# Linked Lists

Part One

# Outline for Today

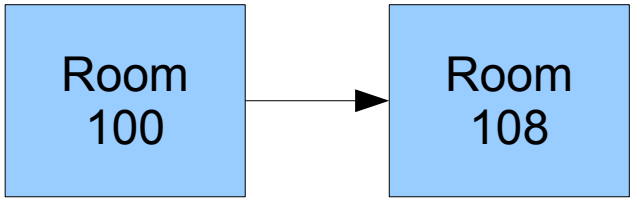
- ***Linked Lists, Conceptually***
  - A different way to represent a sequence.
- ***Linked Lists, In Code***
  - Some cool new C++ tricks.
- ***Manipulating Lists Recursively***
  - … is way easier than you'd expect!
- ***Manipulating Lists Iteratively***
  - … is trickier than you'd expect!

# Changing Offices

# The Sign on Room 100

Dr. Cynthia Lee is no longer in room 100.

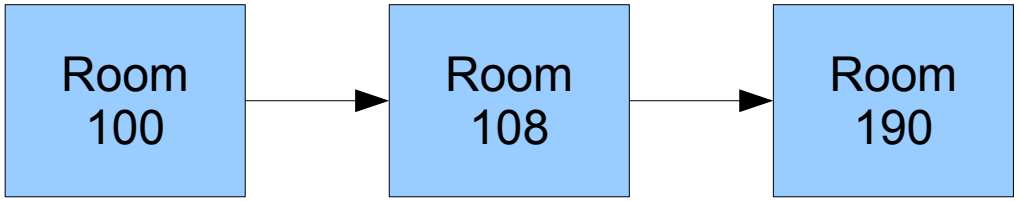
She can be found in room 108.



# The Sign on Room 108

Dr. Cynthia Lee is no longer in room 108.

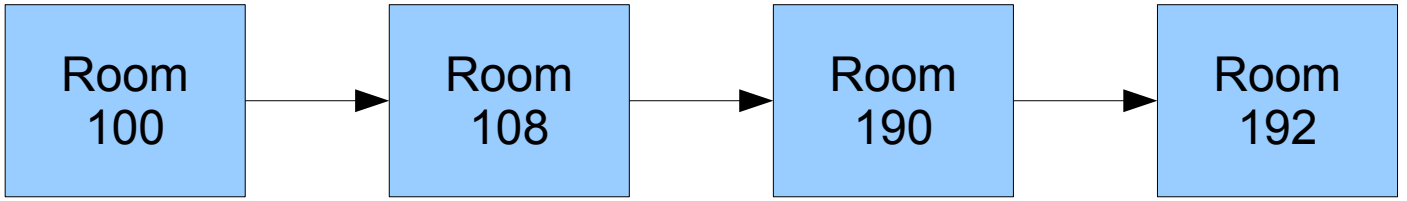
She can be found in room 190.



# The Sign on Room 190

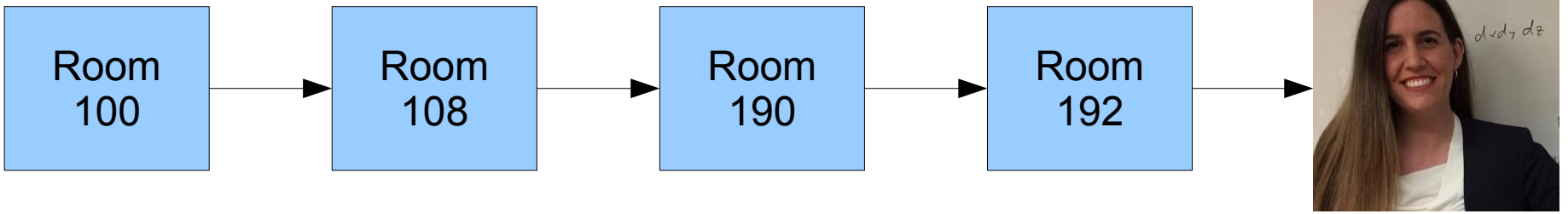
Dr. Cynthia Lee is no longer in room 190.

She can be found in room 192.



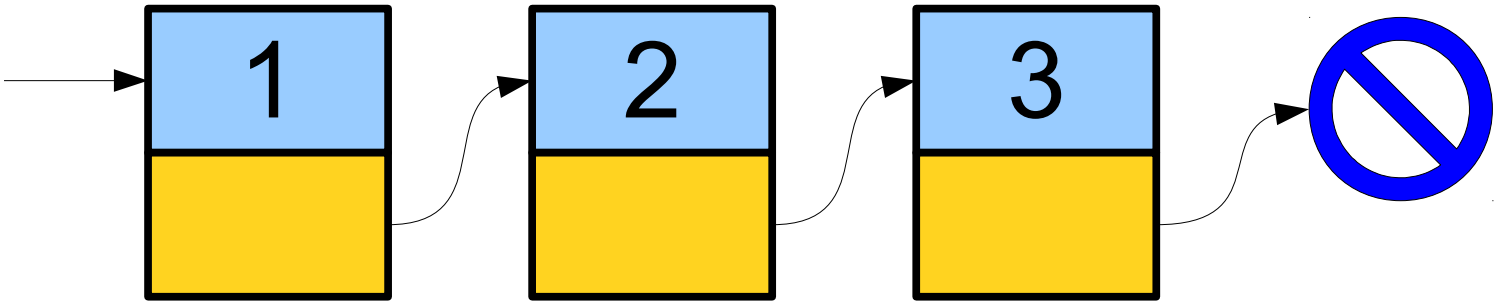
# The Sign on Room 192

Welcome to Cynthia's Office!



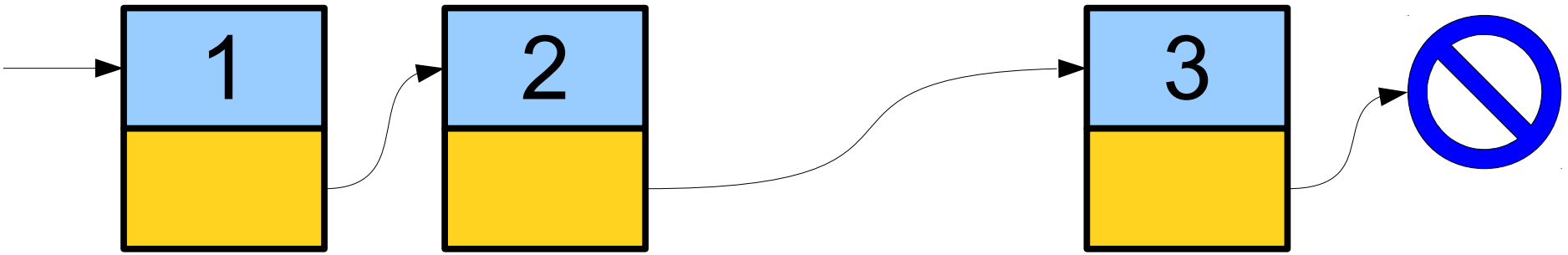
# Linked Lists at a Glance

- 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.



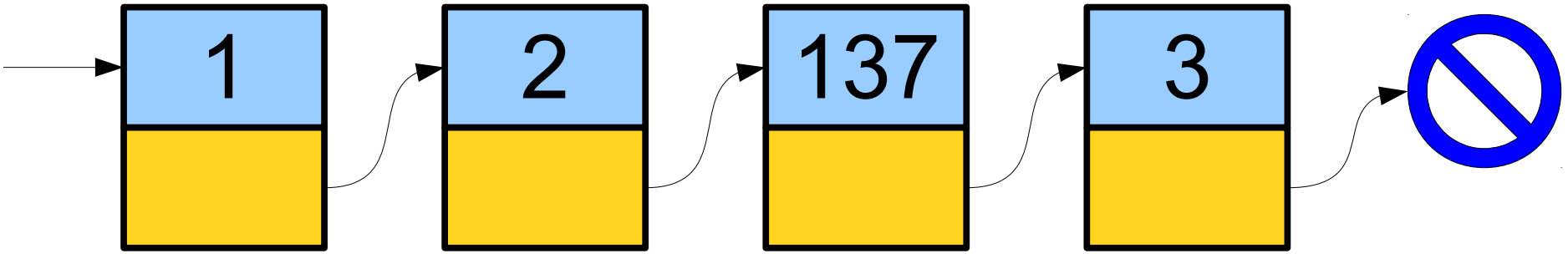
# Linked Lists at a Glance

- 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.



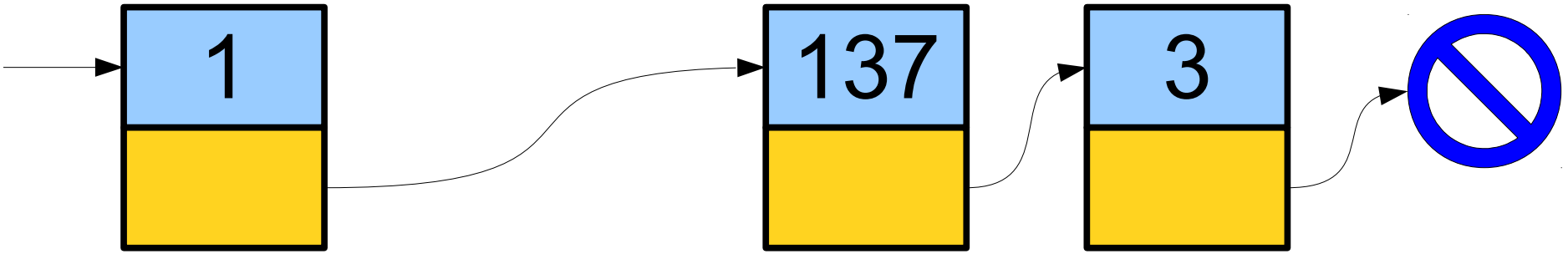
# Linked Lists at a Glance

- 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.



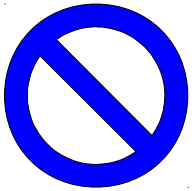
# Linked Lists at a Glance

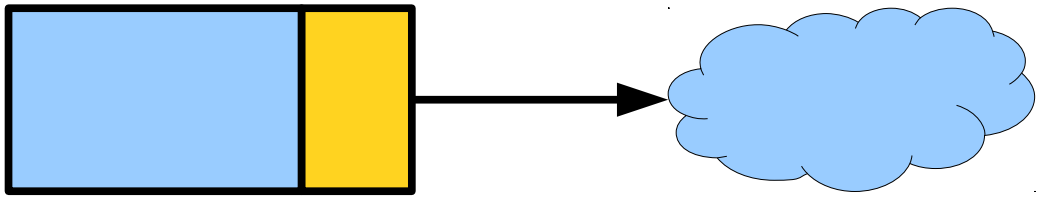
- A ***linked list*** is a data structure for storing a sequence of elements.
- Each element is stored separately from the rest.
- The elements are then chained together into a sequence.
- The end of the list is marked with some special indicator.



# A Linked List is Either…

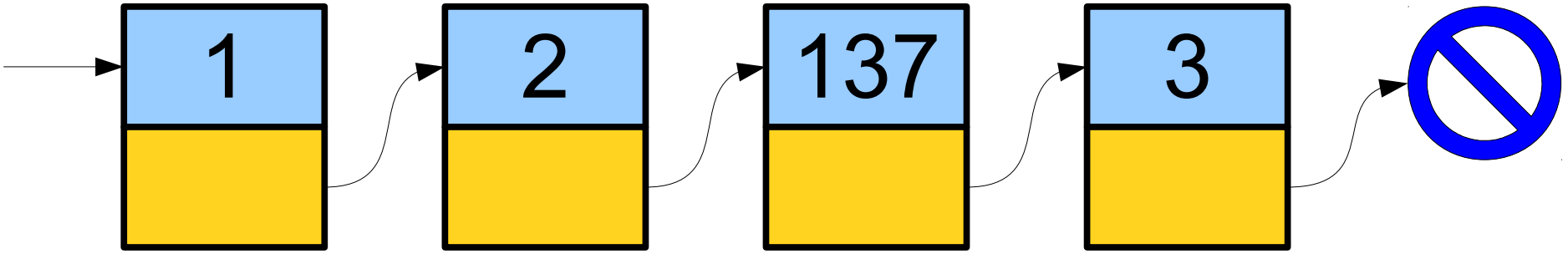
…an empty list,
or…





a single cell…

… that points at another linked list.

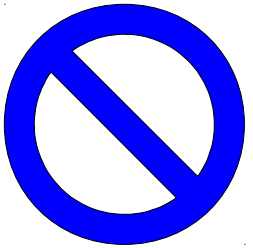


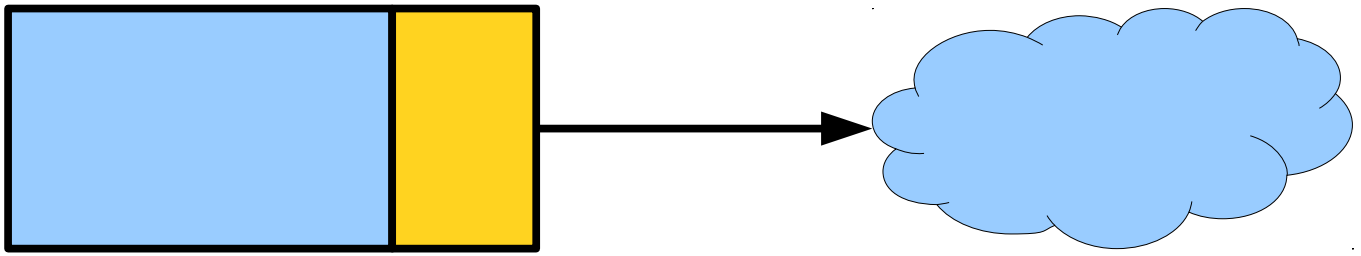
# Representing Linked Lists

# A Linked List is Either…

…an empty list,

or…





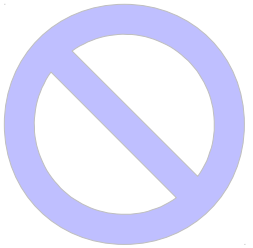
a single cell…

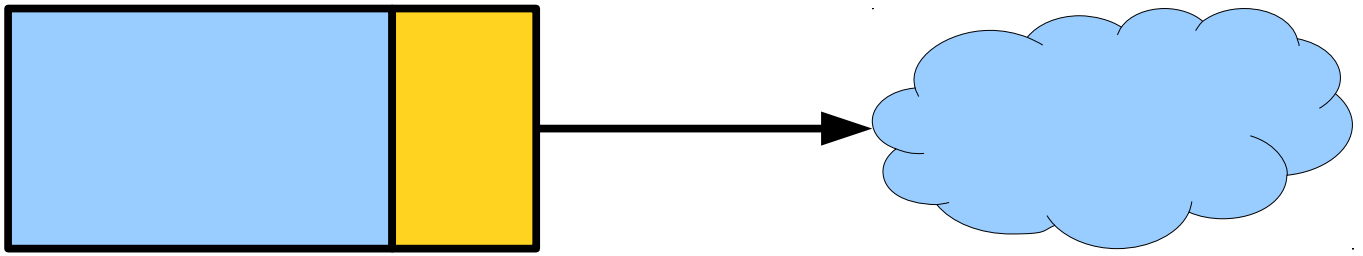
… that points at another linked list.

# A Linked List is Either…

…an empty list,

or…



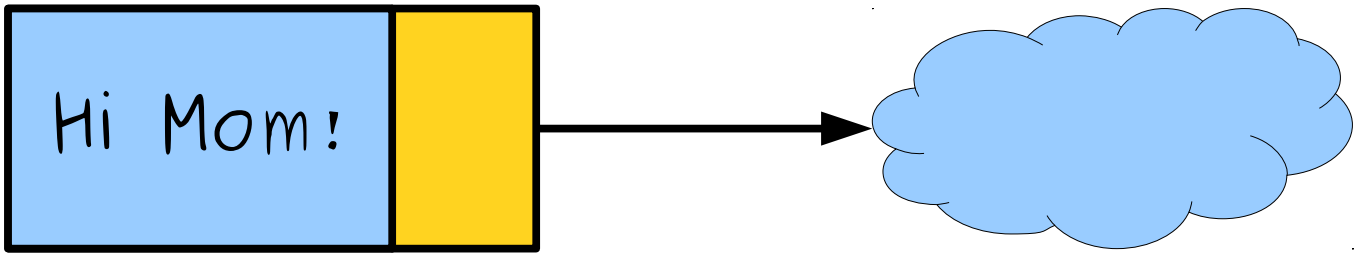


a single cell…

… that points at another linked list.

# A Linked List is Either…

```
struct Cell {
    string value;
    Cell* next;
};
```



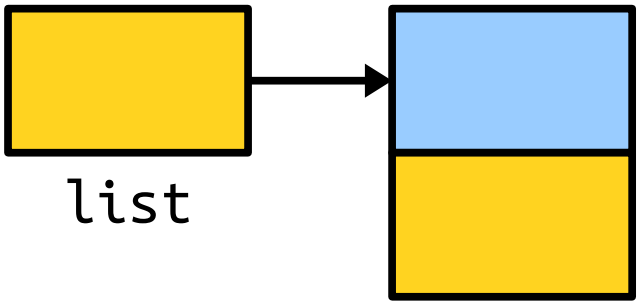

a single cell…

… that points at another linked list.

```
struct Cell {
    string value;
    Cell* next;
};

Cell* list = new Cell;
```

We just want a single cell, not an array of cells. To get the space we need, we'll just say `new Cell`.

Notice that `list` is still a `Cell*`, a pointer to a cell. It still says "look over there for your `Cell`" rather than "I'm a `Cell`!"

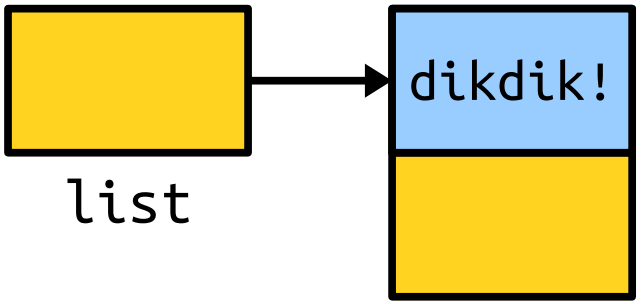



Yes, it's a bit confusing that C++ uses the same types to mean "look over there for an array of `Cells`" and "look over there for a single `Cell`."

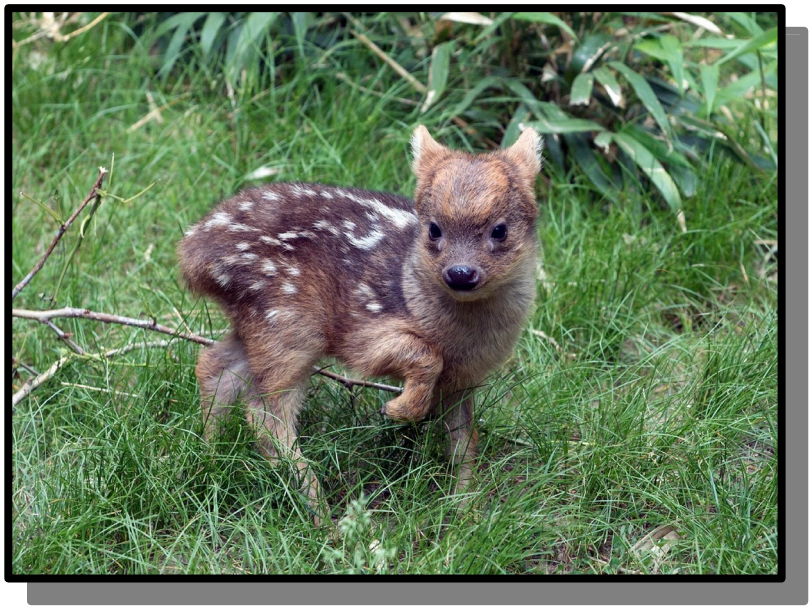
```
struct Cell {
    string value;
    Cell* next;
};

Cell* list = new Cell;
list->value = "dikdik!";
```

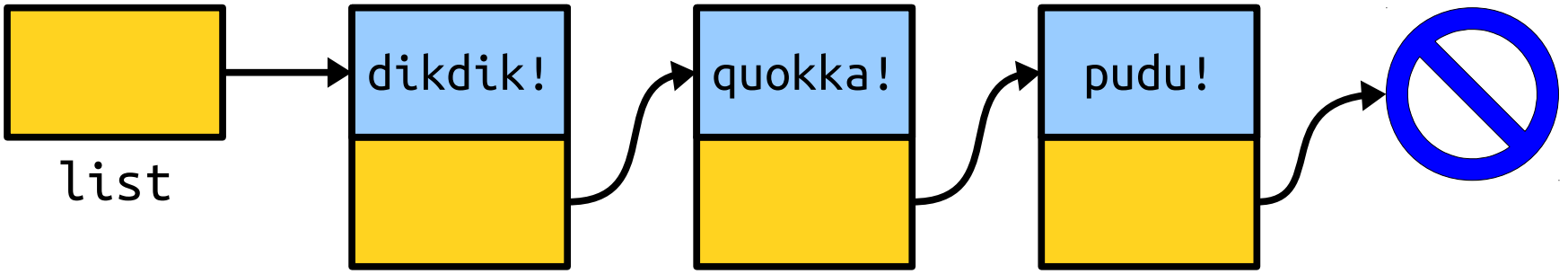
Because list is a pointer to a `Cell`, we use the arrow operator `->` instead of the dot operator.

Think of `list->value` as saying “start at `list`, follow an arrow, then pick the `value` field.”



```
struct Cell {
    string value;
    Cell* next;
};

Cell* list = new Cell;
list->value = "dikdik!";
list->next = new Cell;
list->next->value = "quokka!";
list->next->next = new Cell;
list->next->next->value = "pudu!";
list->next->next->next = nullptr;
```

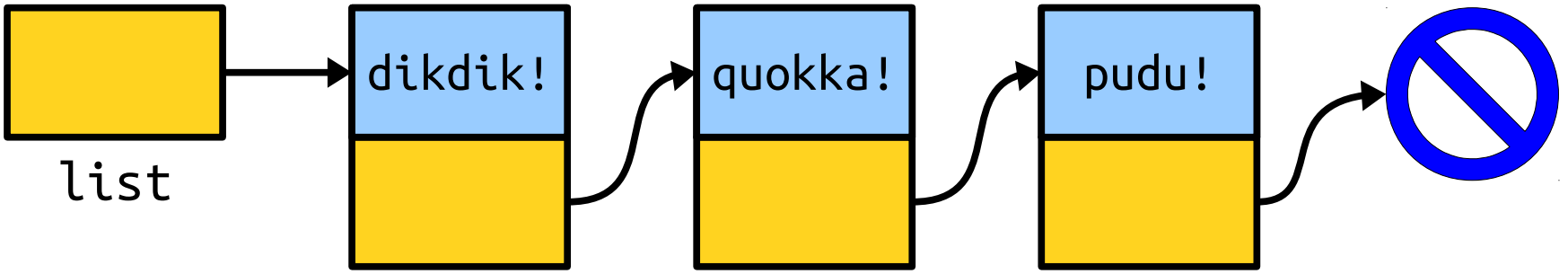




```
struct Cell {
    string value;
    Cell* next;
};

Cell* list = new Cell;
list->value = "dikdik!";
list->next = new Cell;
list->next->value = "quokka!";
list->next->next = new Cell;
list->next->next->value = "pudu!";
list->next->next->next = nullptr;
```

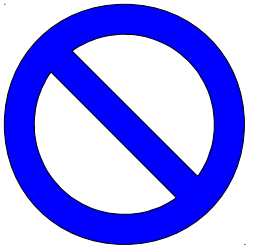
C++ uses the **nullptr** keyword to mean “a pointer that doesn’t point at anything.”

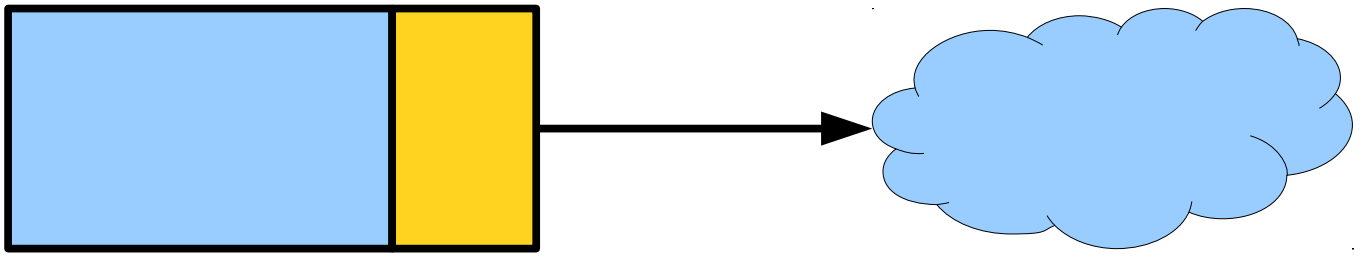
(Older code uses NULL instead of **nullptr**; that’s also okay, but we recommend **nullptr**.)



# A Linked List is Either…

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

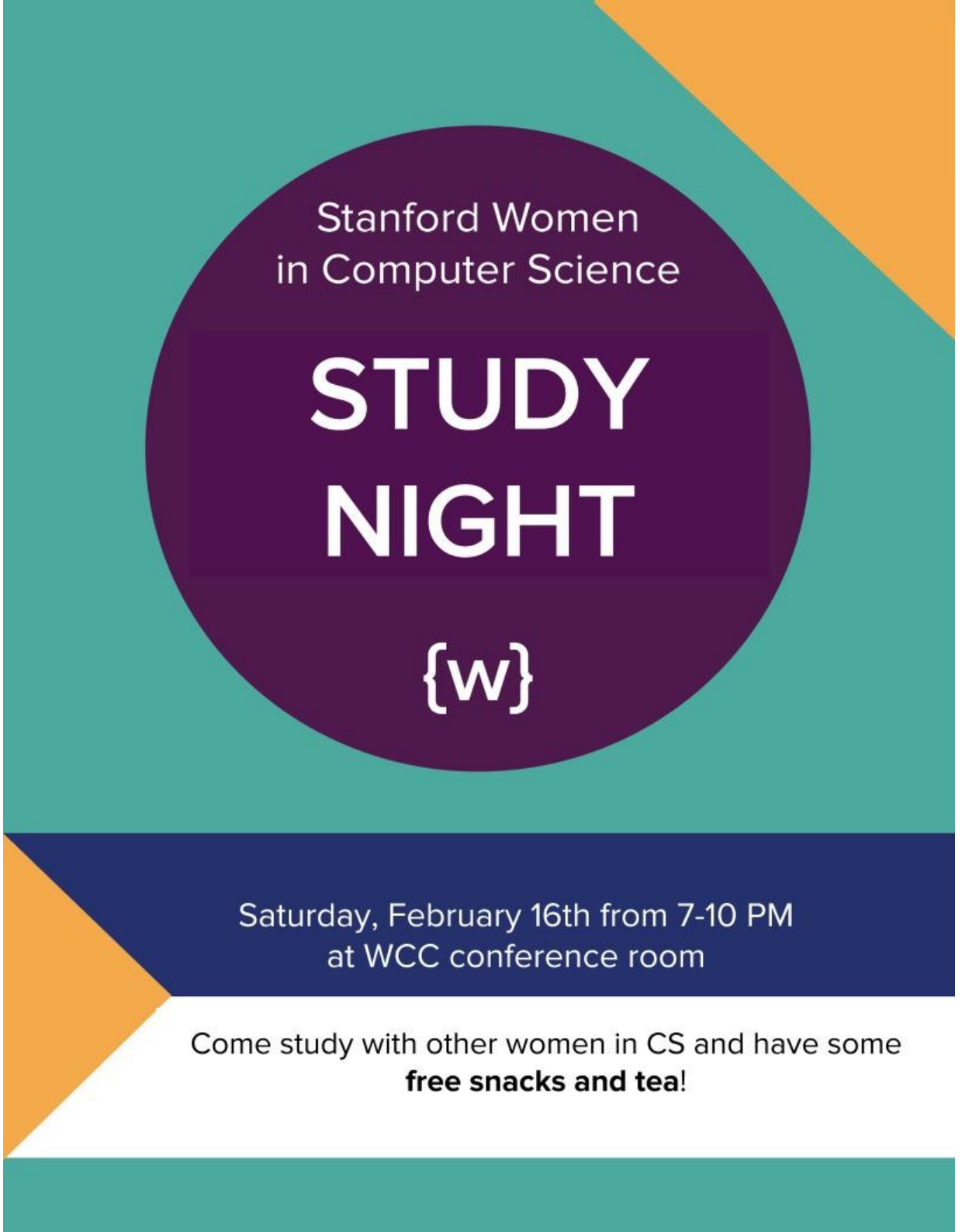




a single linked list cell that points…

… at another linked list.

Time-Out for Announcements!


Stanford Women
in Computer Science
STUDY
NIGHT
{w}
Saturday, February 16th from 7-10 PM
at WCC conference room
Come study with other women in CS and have some
**free snacks and tea**!

# Assignment 5

- Assignment 4 was due at the start of class today.
- Assignment 5 (***Data Sagas***) goes out today. It's due Wednesday, February 27th.
  - Play around with searching, sorting, big-O notation, and class design!
  - Discover some cool patterns in real data sets!
- YEAH Hours are today at 3:30PM in room 380-380Y. Slides will be posted in case you can't make it.

# Midterm Exam

- The midterm exam is next ***Tuesday, February 19*** from ***7:00PM – 10:00PM***. Locations are divvied up by last (family) name:
  - A – K: Go to ***Bishop Auditorium***
  - L – Z: Go to ***Hewlett 200***
- It covers topics from Lectures 01 – 12 (up through and including big-O notation) and Assignments 0 – 4.
- The exam is closed-book and limited-note. You may bring one double-sided sheet of 8.5” × 11” of notes to the exam with you.

# Midterm Exam

- We will be administering the exam using a software tool called ***BlueBook***.
- Visit the CS106B website, click the “BlueBook” link under the “Resources” tab, then download the BlueBook software.
- If you need a laptop for the exam and haven’t contacted us yet, please do so ASAP so we can plan accordingly.

# Practice Midterm Exam

- There's a practice midterm exam up on the course website. It's a minimally-modified version of the exam we gave out in Winter 2017.
- The password is

  **`maplesyrup`**

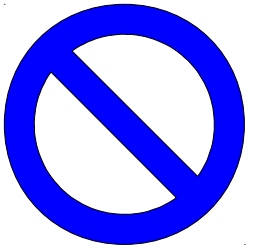
  and you'll see why when you start the exam. 😀

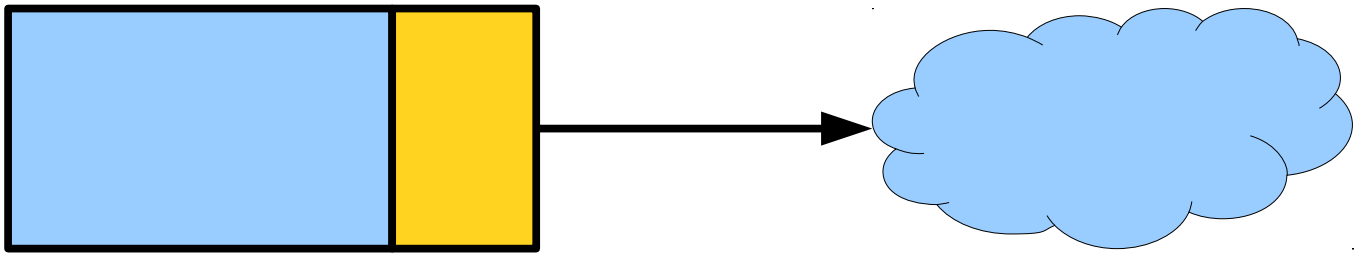
Back to CS106B!

# Printing a Linked List

# A Linked List is Either…

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



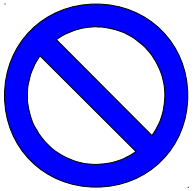


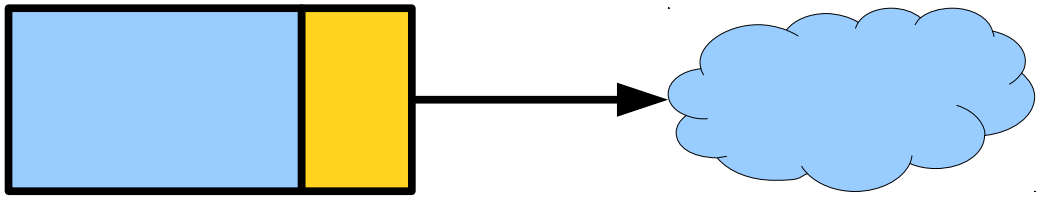
a single linked list cell that points…

… at another linked list.

# A Linked List is Either…

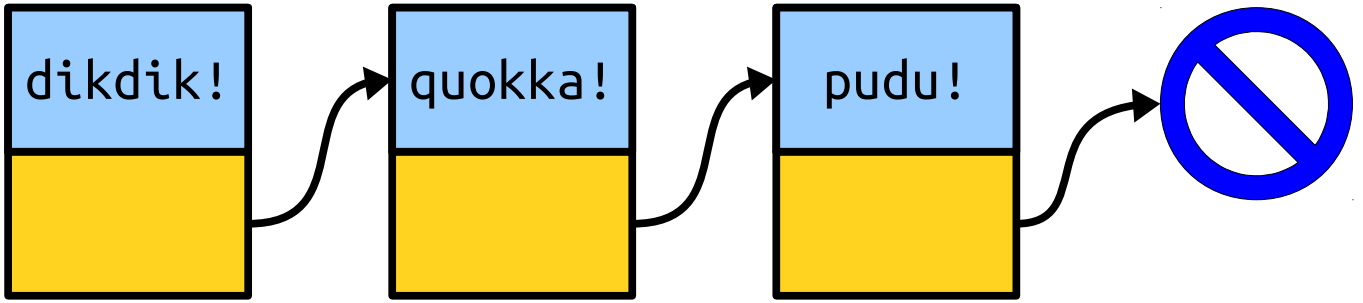
…an empty list, represented by `nullptr`, or…





a single linked list cell that points…

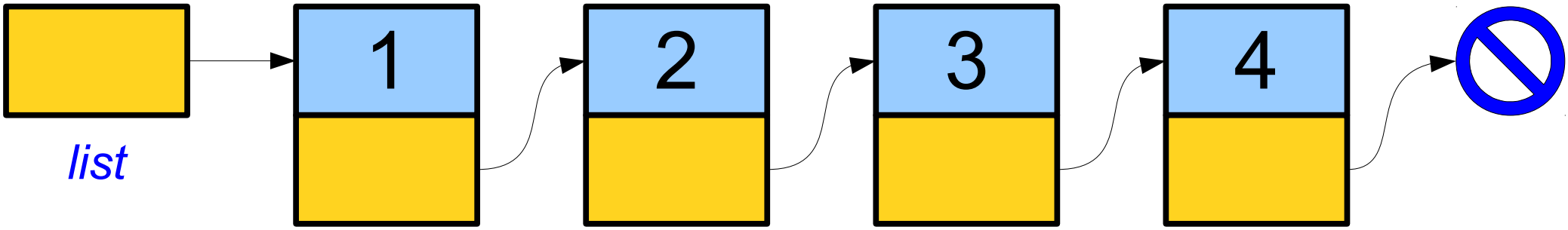
… at another linked list.



# Linked Lists, Iteratively

- You can also navigate a linked list using a traditional `for` loop:

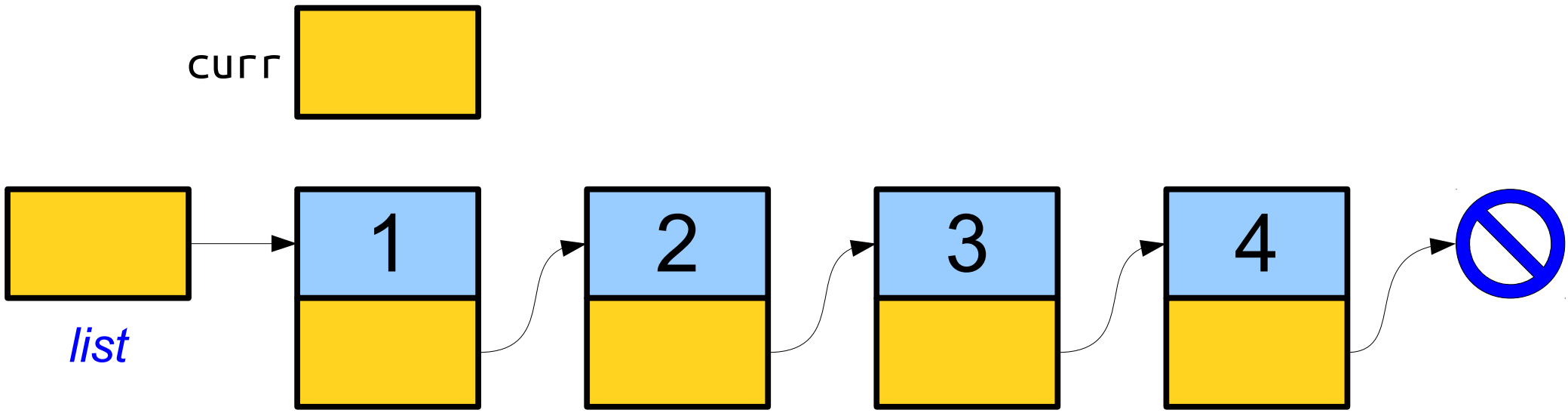
```
for (Cell* curr = list; curr != nullptr; curr = curr->next) {
    /* … do something with curr->value … */
}
```



# Linked Lists, Iteratively

- You can also navigate a linked list using a traditional `for` loop:

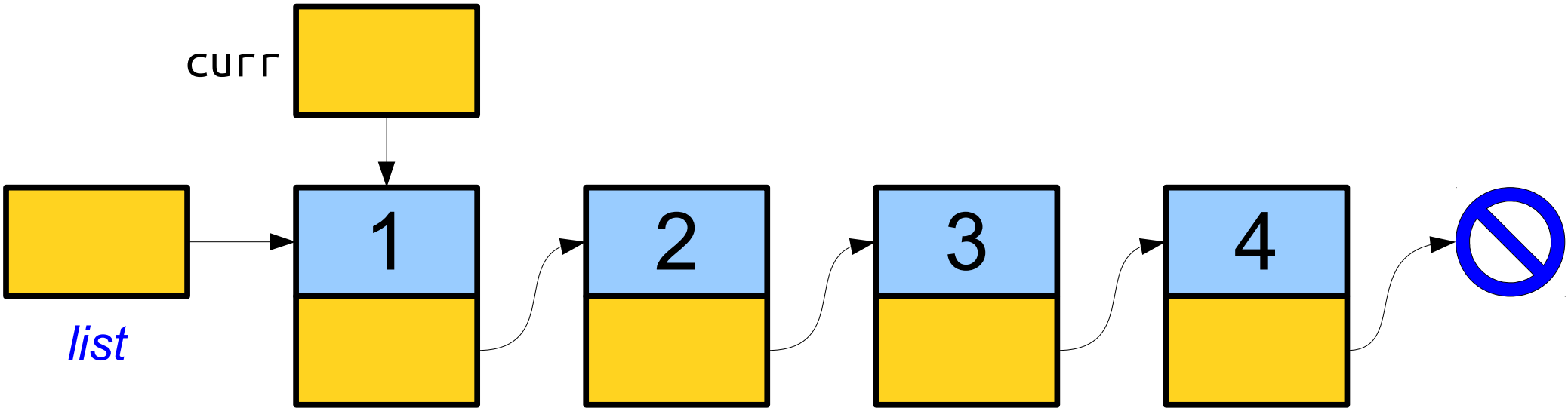
```
for (Cell* curr = list; curr != nullptr; curr = curr->next) {
    /* … do something with curr->value … */
}
```



# Linked Lists, Iteratively

- You can also navigate a linked list using a traditional `for` loop:

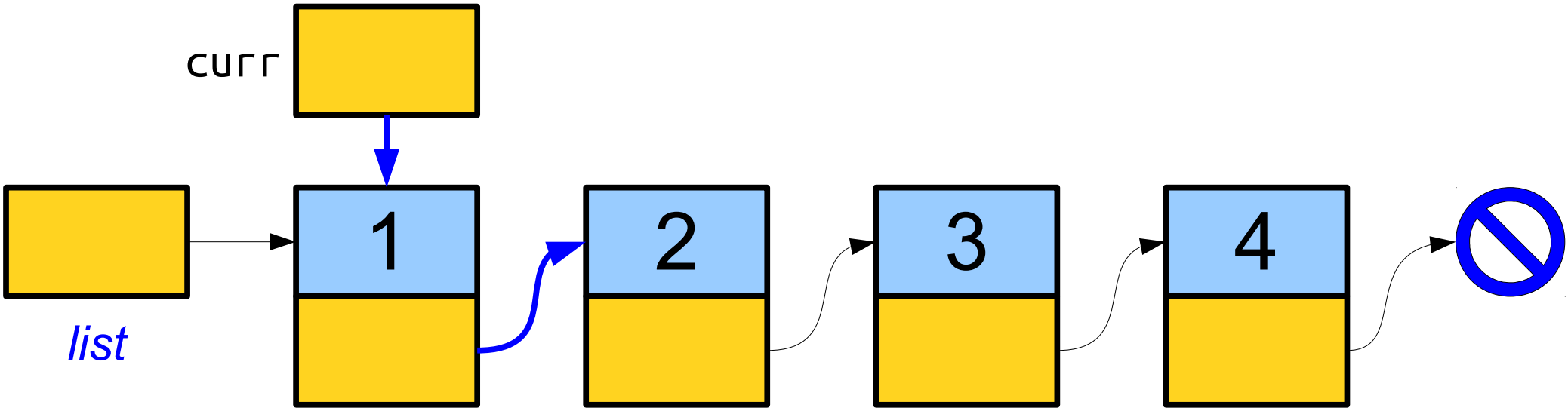
```
for (Cell* curr = list; curr != nullptr; curr = curr->next) {
    /* … do something with curr->value … */
}
```



# Linked Lists, Iteratively

- You can also navigate a linked list using a traditional `for` loop:

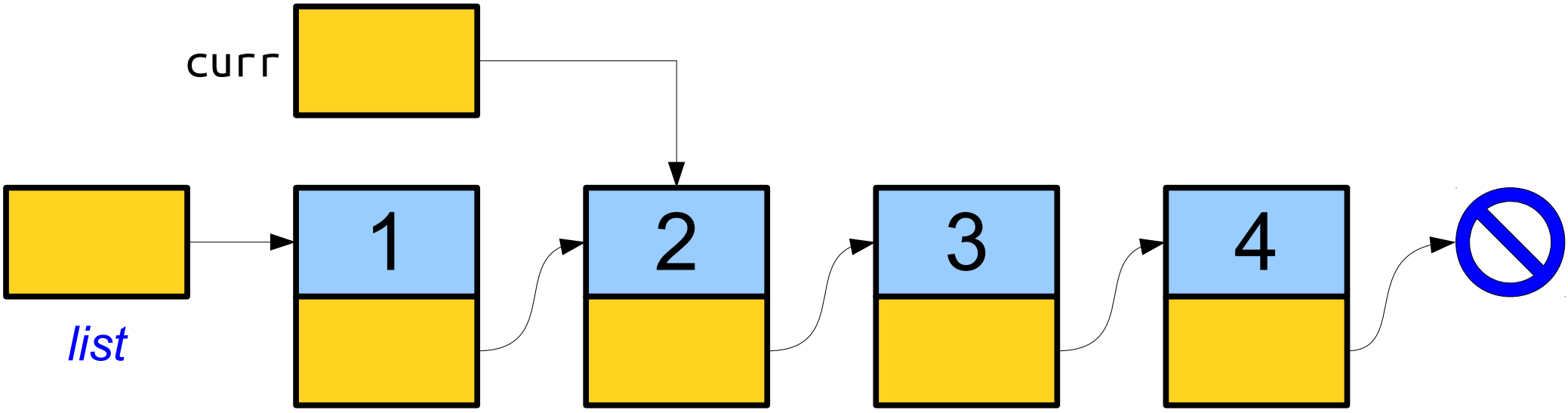
```
for (Cell* curr = list; curr != nullptr; curr = curr->next) {
    /* … do something with curr->value … */
}
```



# Linked Lists, Iteratively

- You can also navigate a linked list using a traditional `for` loop:

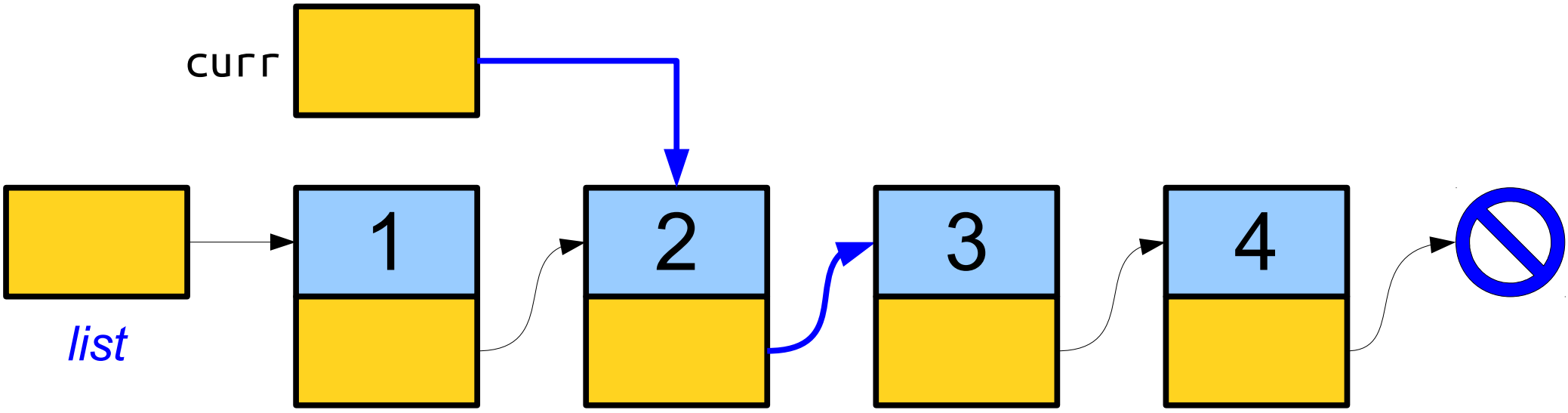
```
for (Cell* curr = list; curr != nullptr; curr = curr->next) {
    /* … do something with curr->value … */
}
```



# Linked Lists, Iteratively

- You can also navigate a linked list using a traditional `for` loop:

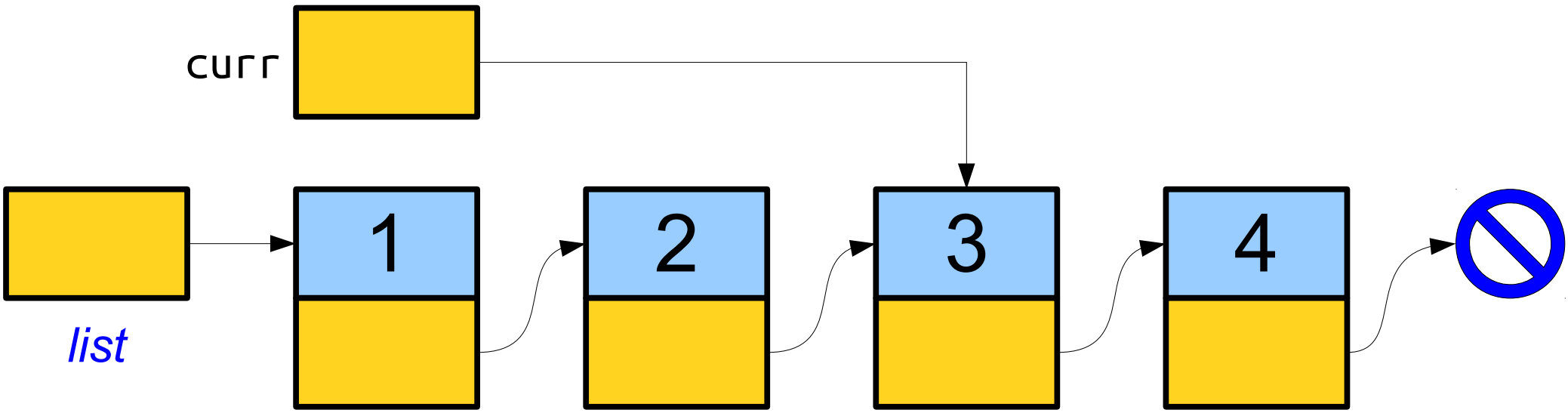
```
for (Cell* curr = list; curr != nullptr; curr = curr->next) {
    /* … do something with curr->value … */
}
```



# Linked Lists, Iteratively

- You can also navigate a linked list using a traditional `for` loop:

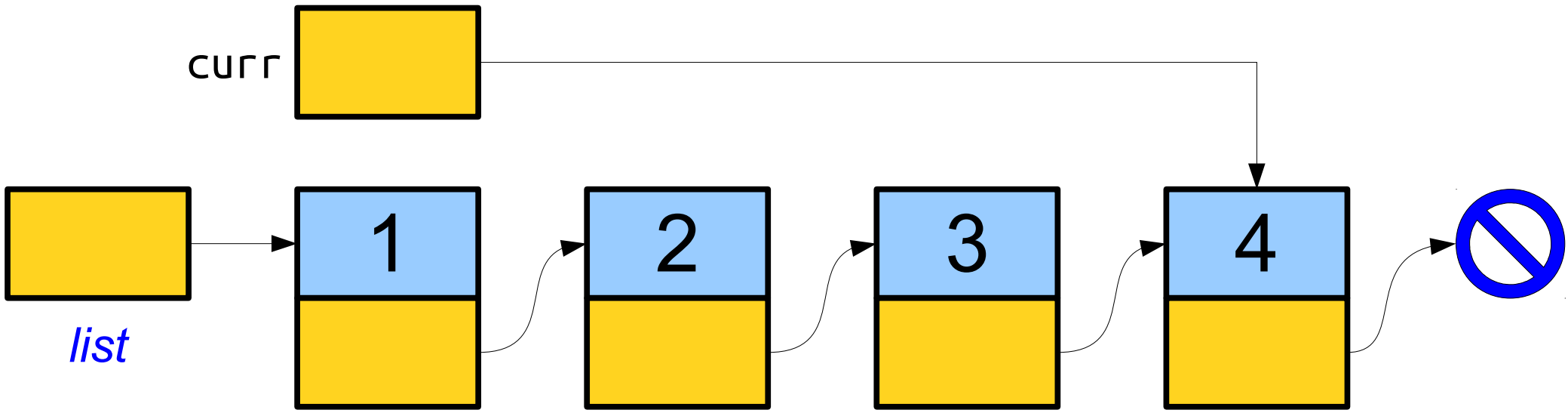
```
for (Cell* curr = list; curr != nullptr; curr = curr->next) {
    /* … do something with curr->value … */
}
```



# Linked Lists, Iteratively

- You can also navigate a linked list using a traditional `for` loop:

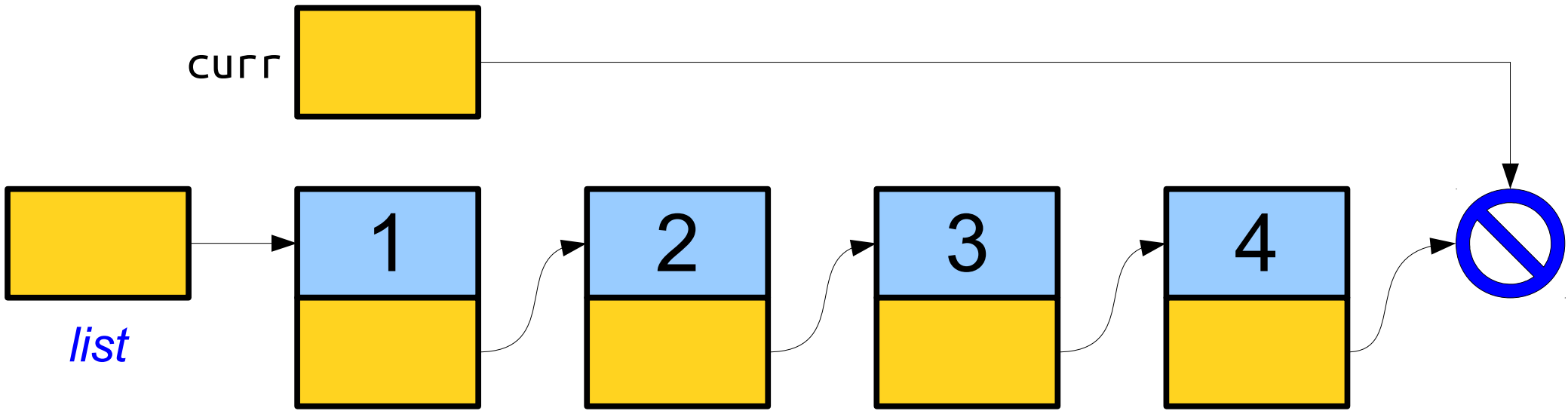
```
for (Cell* curr = list; curr != nullptr; curr = curr->next) {
    /* … do something with curr->value … */
}
```



# Linked Lists, Iteratively

- You can also navigate a linked list using a traditional `for` loop:

```
for (Cell* curr = list; curr != nullptr; curr = curr->next) {
    /* … do something with curr->value … */
}
```

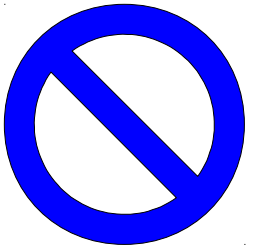


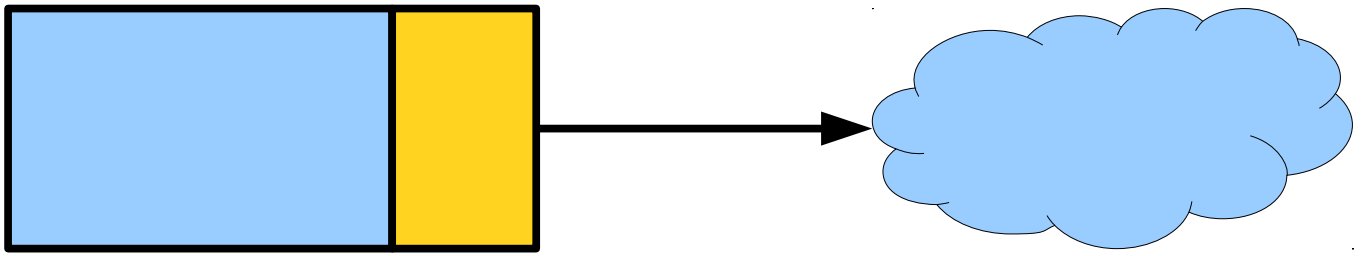
# Building a Linked List

*(without hardcoding it)*

# A Linked List is Either…

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





a single linked list cell that points…

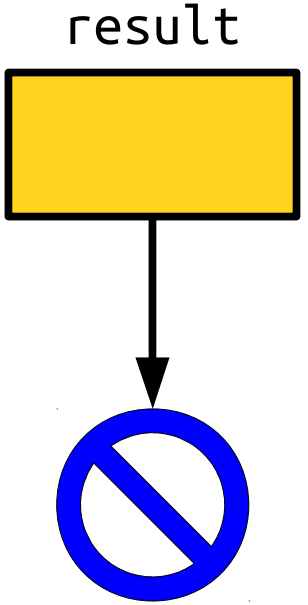
… at another linked list.

# Once More, With Iteration!

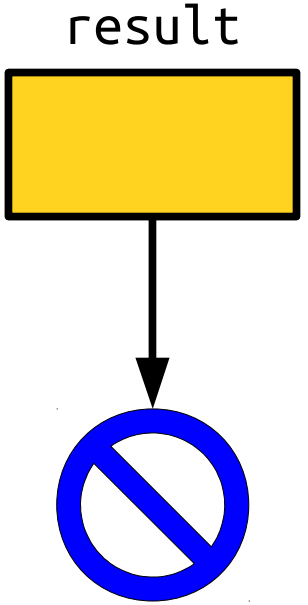
```
Cell* result = nullptr;
while (true) {









}
return result;
```

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




}
return result;
```

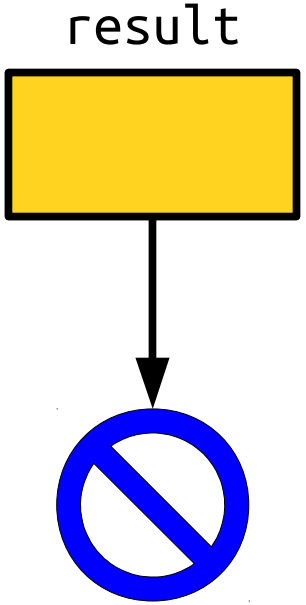
```
Cell* result = nullptr;
while (true) {




}
return result;
```



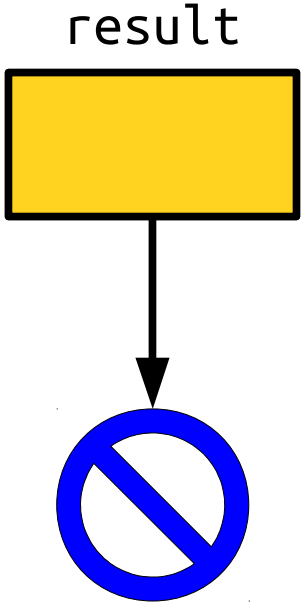
```
Cell* result = nullptr;
while (true) {




}
return result;
```



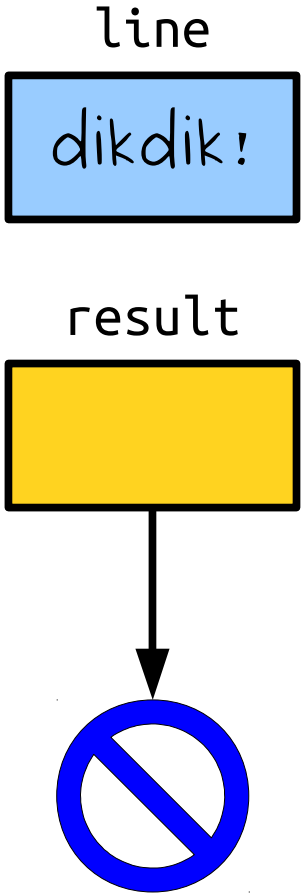
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;




}
return result;
```



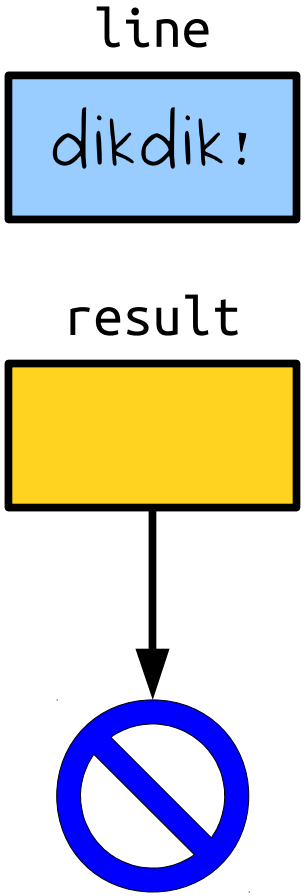
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;




}
return result;
```



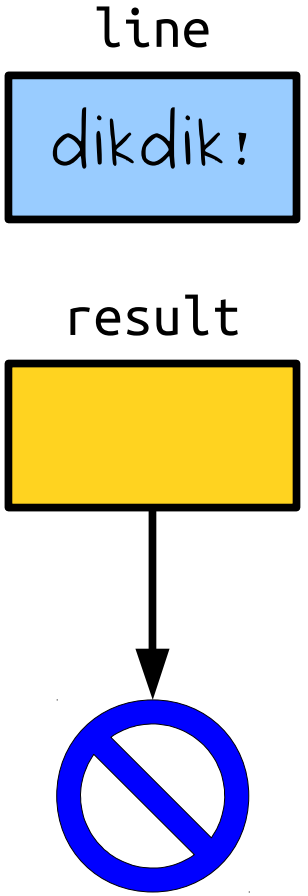
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;




}
return result;
```



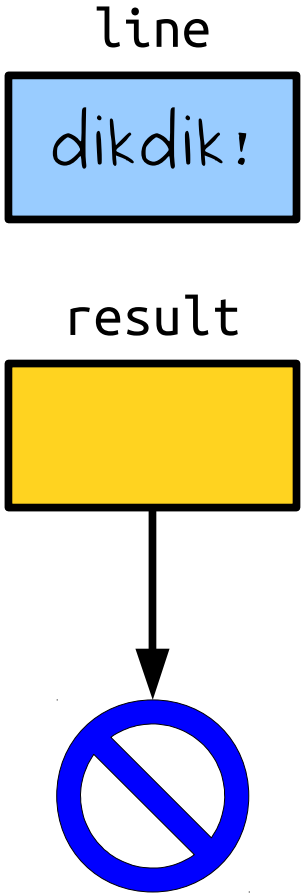
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;


}
return result;
```



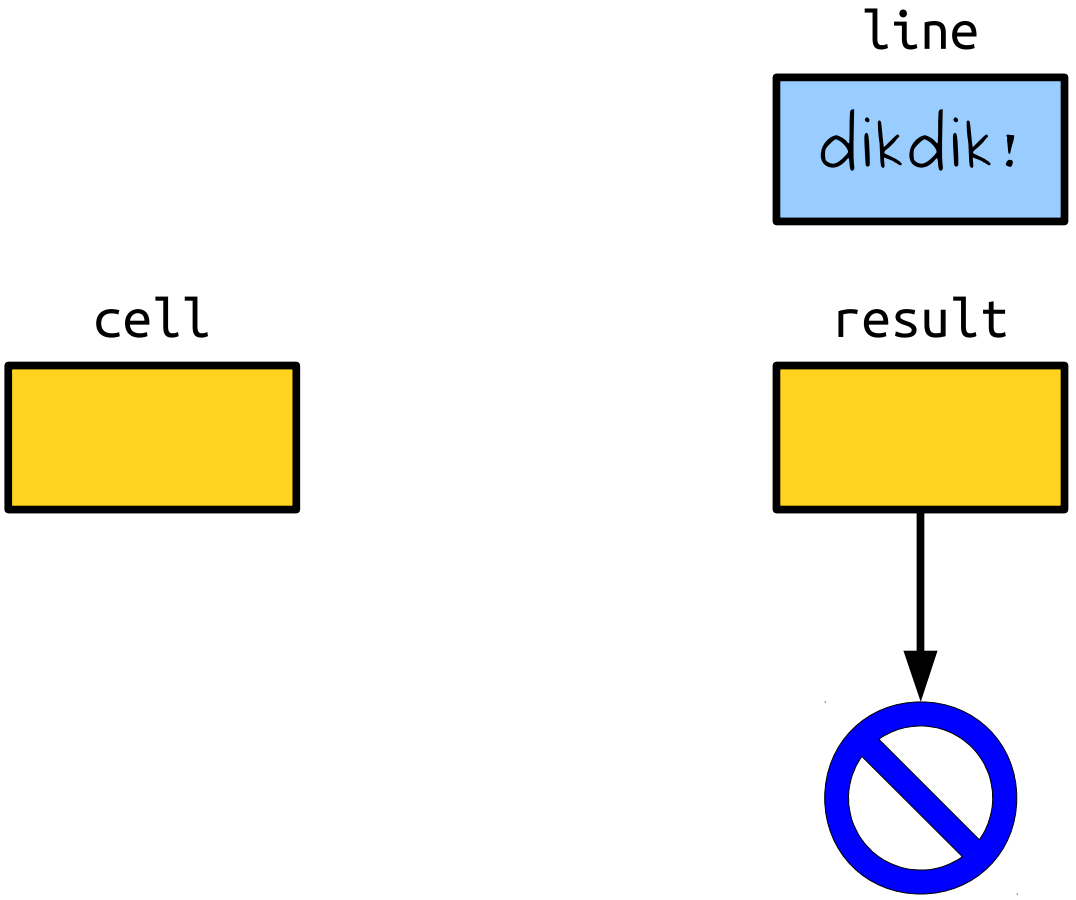
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;



}
return result;
```



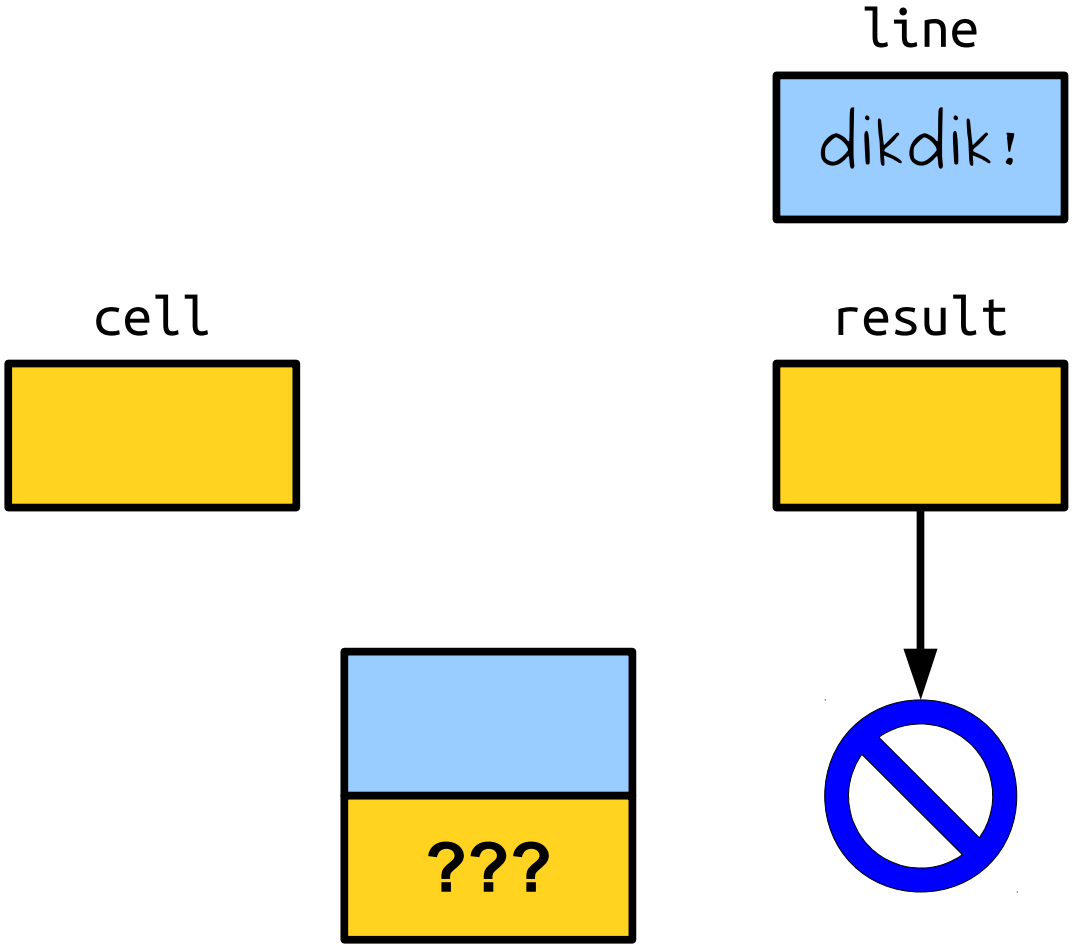
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;



}
return result;
```



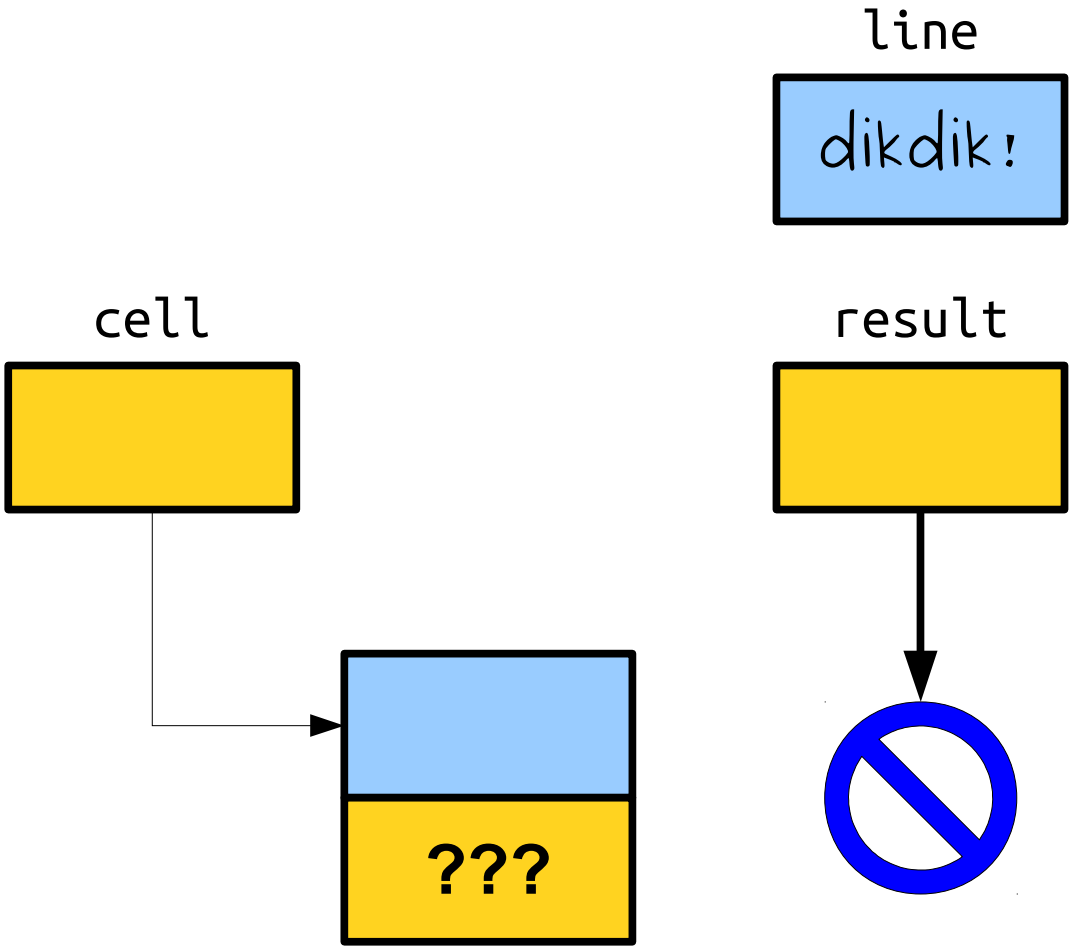
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;



}
return result;
```



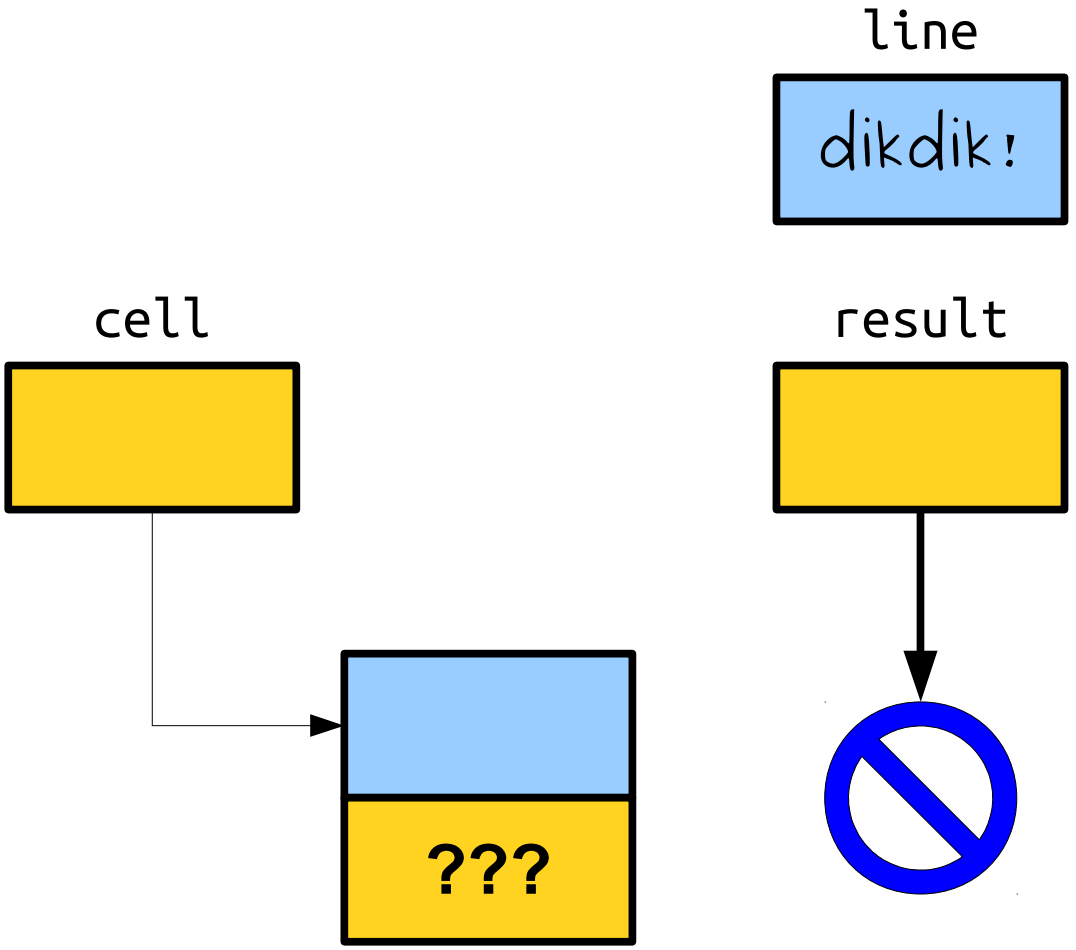
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;



}
return result;
```



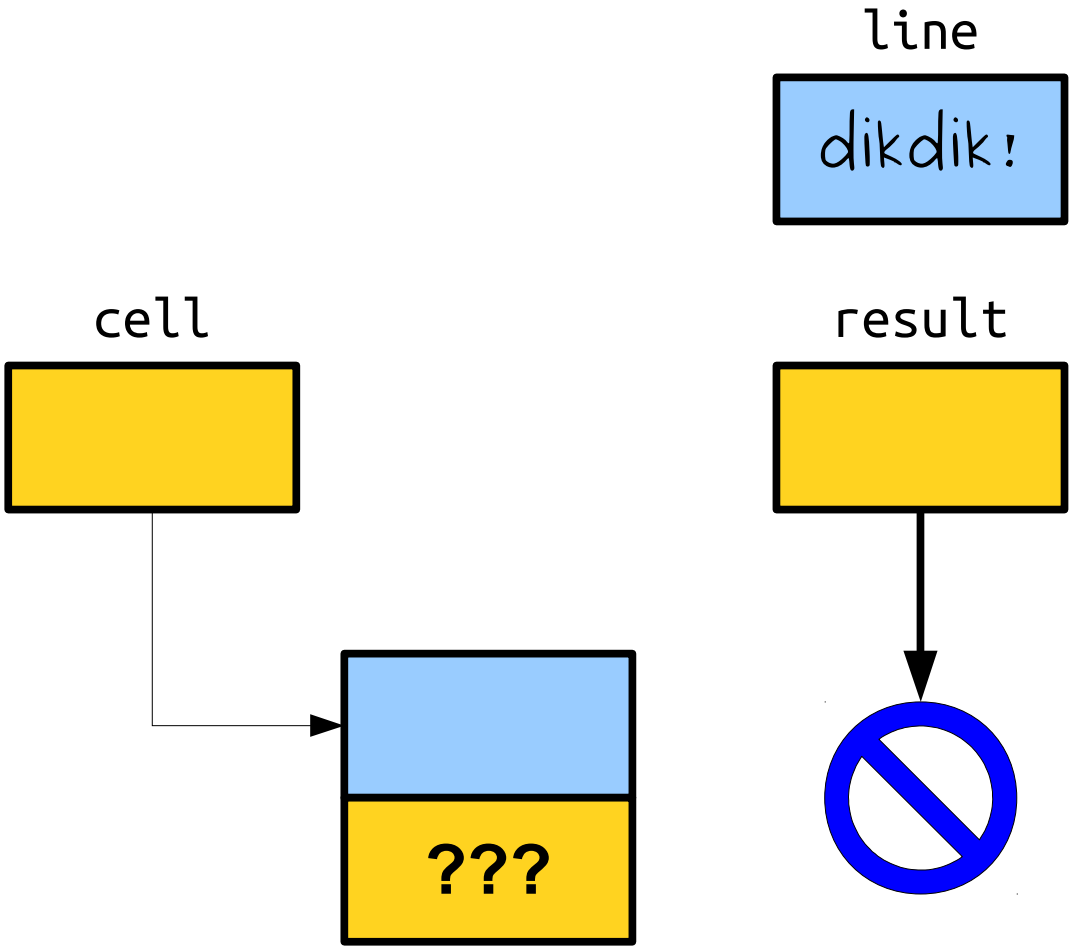
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    Cell* cell = new Cell;



}
return result;
```



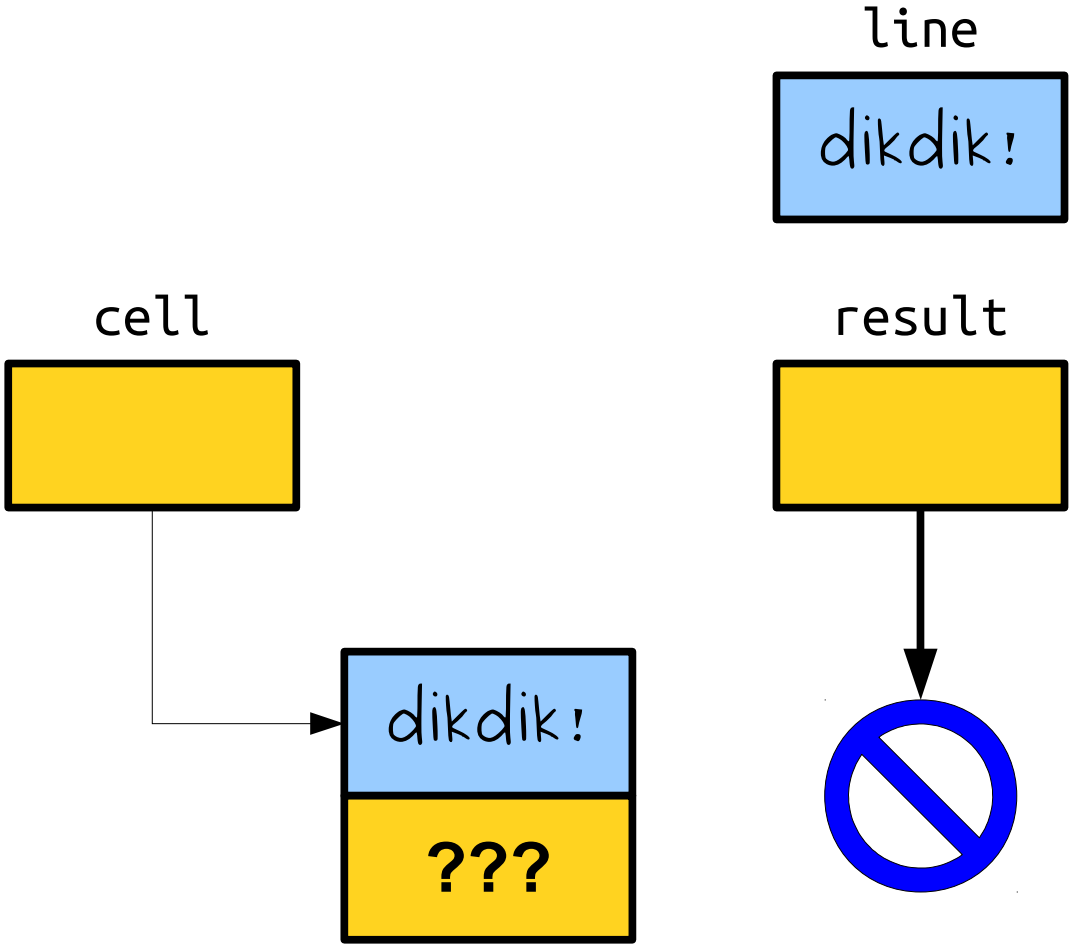
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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



}
return result;
```



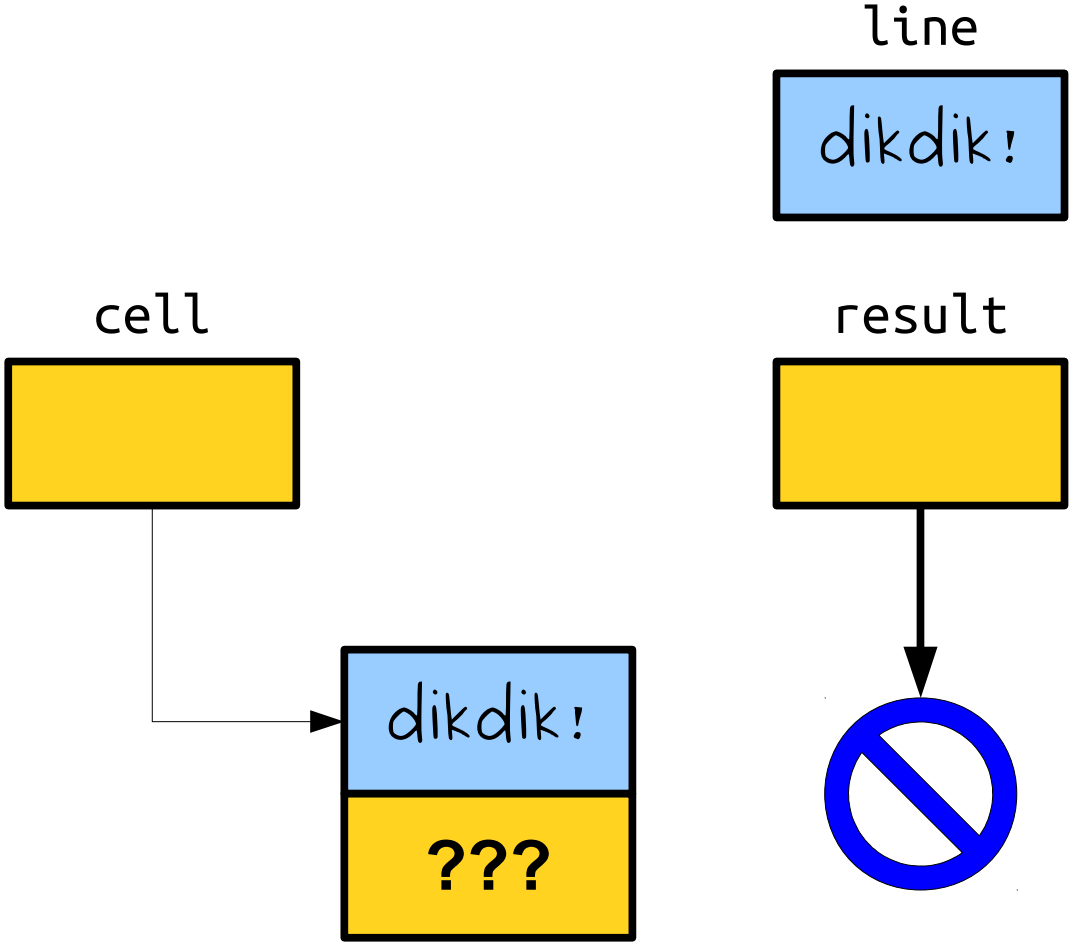
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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



}
return result;
```



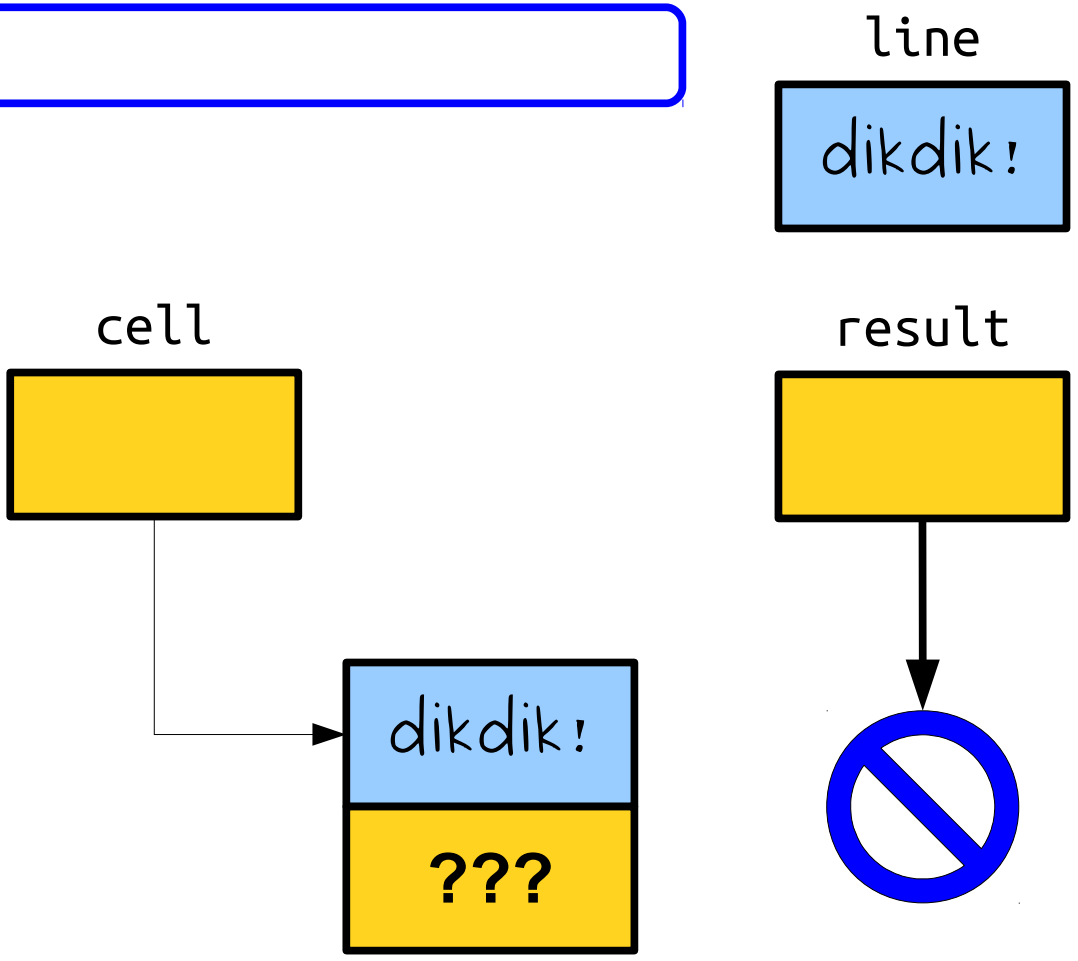
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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



}
return result;
```



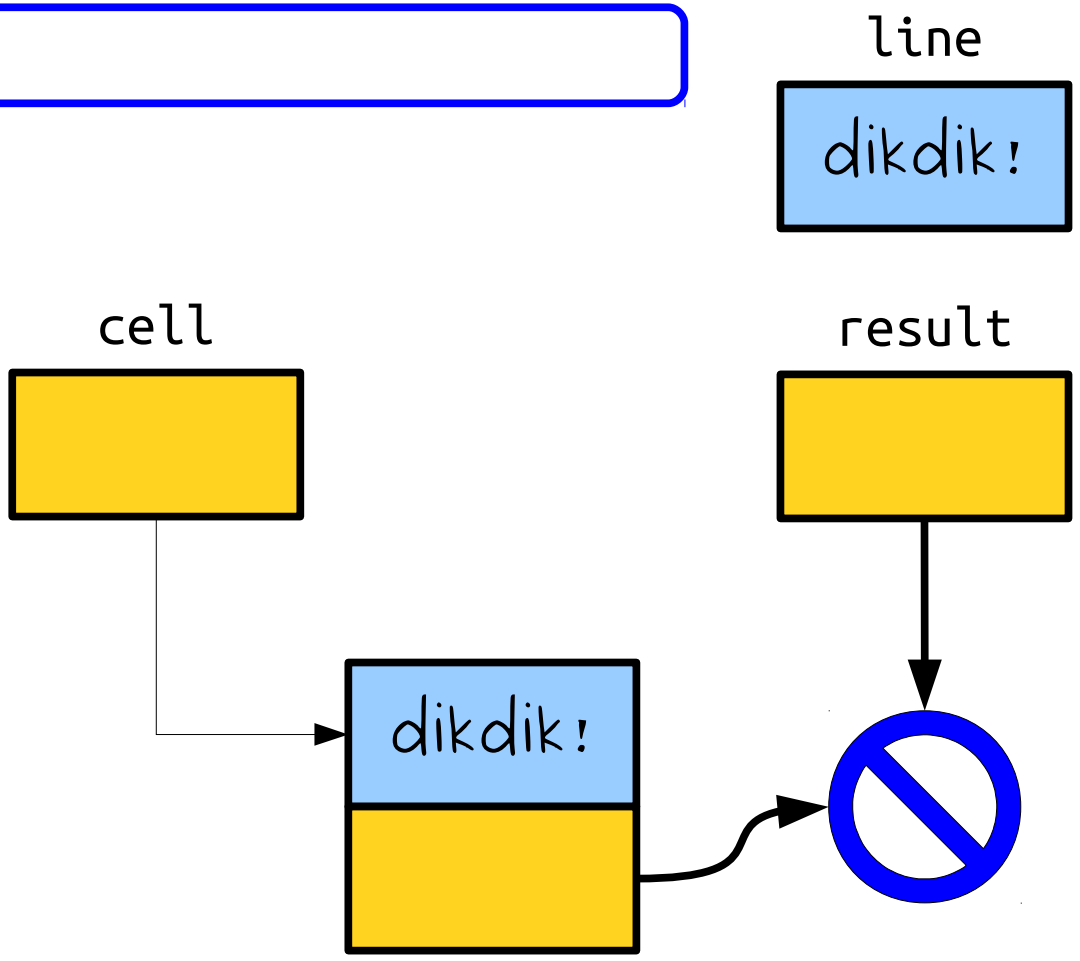
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

    cell->next = result;

}
return result;
```



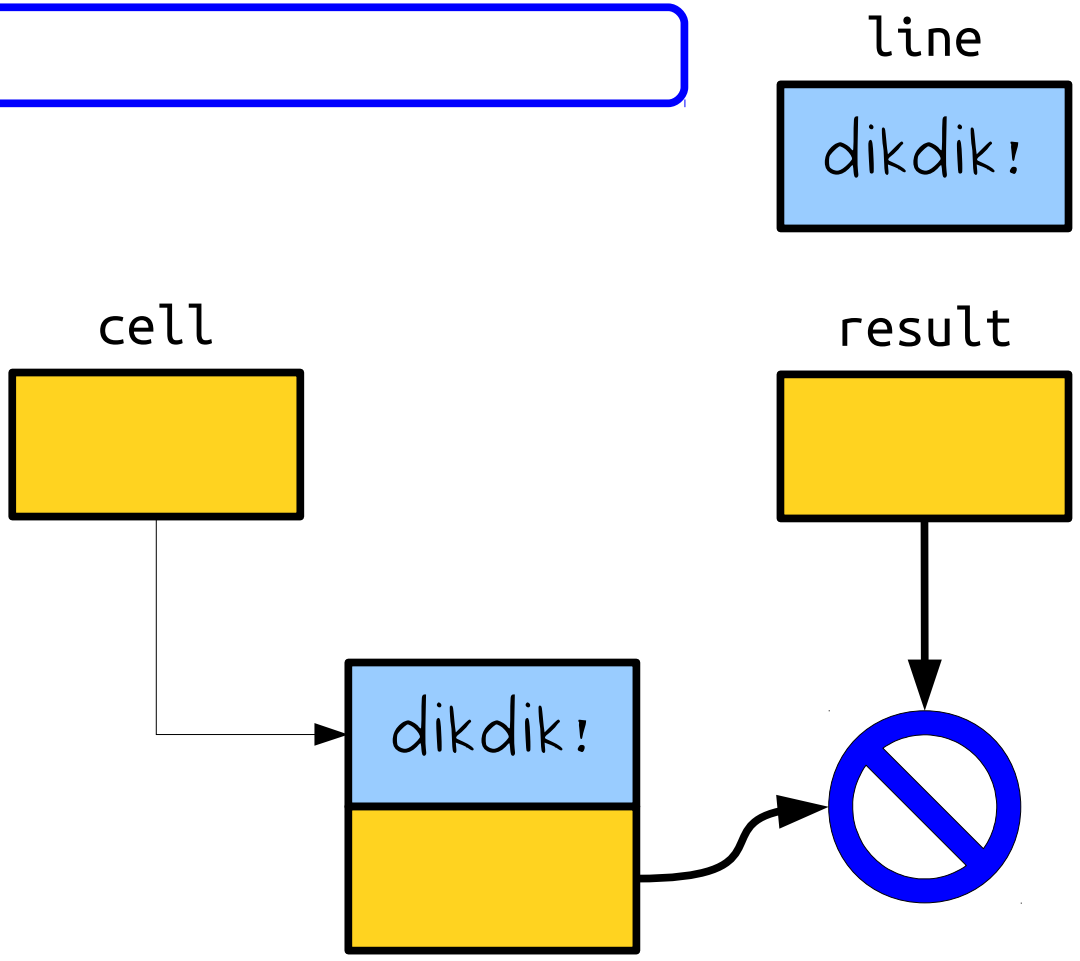
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

    cell->next = result;

}
return result;
```



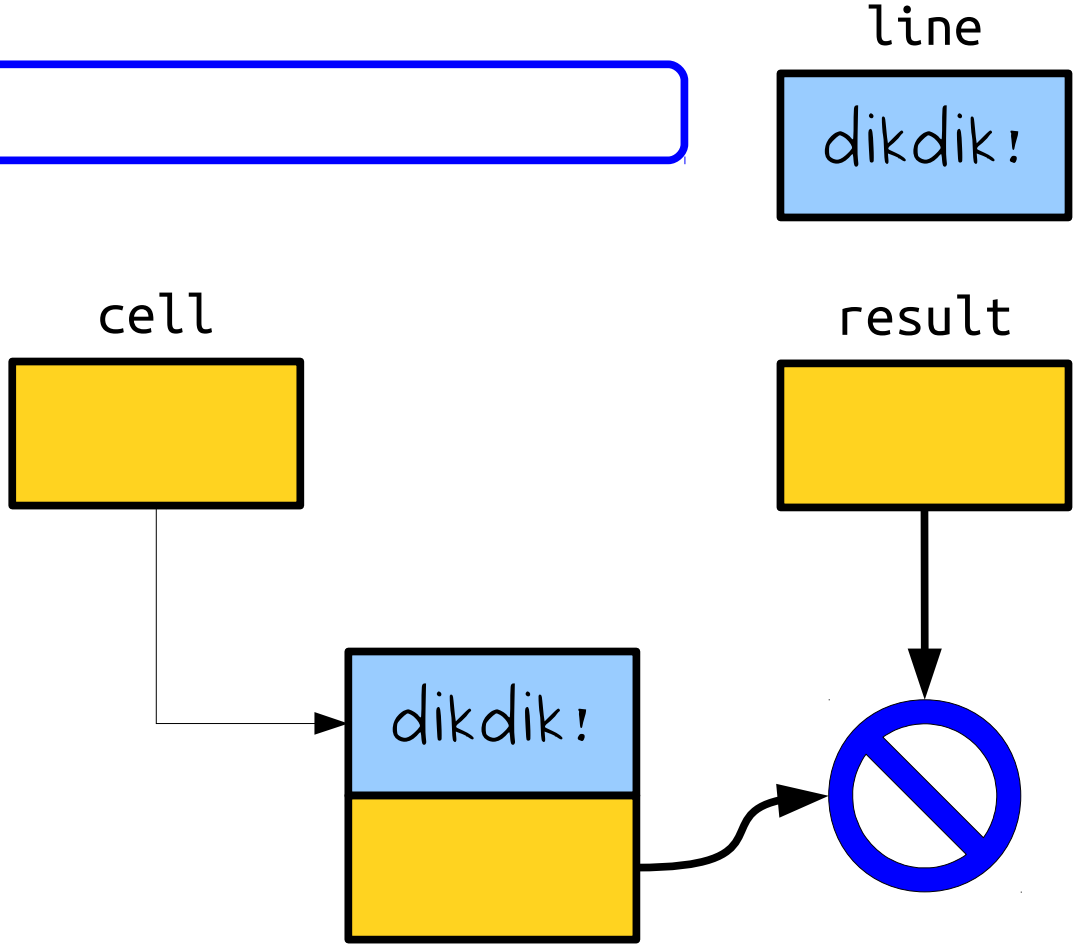
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

    cell->next = result;

}
return result;
```



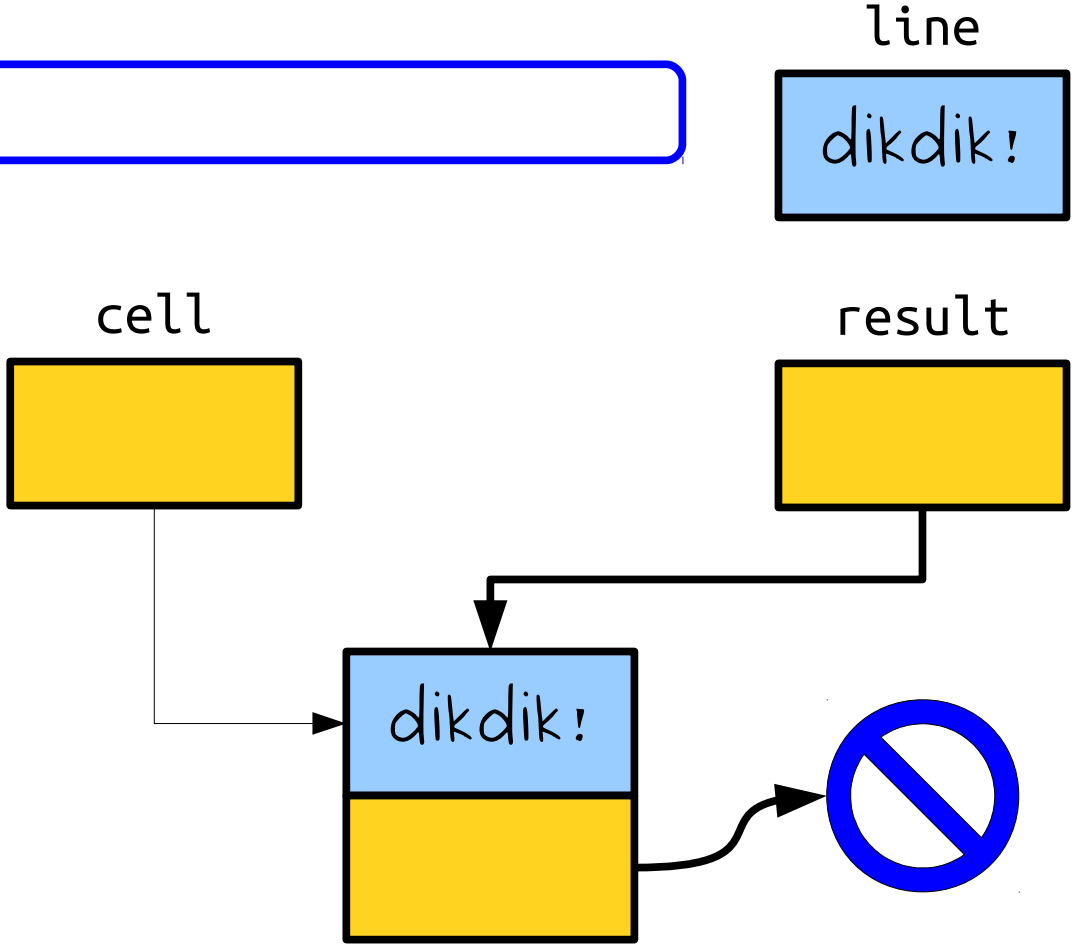
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

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



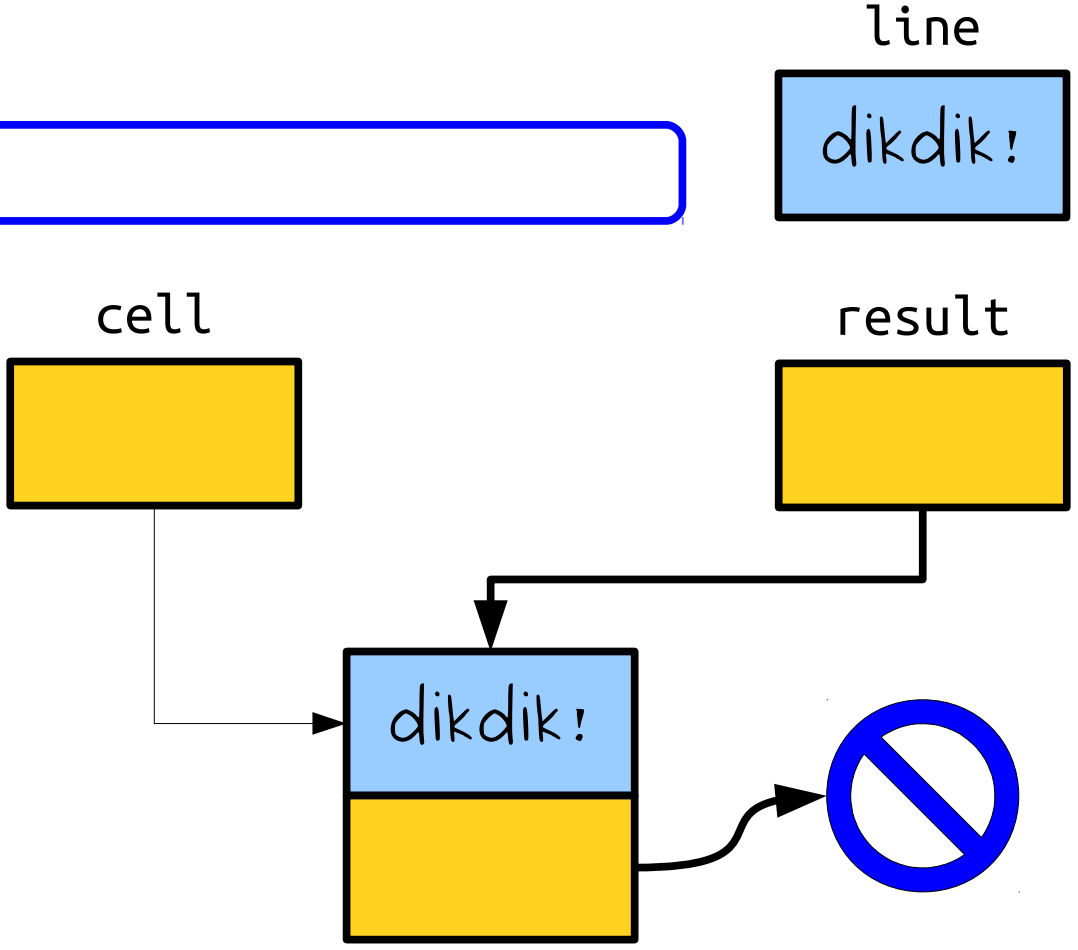
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

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



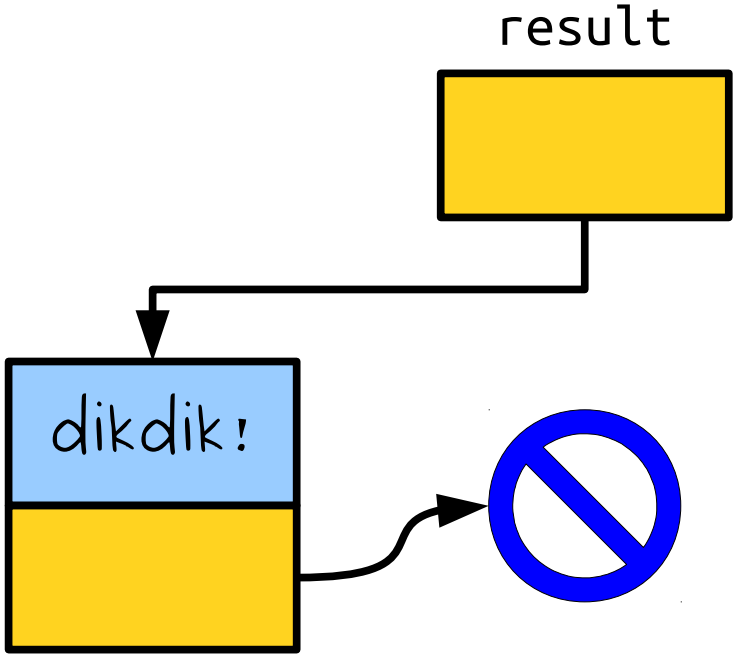
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

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



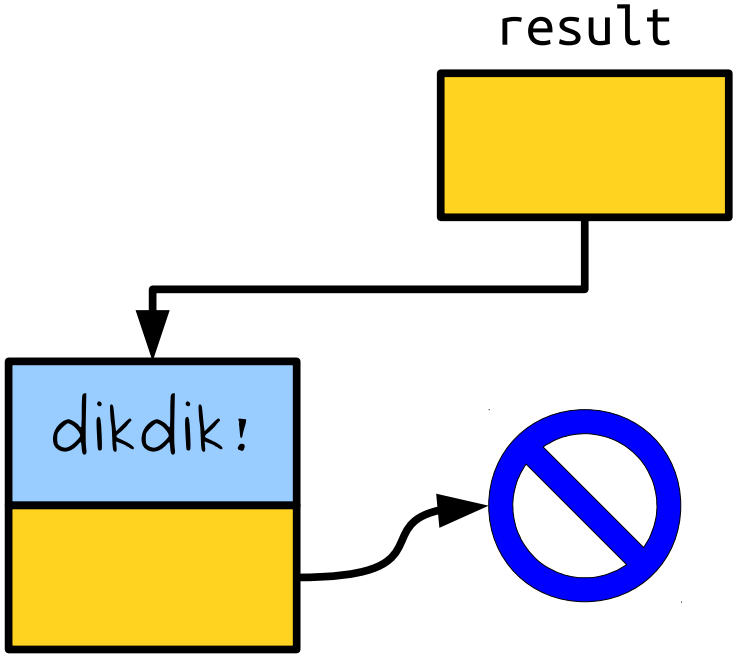
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

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



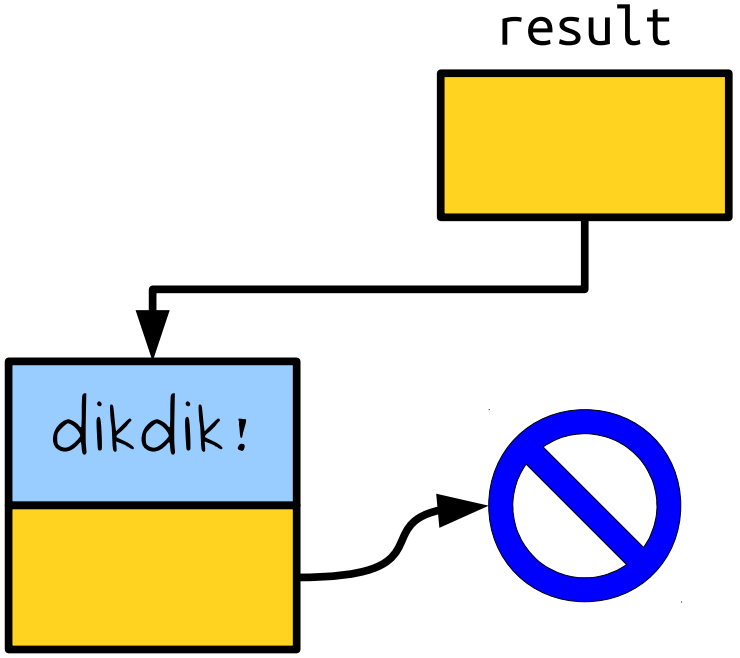
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

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



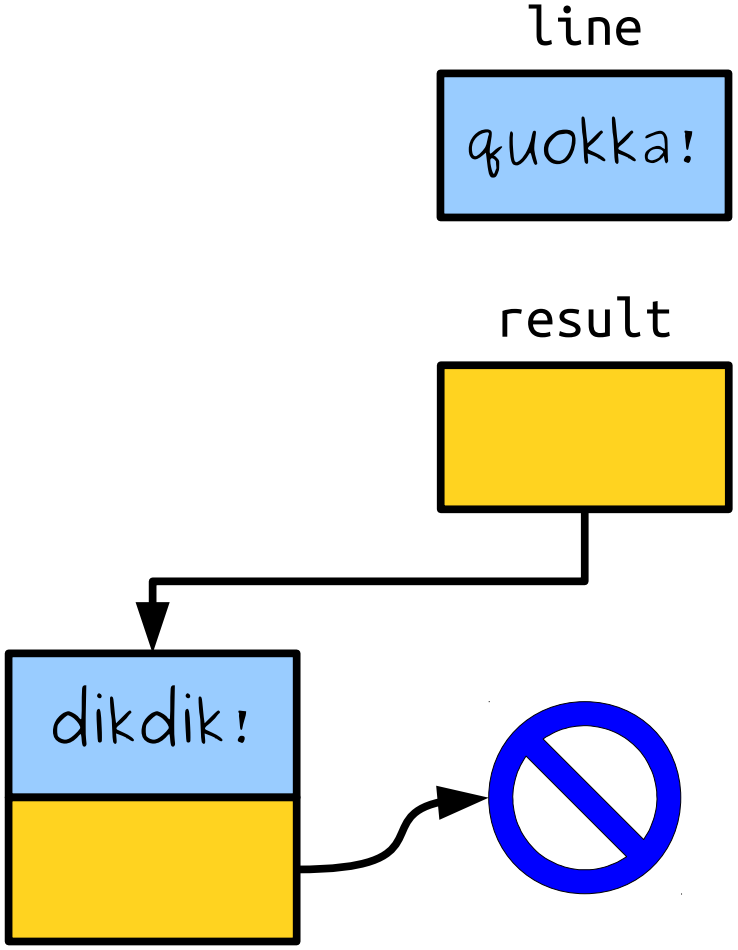
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

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



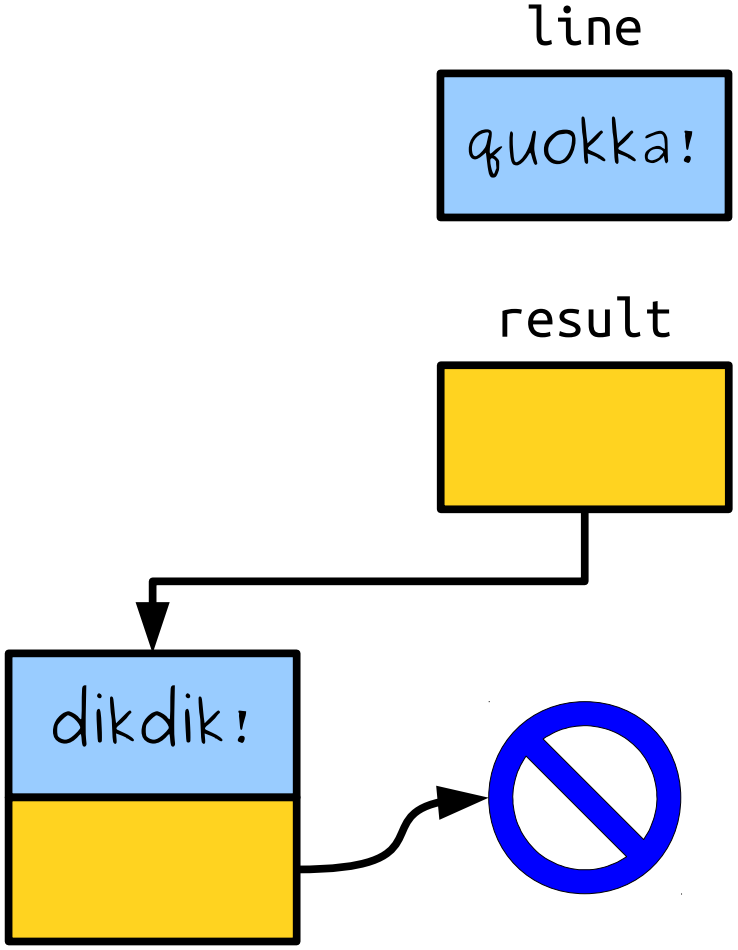
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

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



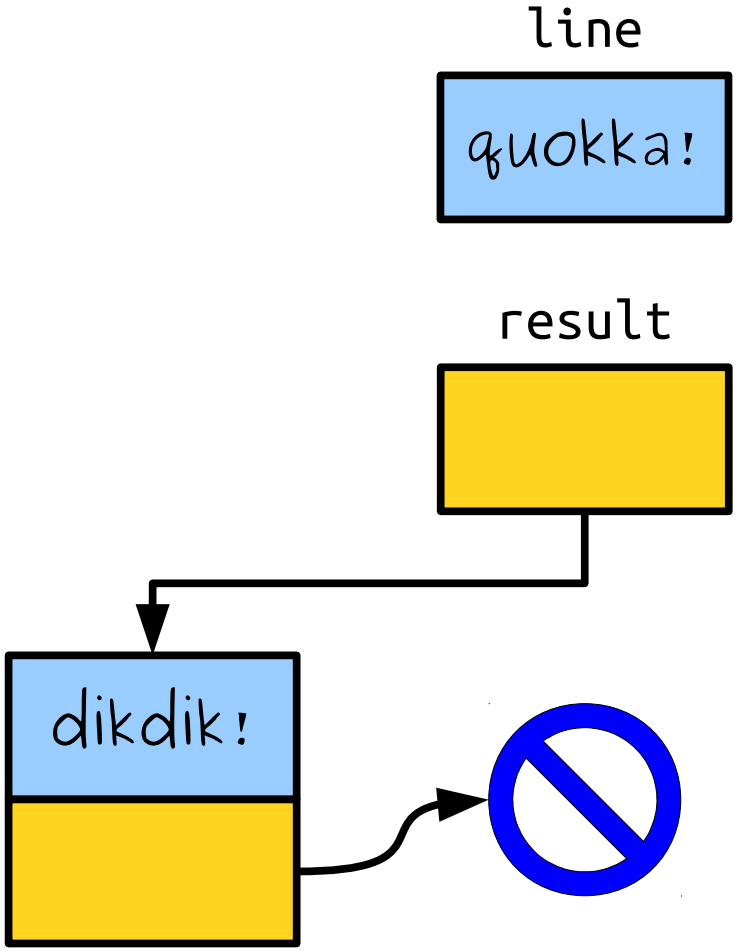
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

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



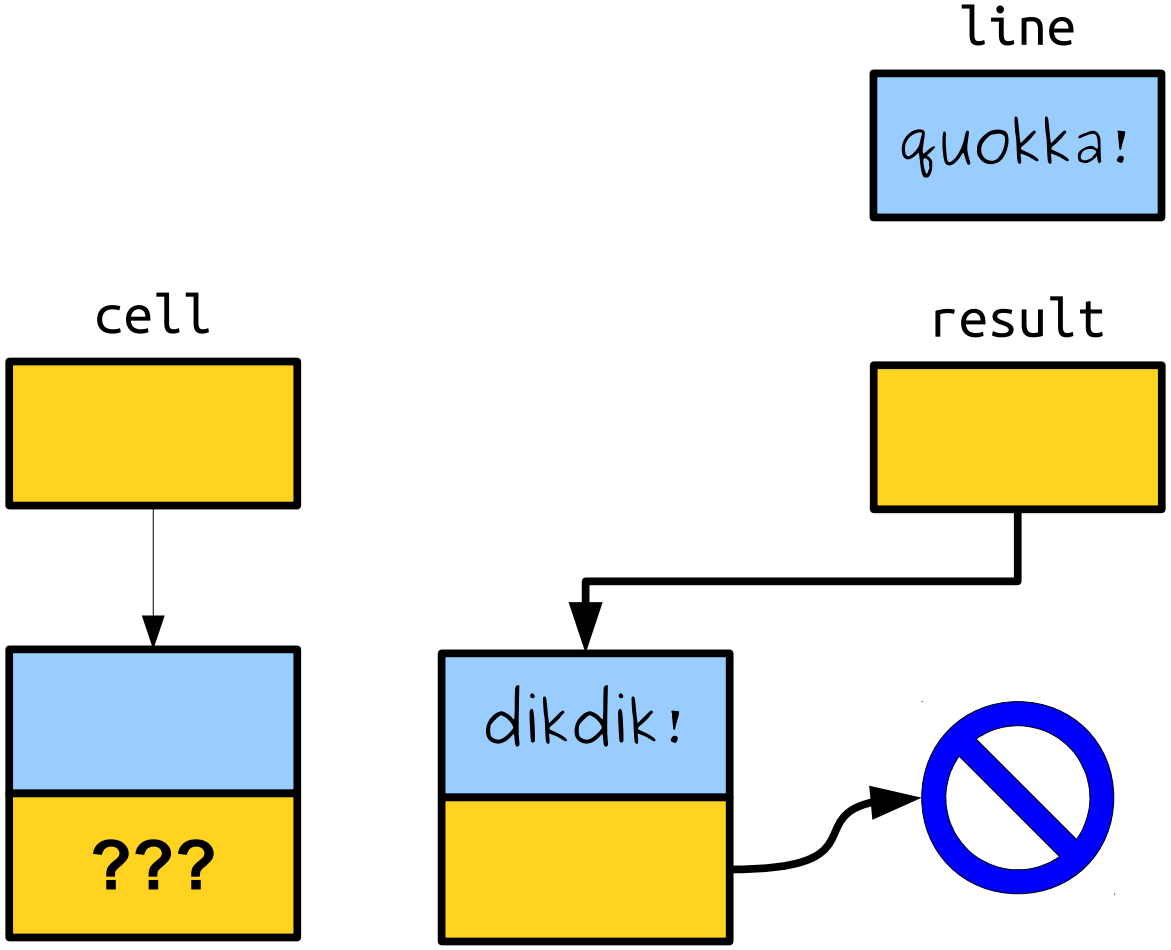
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

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



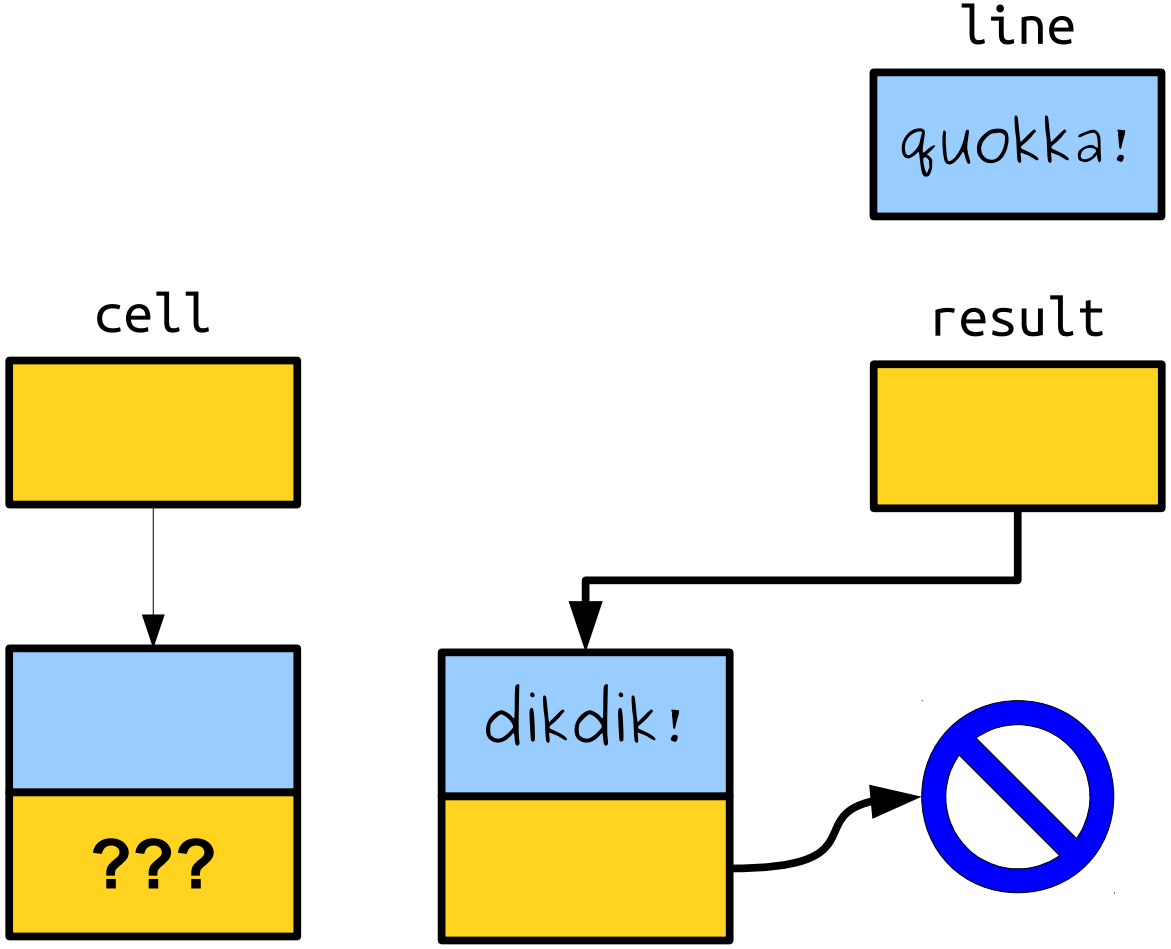
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

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



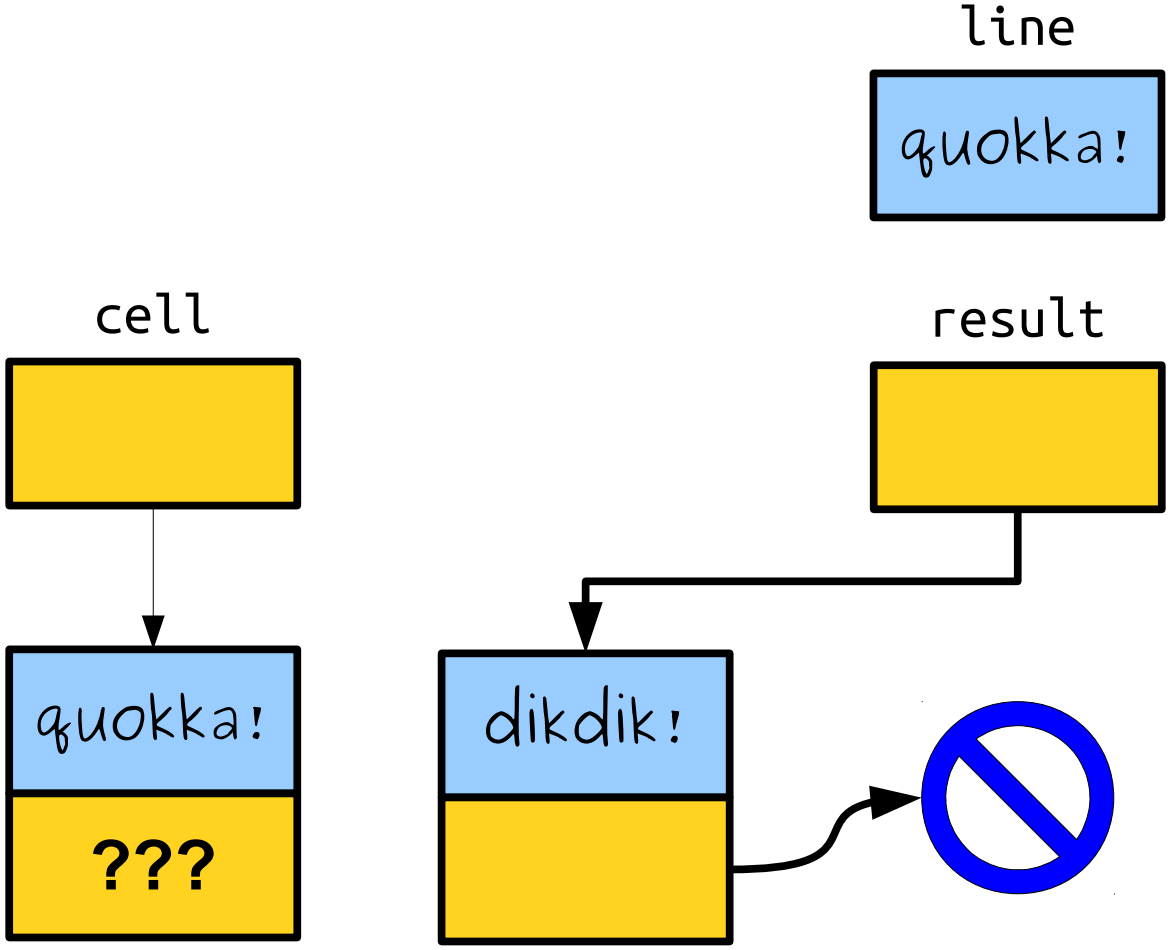
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

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



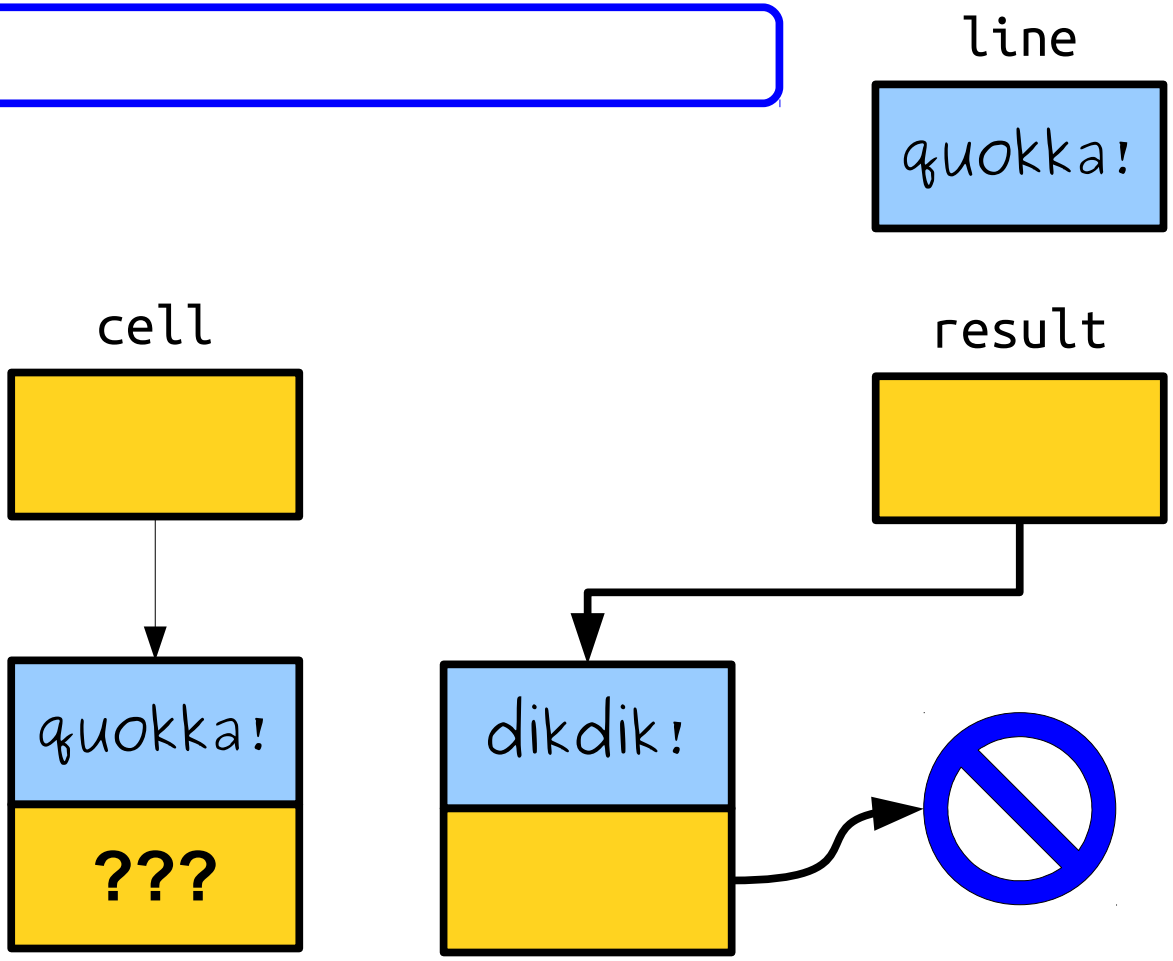
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

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



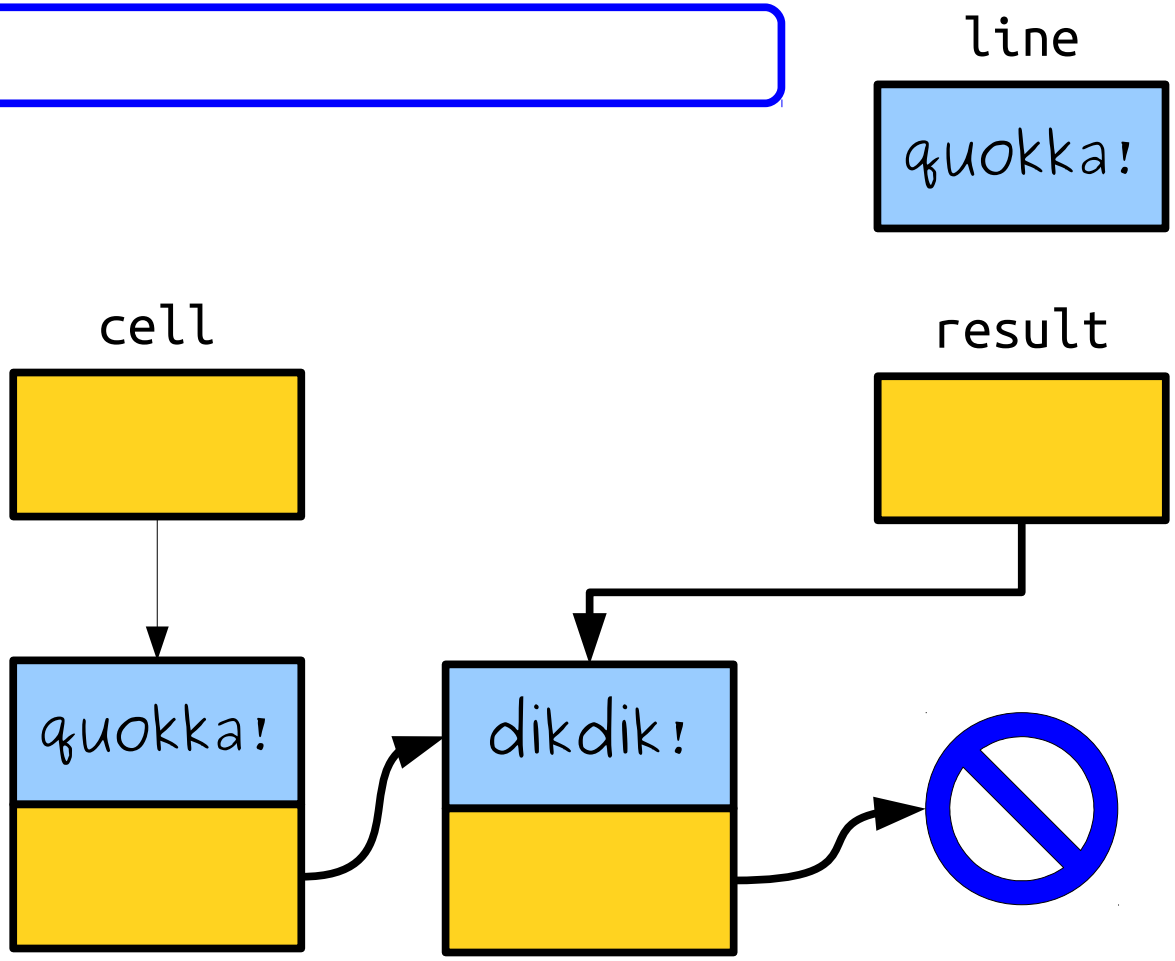
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

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



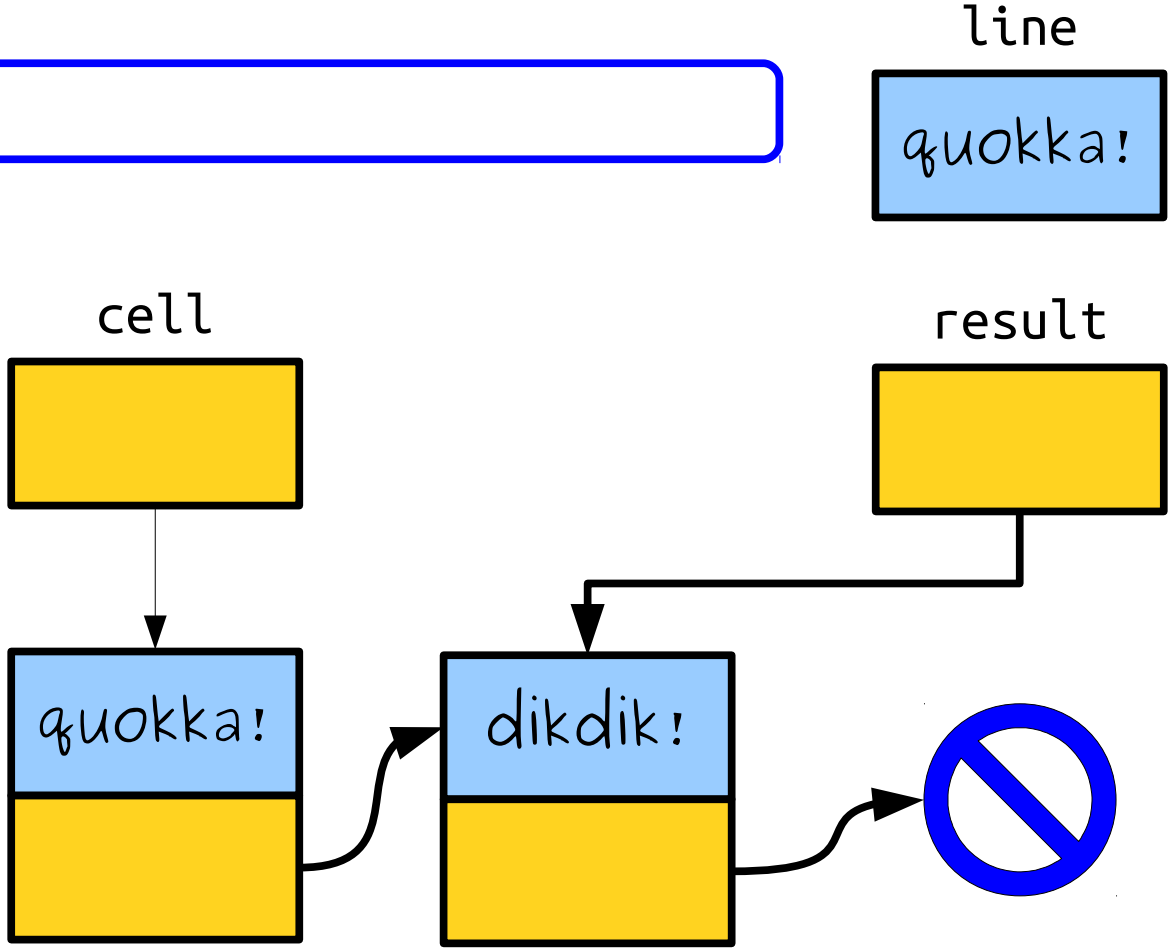
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

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



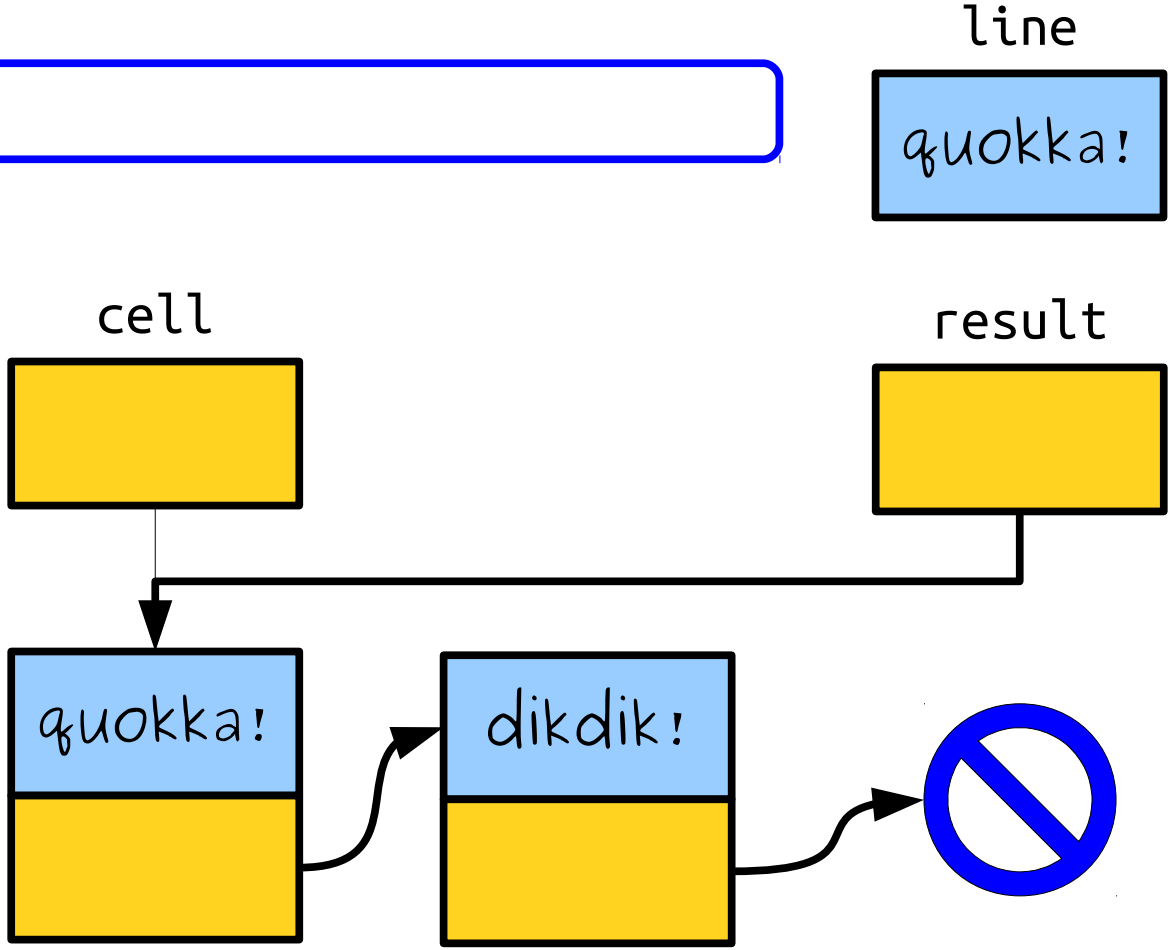
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

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



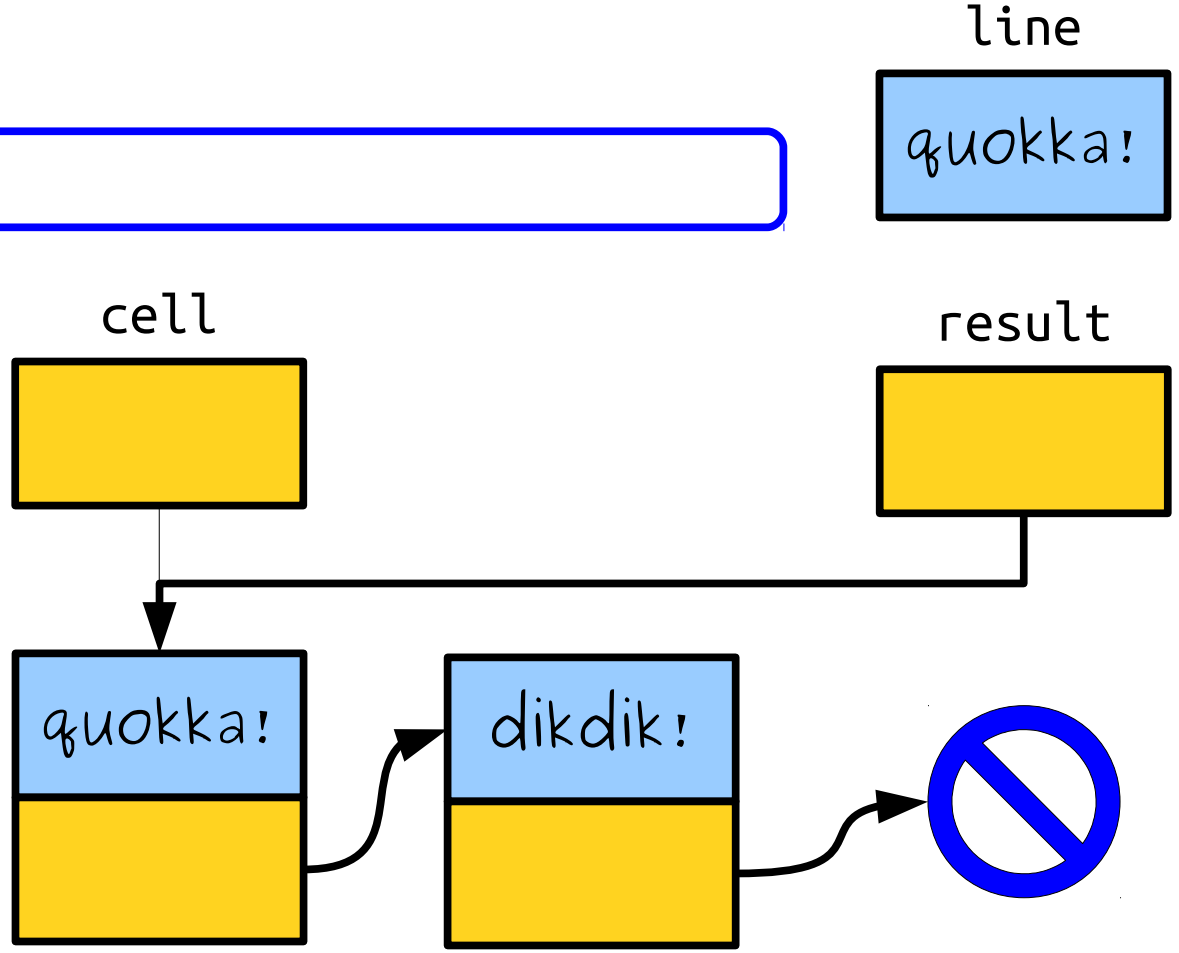
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

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



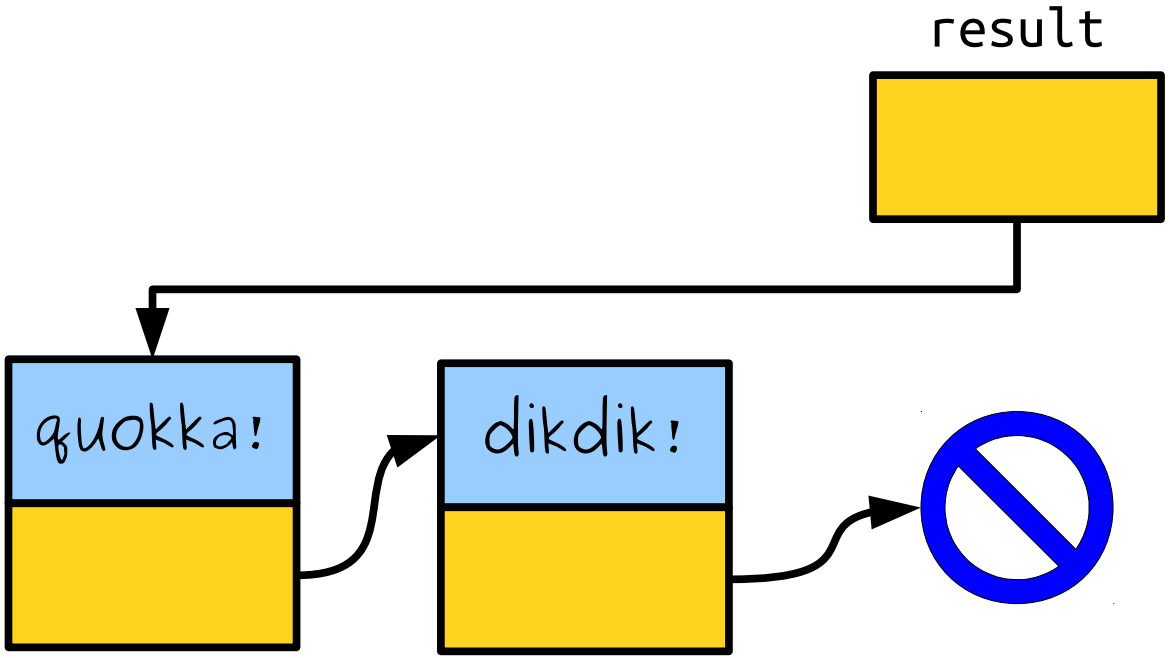
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

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



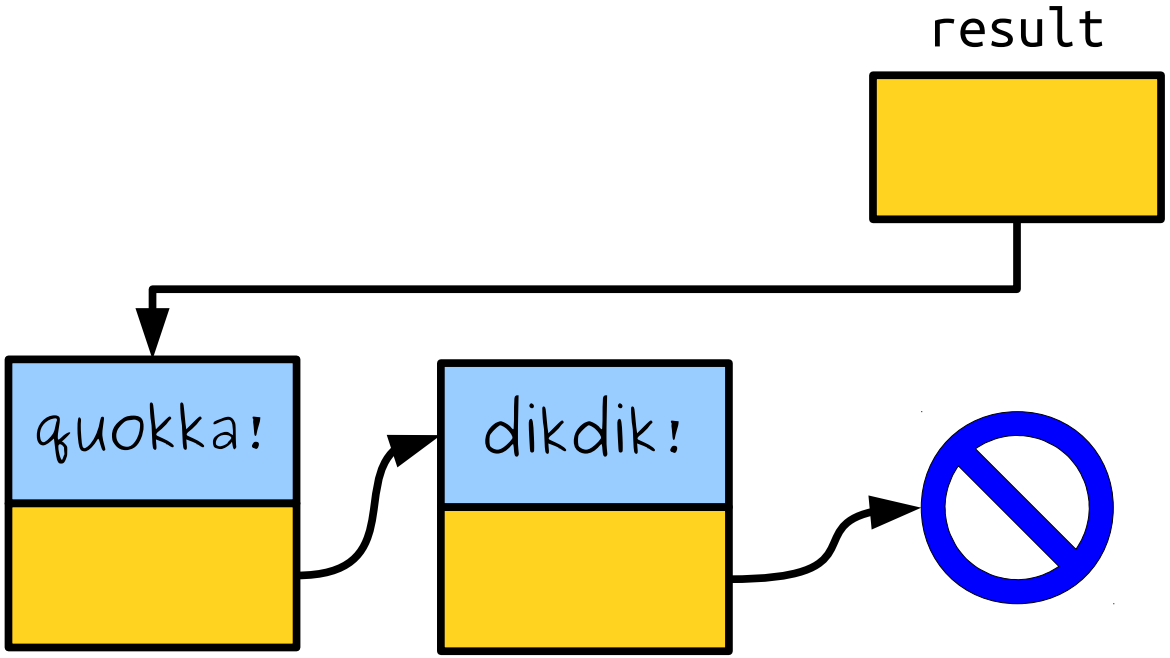
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

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



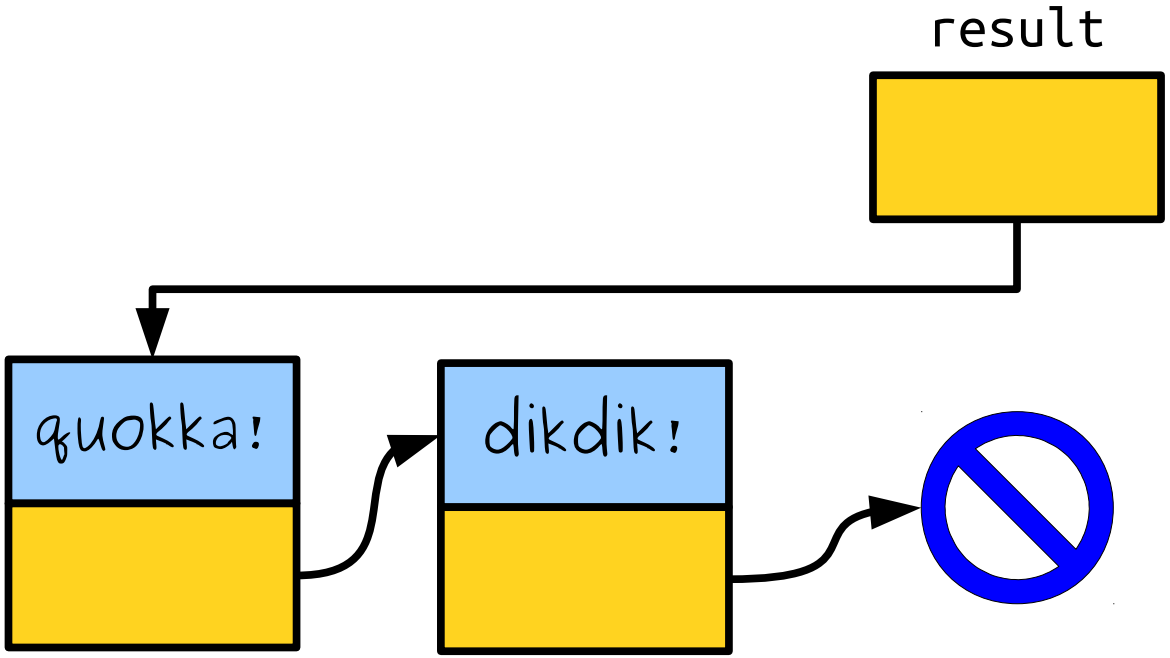
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

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



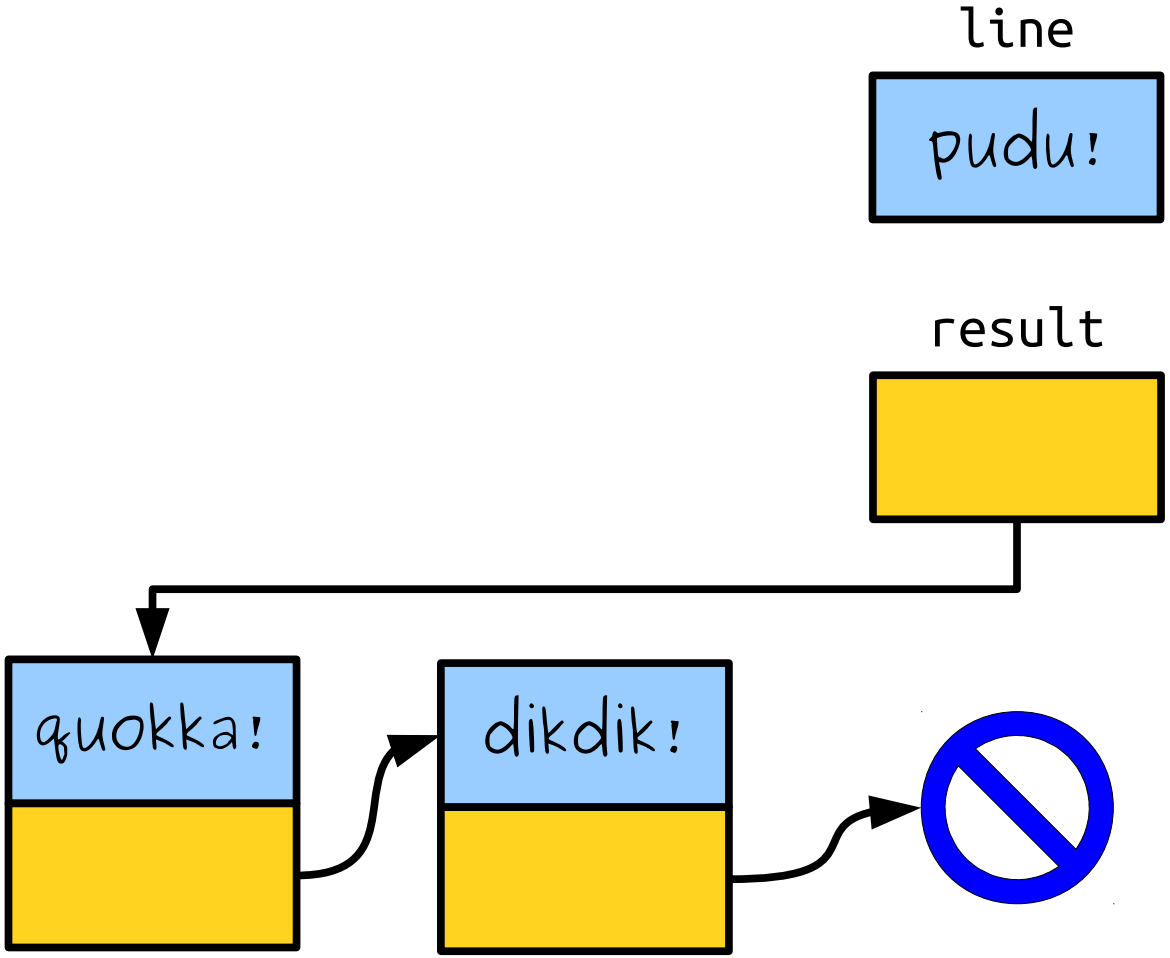
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

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



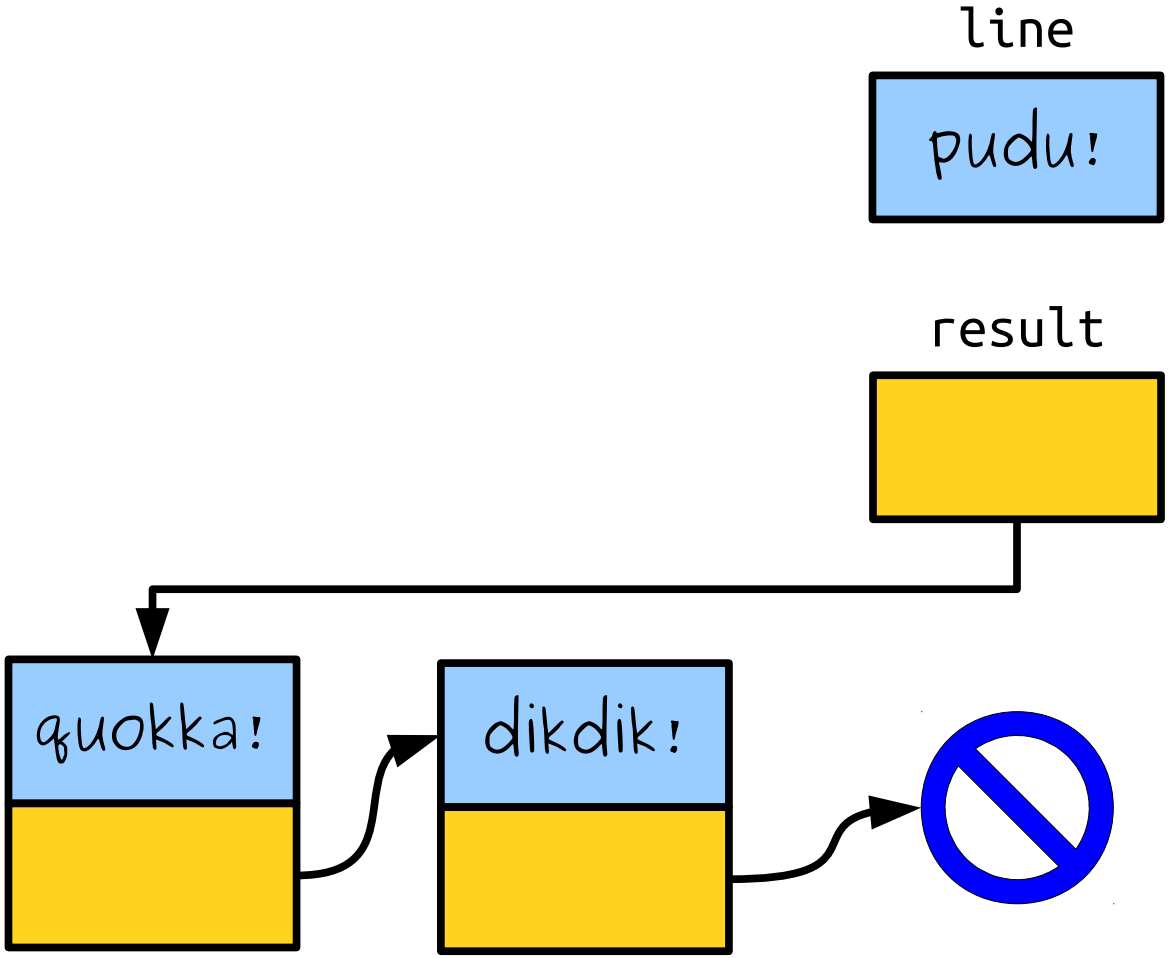
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

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



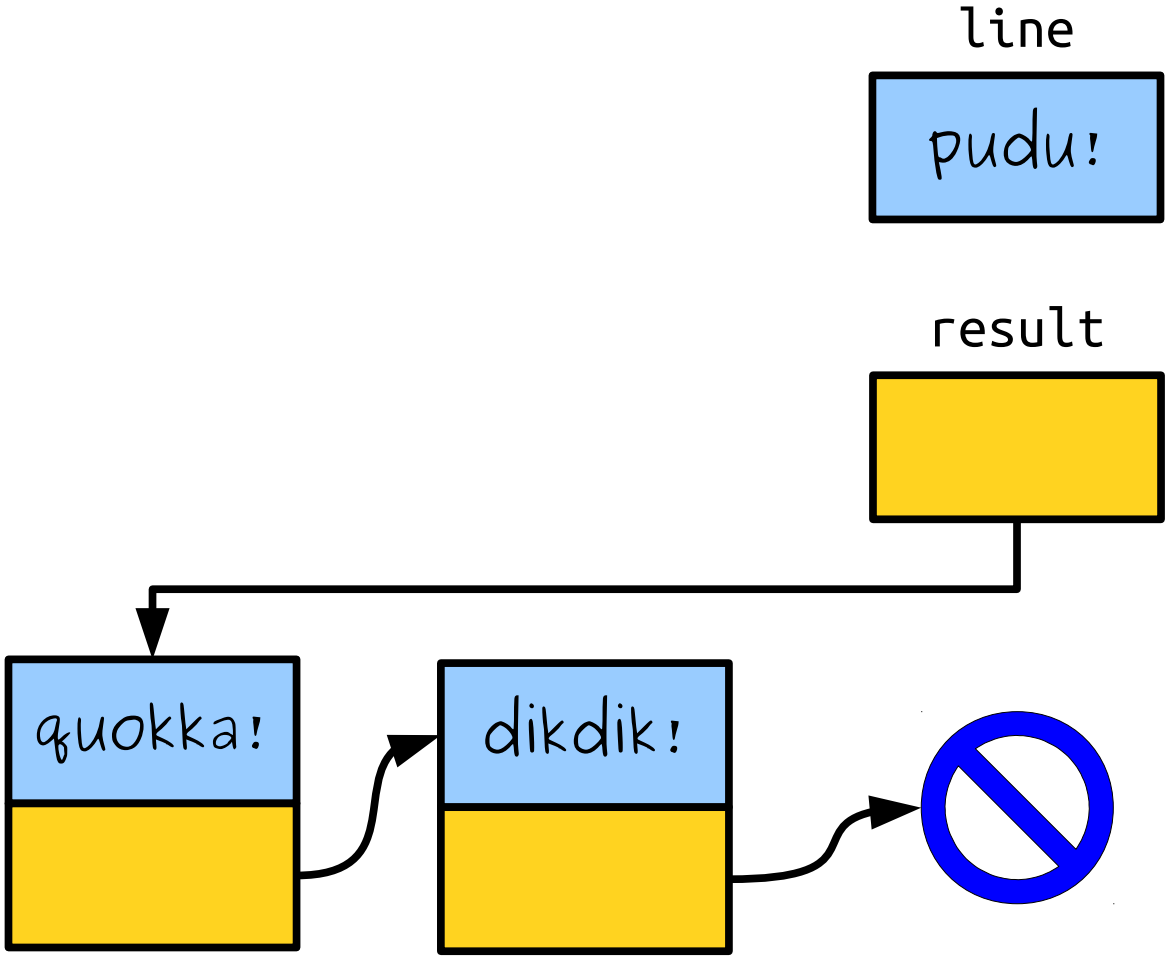
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

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



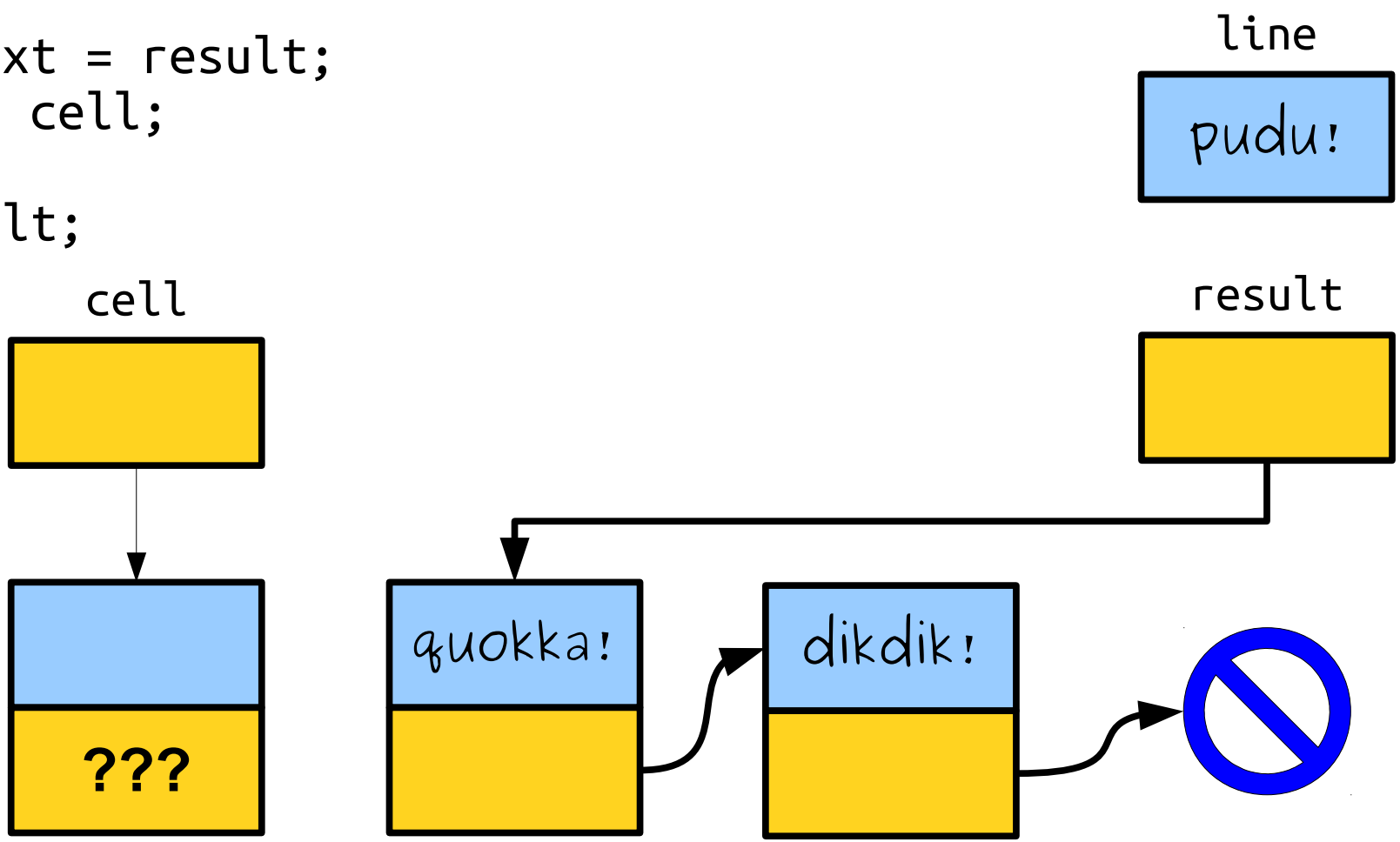
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

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



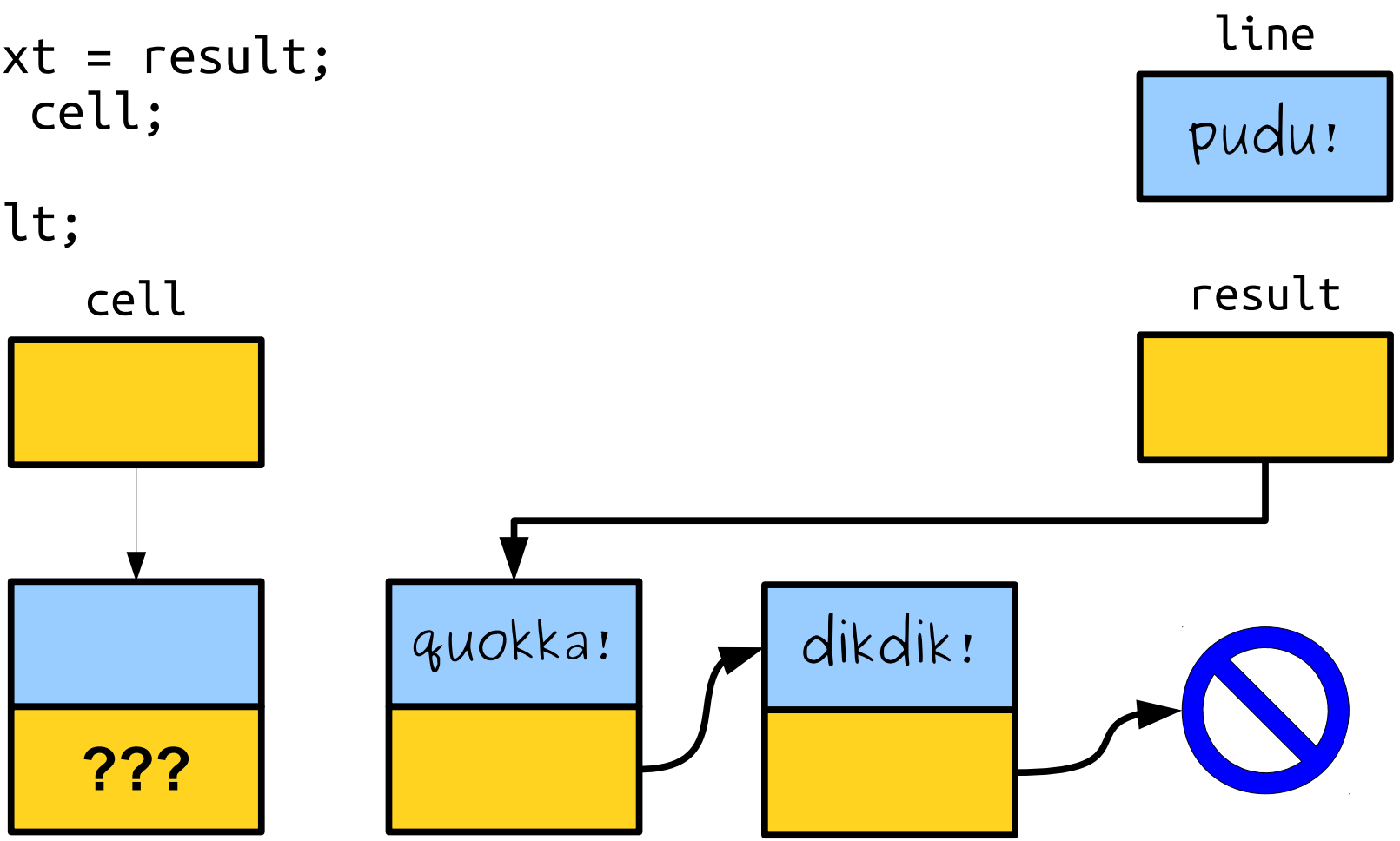
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

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



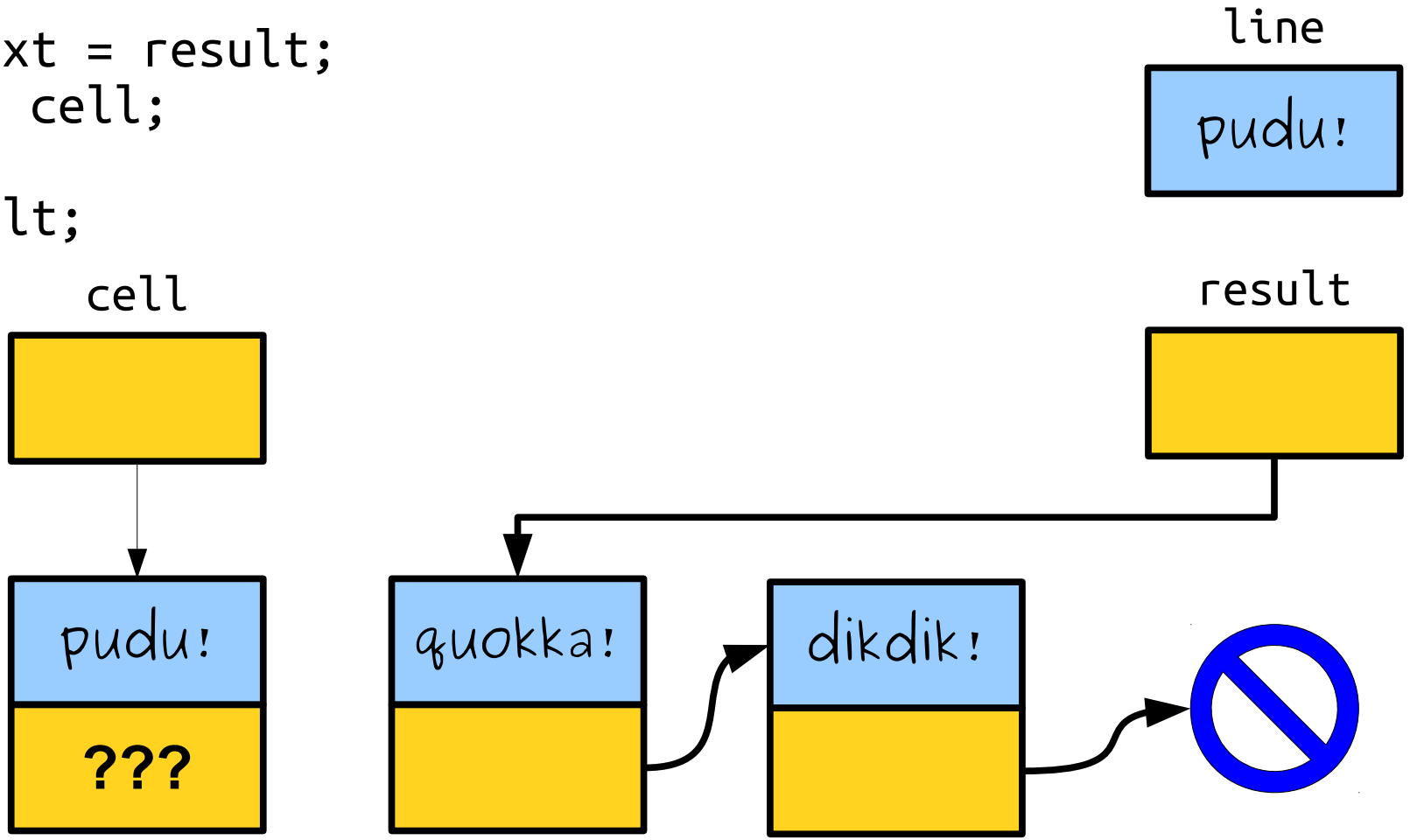
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

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



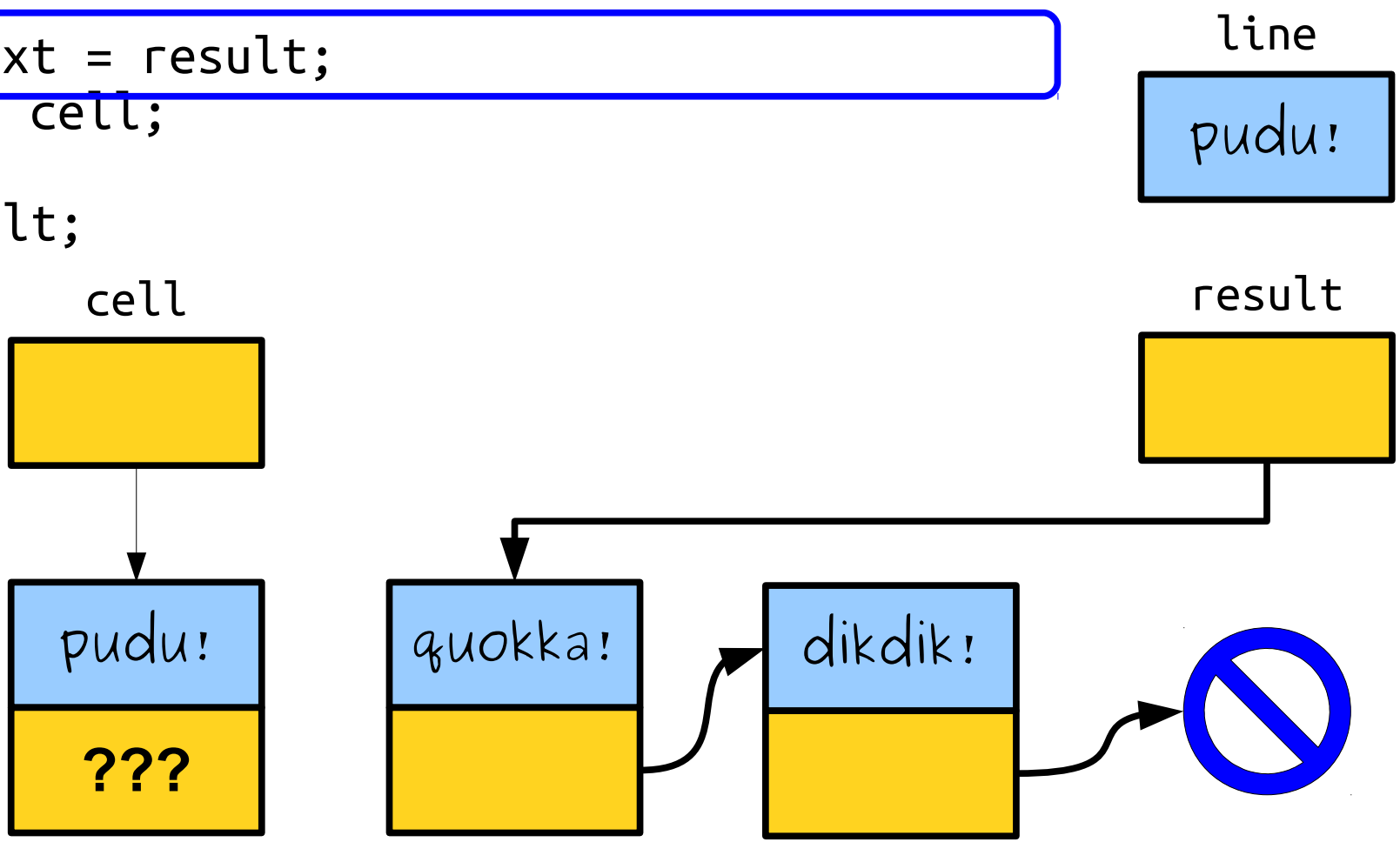
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

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



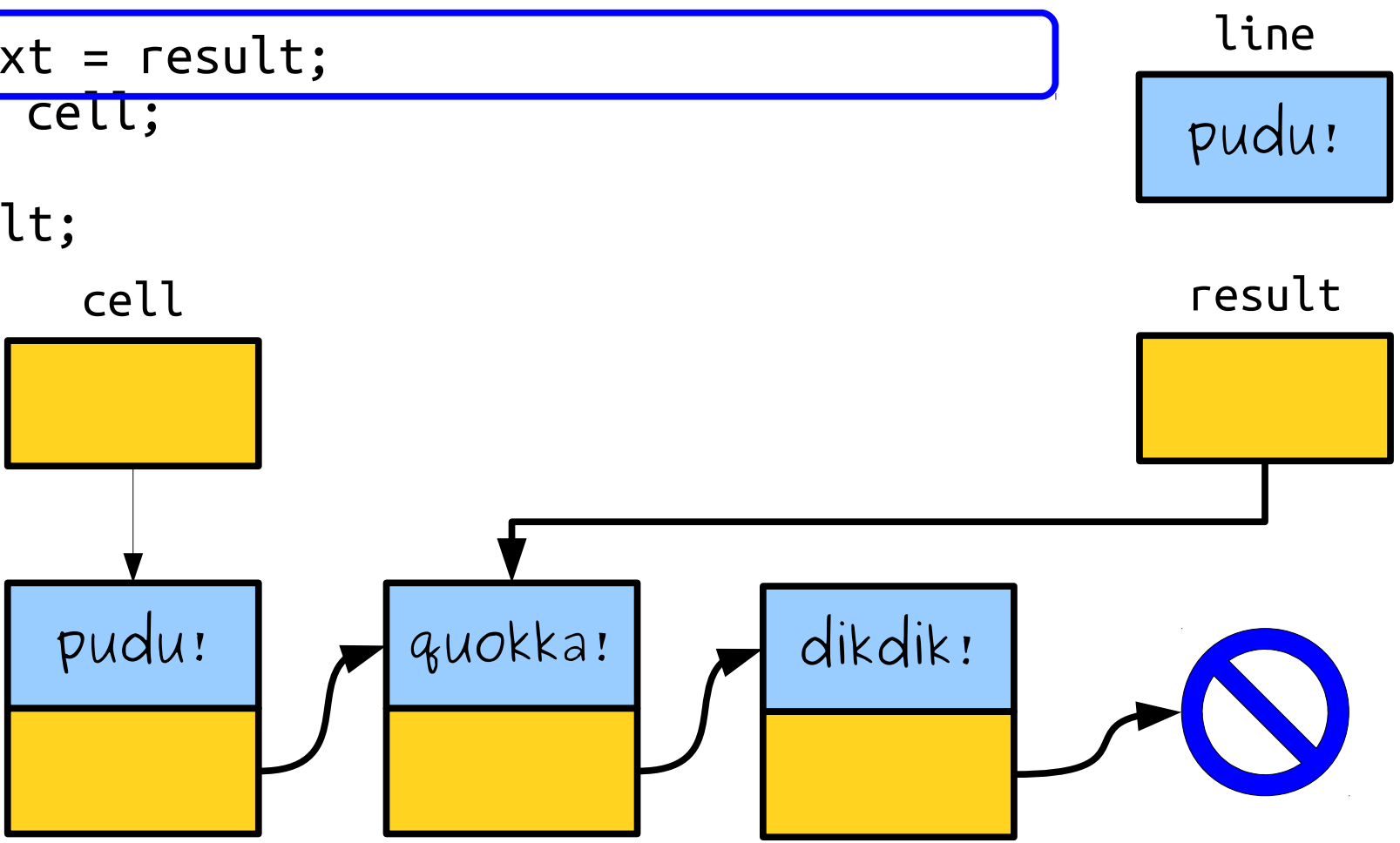
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

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



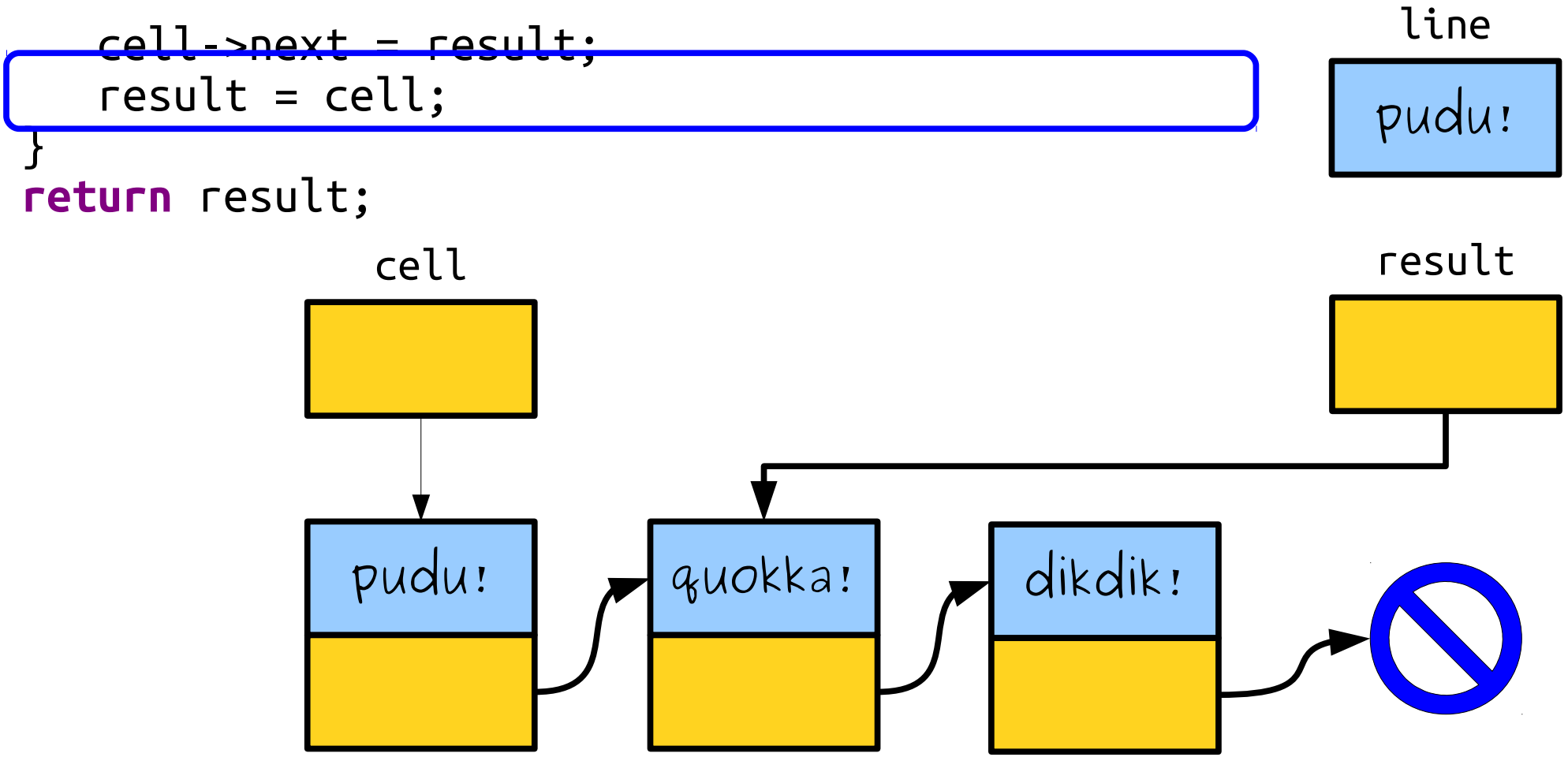
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

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



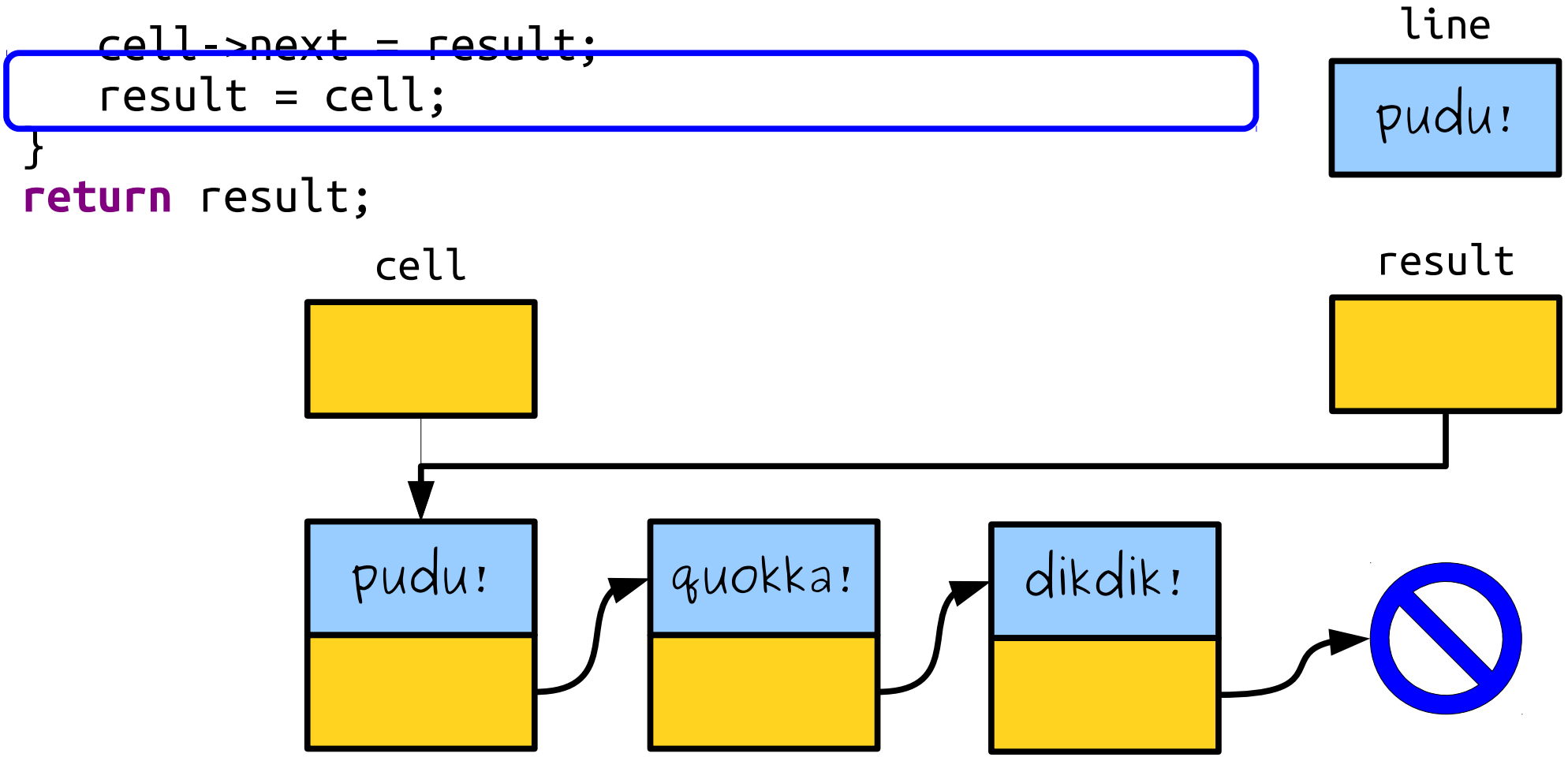
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

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



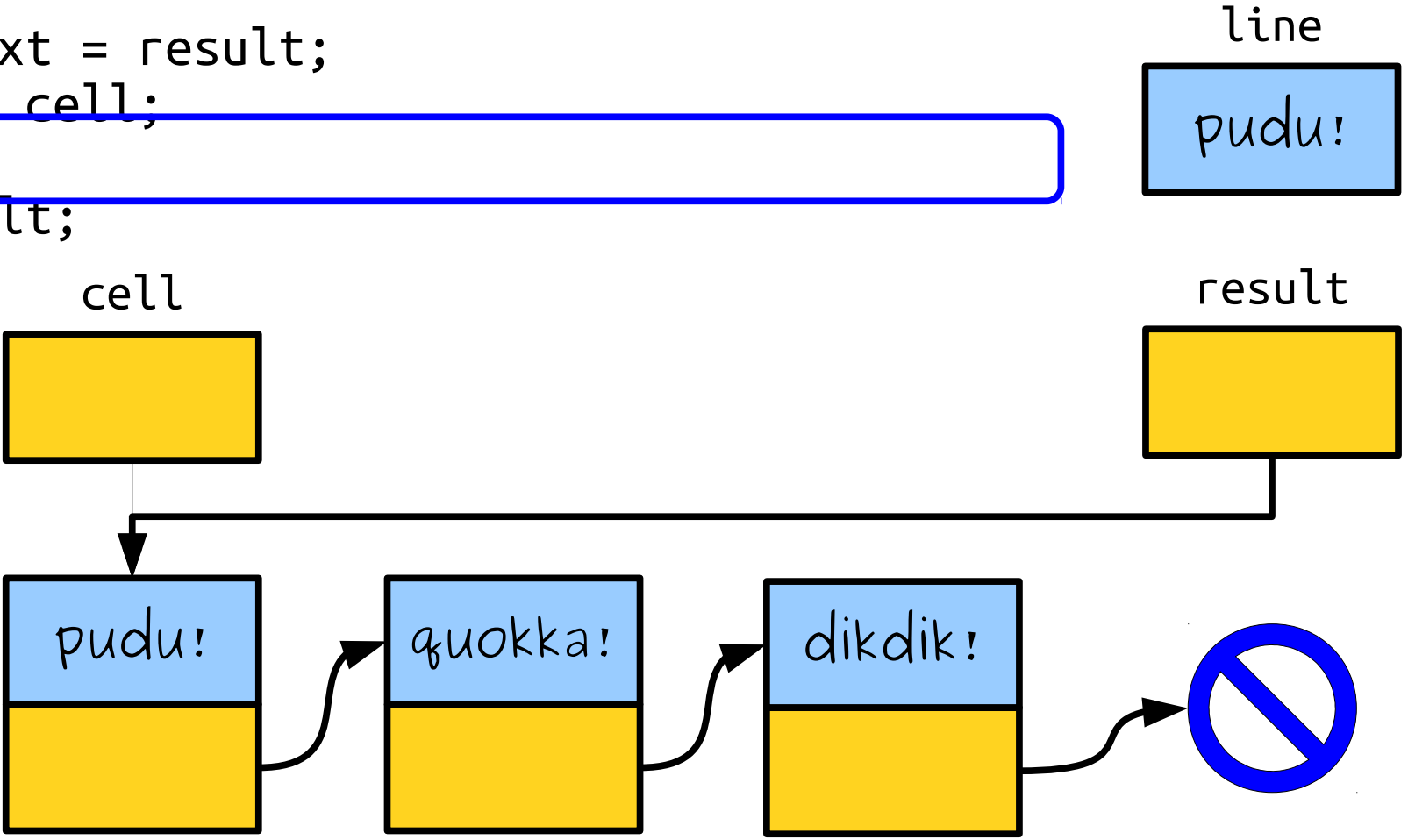
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

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

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



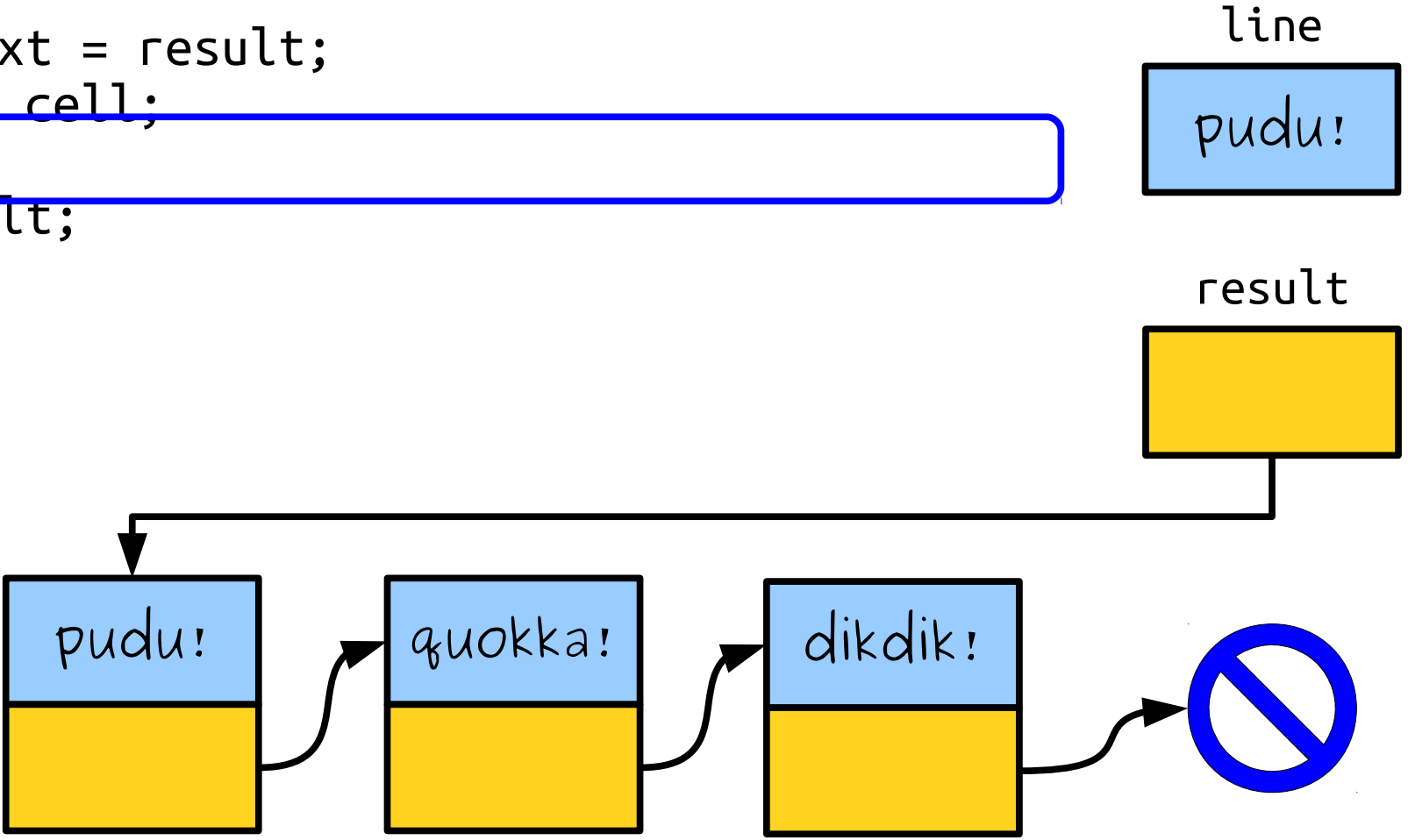
```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    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;
```

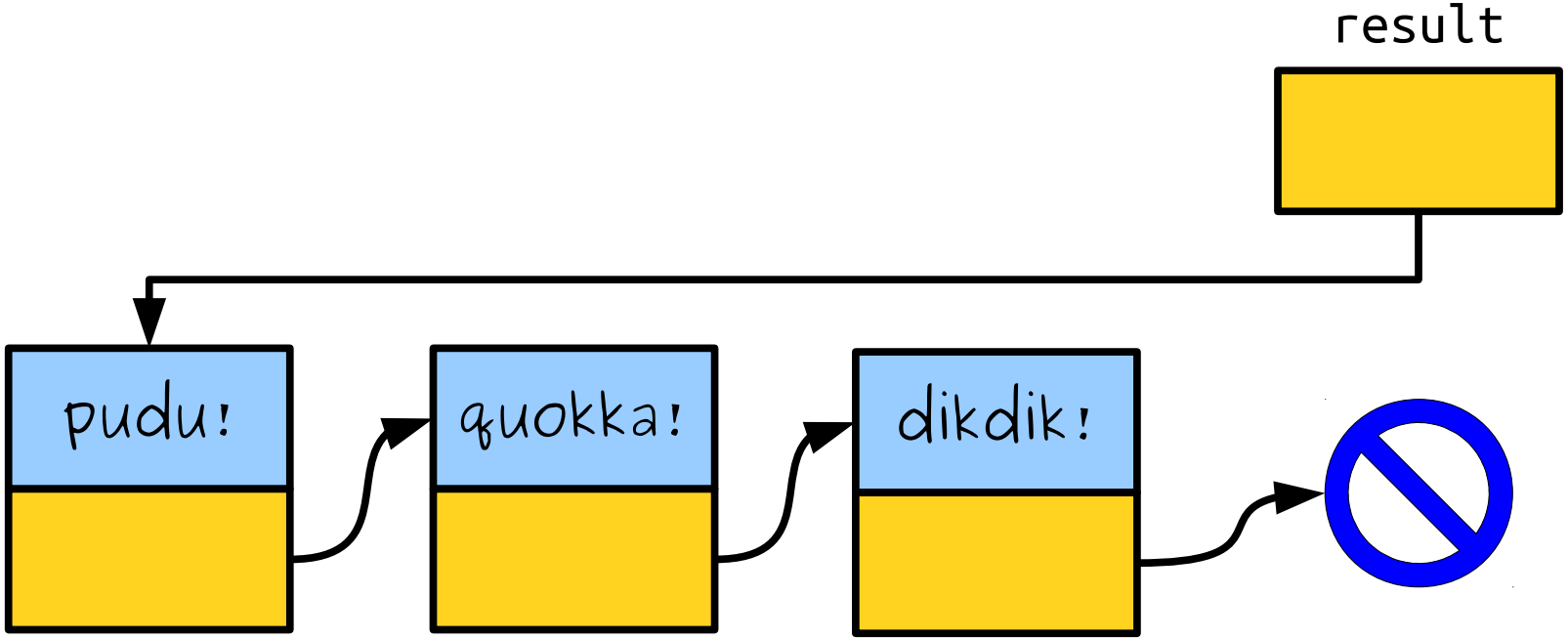


```
Cell* result = nullptr;
while (true) {
    string line = getLine("Next entry? ");
    if (line == "") break;

    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;
```

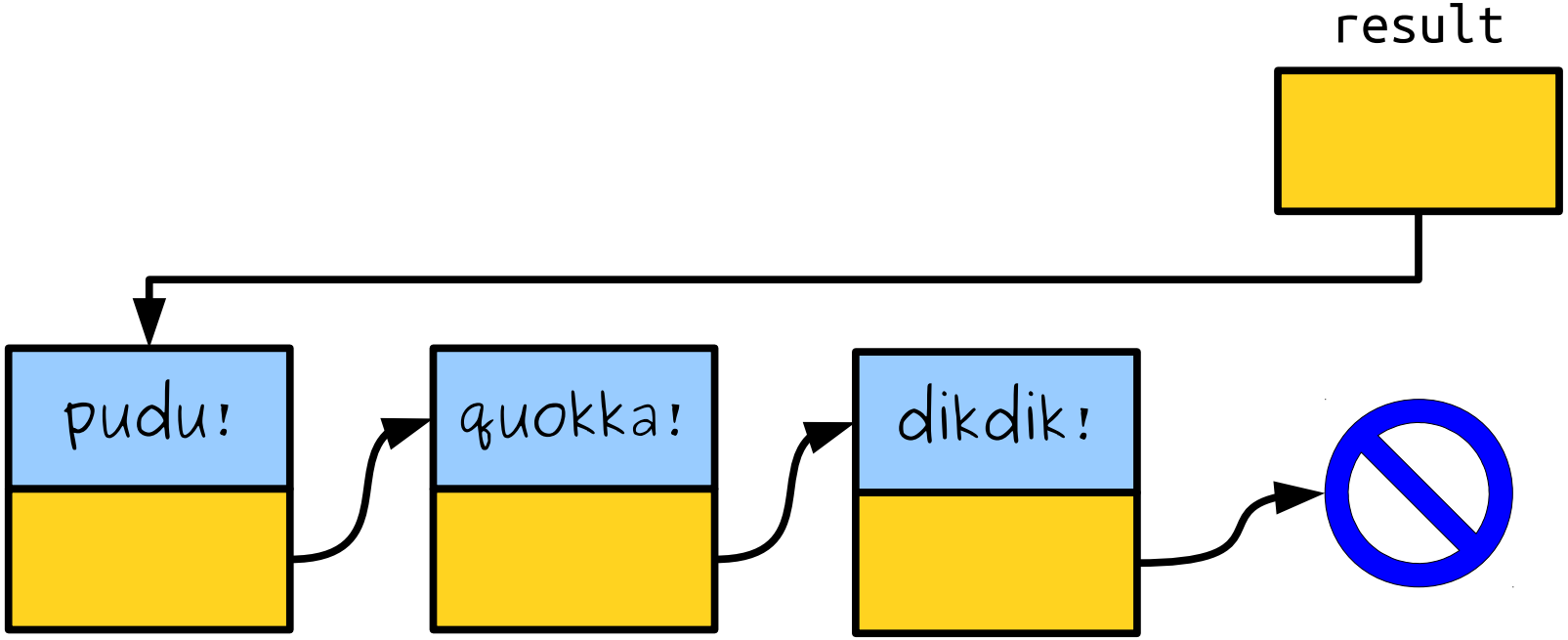
***It's a bug:*** these elements are in the wrong order!



```
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 11 of the Textbook***
  - More on pointers, arrays, lists, etc.
- ***Download BlueBook***
  - Gonna need that for the exam!

# Next Time

- ***Pointers by Reference***
  - Fun for the whole linked list family!
- ***Reimplementing Stacks and Queues***
  - Worst-case efficiency, at a price!