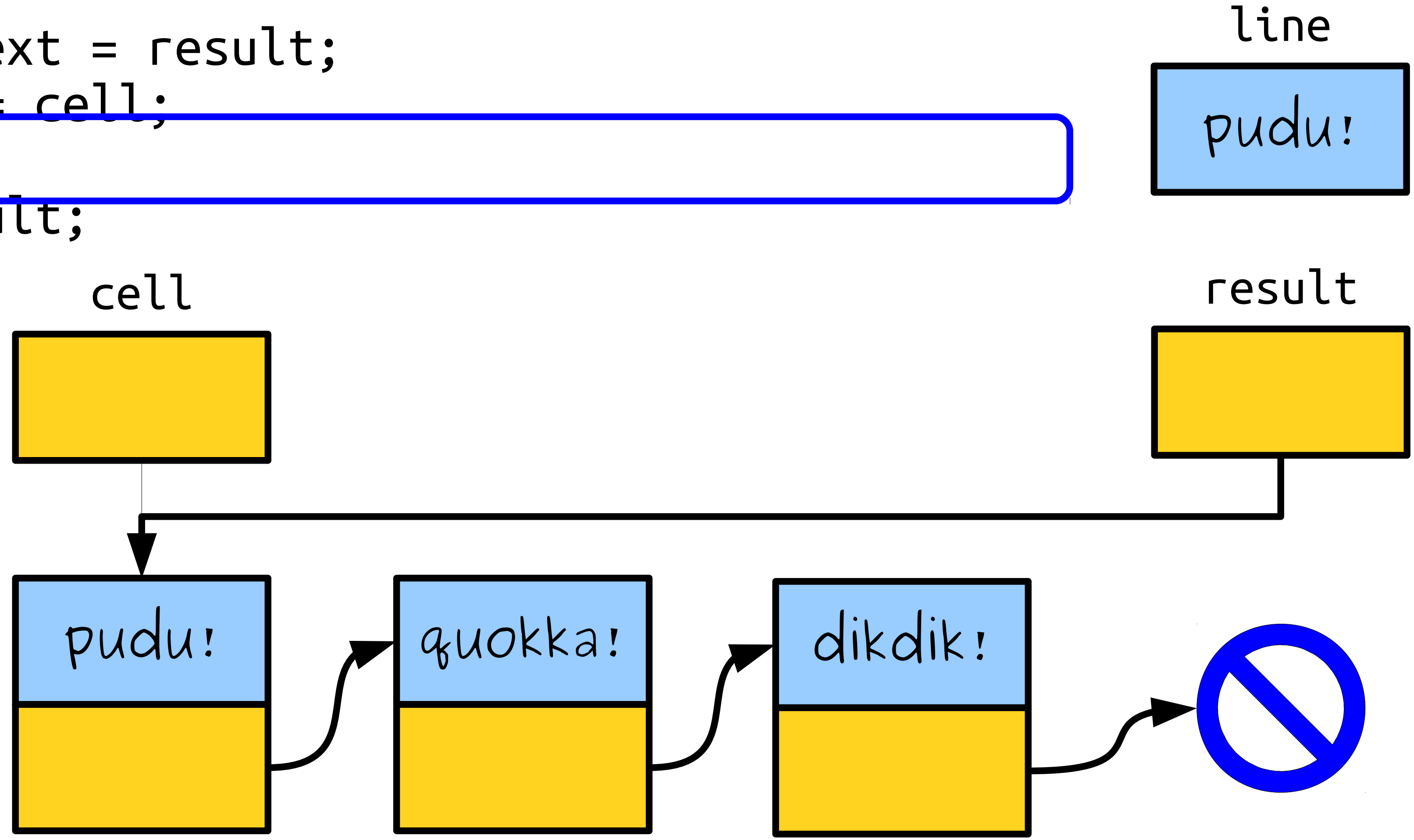
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;
    result = cell;
}
return result;
```



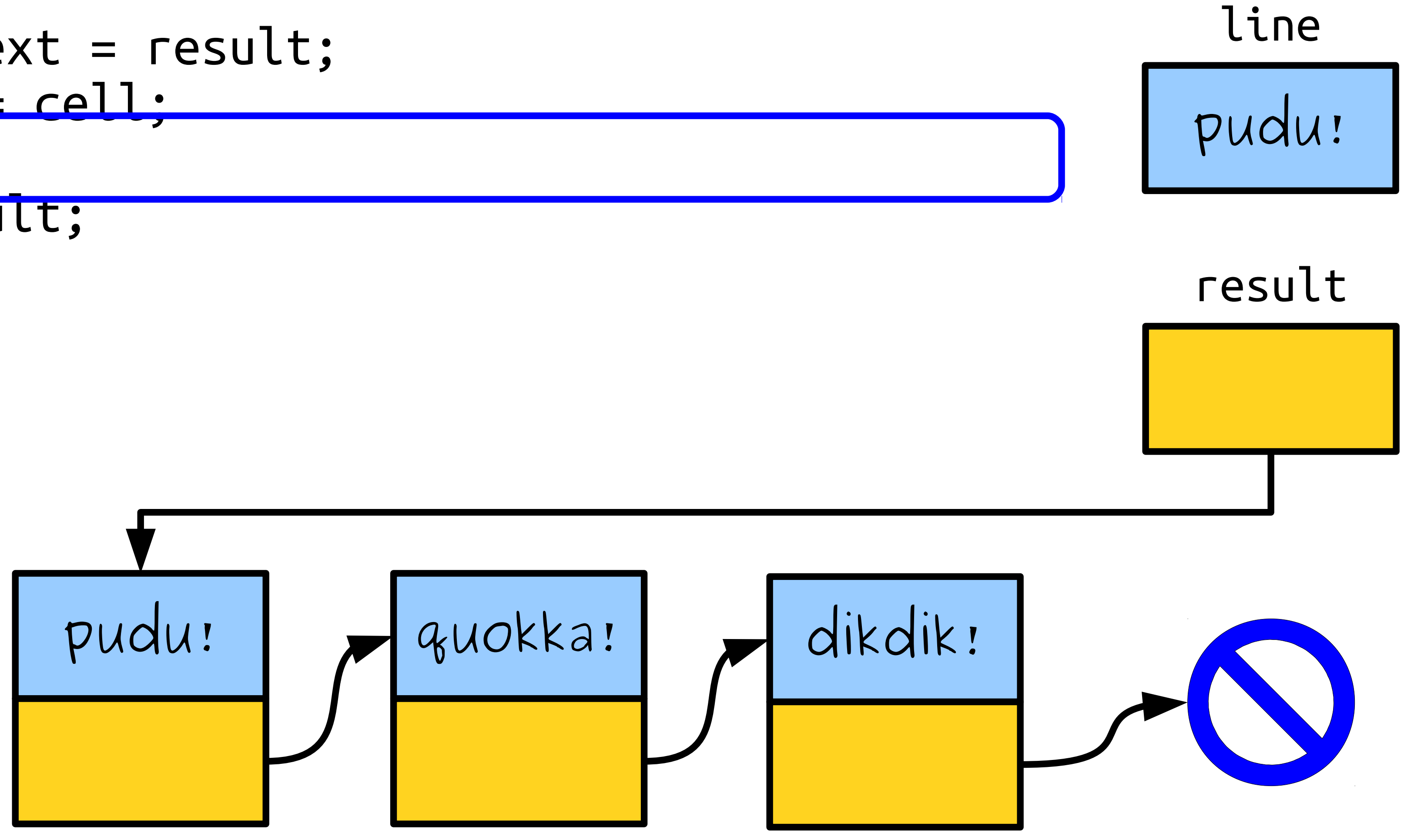
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;
    result = cell;
}
return result;
```



```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;
    result = cell;
}
return result;
```

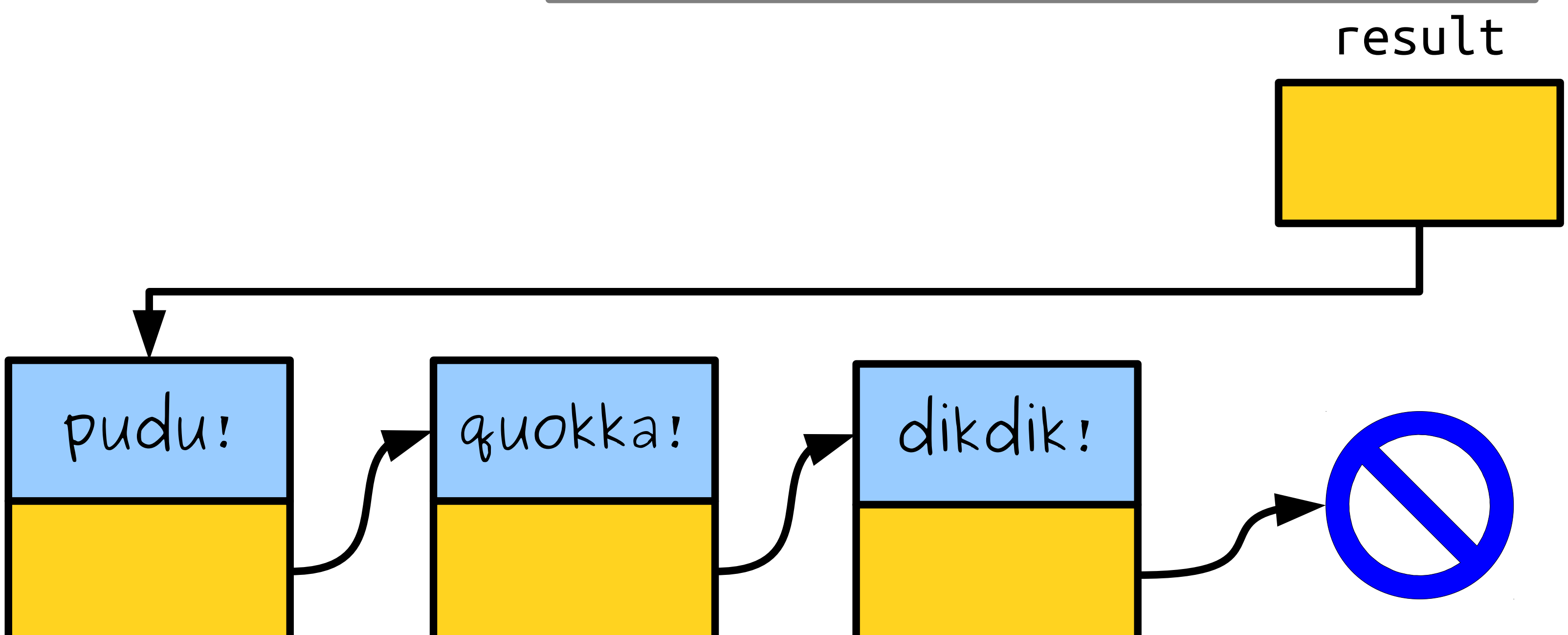


```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;
    cell->value = line;

    cell->next = result;
    result = cell;
}
return result;
```



```
Cell* result = nullptr;
while (true) {
   string line = getLine("Next entry? ");
   if (line == "") break;

   Cell* cell = new Cell;
   cell->value = line;

   cell->next = result;
   result = cell;
}
return result;
```

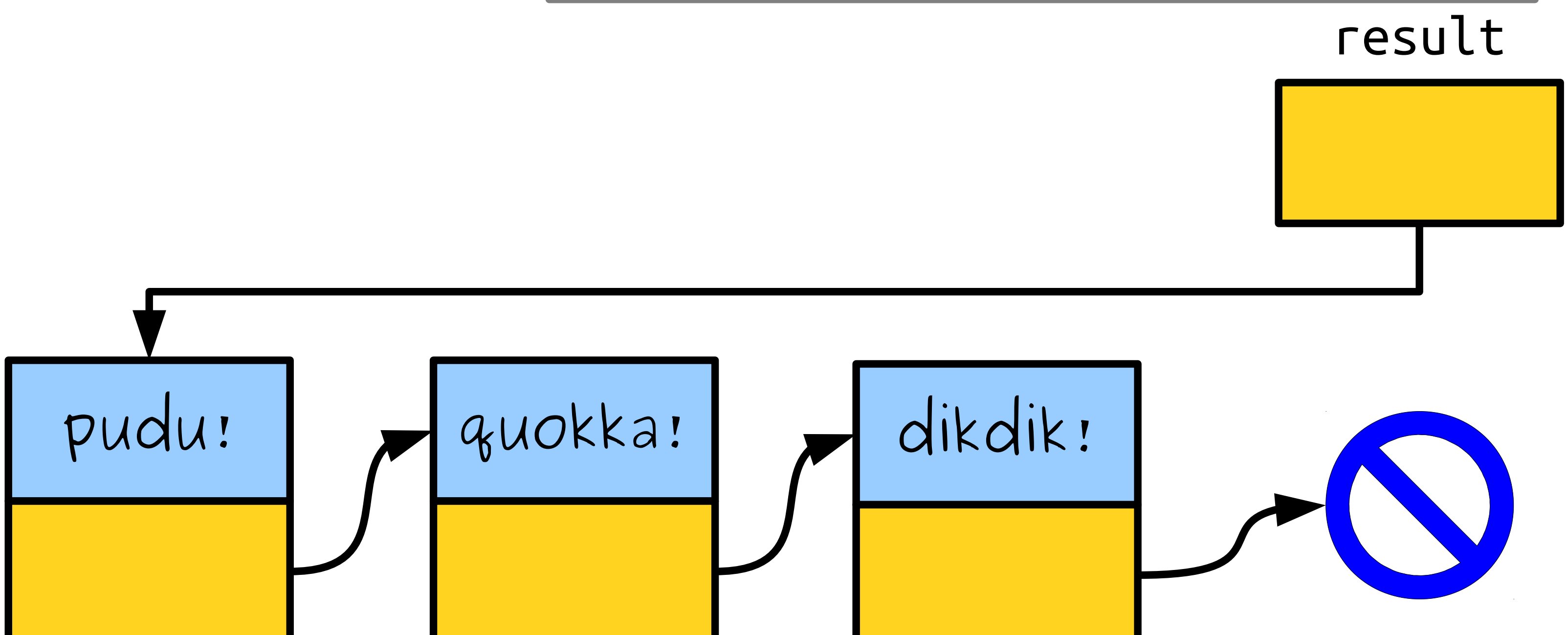
It's a bug: these elements are in the wrong order!



```
Cell* result = nullptr;
while (true) {
   string line = getLine("Next entry? ");
   if (line == "") break;

   Cell* cell = new Cell;
   cell->value = line;

   cell->next = result;
   result = cell;
}
return result;
```

It's a bug: these elements are in the wrong order!

It's a feature: we just implemented a stack using linked lists!



Your Action Items

- ***Read Chapter 12.1 – 12.3.***
  - It’s a good overview of linked lists.
- ***Finish Assignment 6.***
  - Need help? Come talk to us! That’s what we’re here for.

Next Time

- ***Tail Pointers***
  - Tracking the start and end of a list.
- ***Implementing the Queue***
  - A Tale of Two Implementations.
- ***Variations on Linked Lists***
  - What linked lists look like "in the wild."