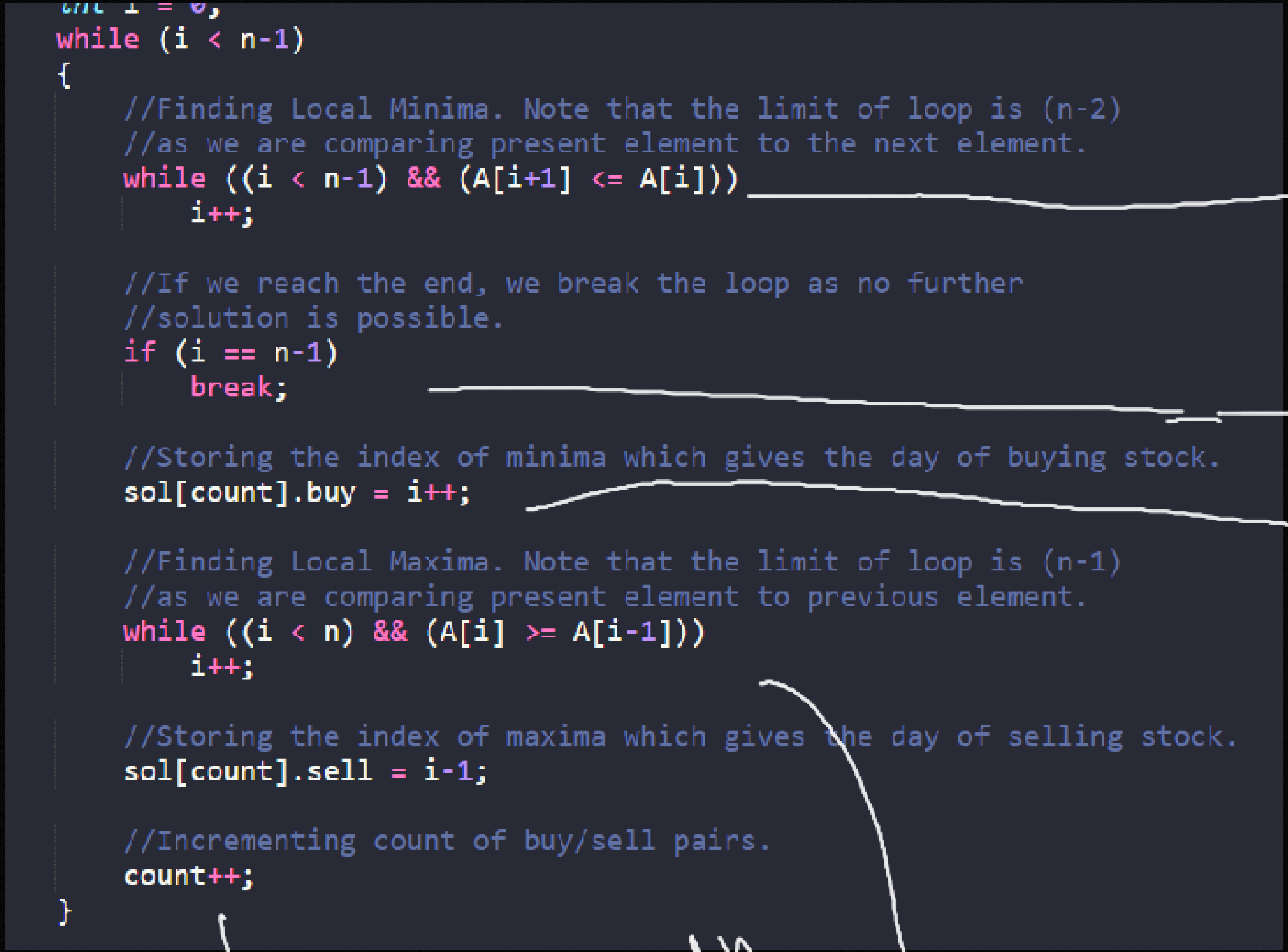
20. Stock buy and sell

Medium   Accuracy: 39.53%   Submissions: 38879   Points: 4

The cost of stock on each day is given in an array **A[]** of size **N**. Find all the days on which you buy and sell the stock so that in between those days your profit is maximum.
Note: There may be multiple possible solutions. Return any one of them.

Example 1:

```
Input:
N = 7
A[] = {100,180,260,310,40,535,695}
Output:
1
Explanation:
One possible solution is (0 3) (4 6)
We can buy stock on day 0,
and sell it on 3rd day, which will
give us maximum profit. Now, we buy
stock on day 4 and sell it on day 6.
```



```
int i = 0;
while (i < n-1)
{
    //Finding Local Minima. Note that the limit of loop is (n-2)
    //as we are comparing present element to the next element.
    while ((i < n-1) && (A[i+1] <= A[i]))
        i++;

    //If we reach the end, we break the loop as no further
    //solution is possible.
    if (i == n-1)
        break;

    //Storing the index of minima which gives the day of buying stock.
    sol[count].buy = i++;

    //Finding Local Maxima. Note that the limit of loop is (n-1)
    //as we are comparing present element to previous element.
    while ((i < n) && (A[i] >= A[i-1]))
        i++;

    //Storing the index of maxima which gives the day of selling stock.
    sol[count].sell = i-1;

    //Incrementing count of buy/sell pairs.
    count++;
}
```

finding minima

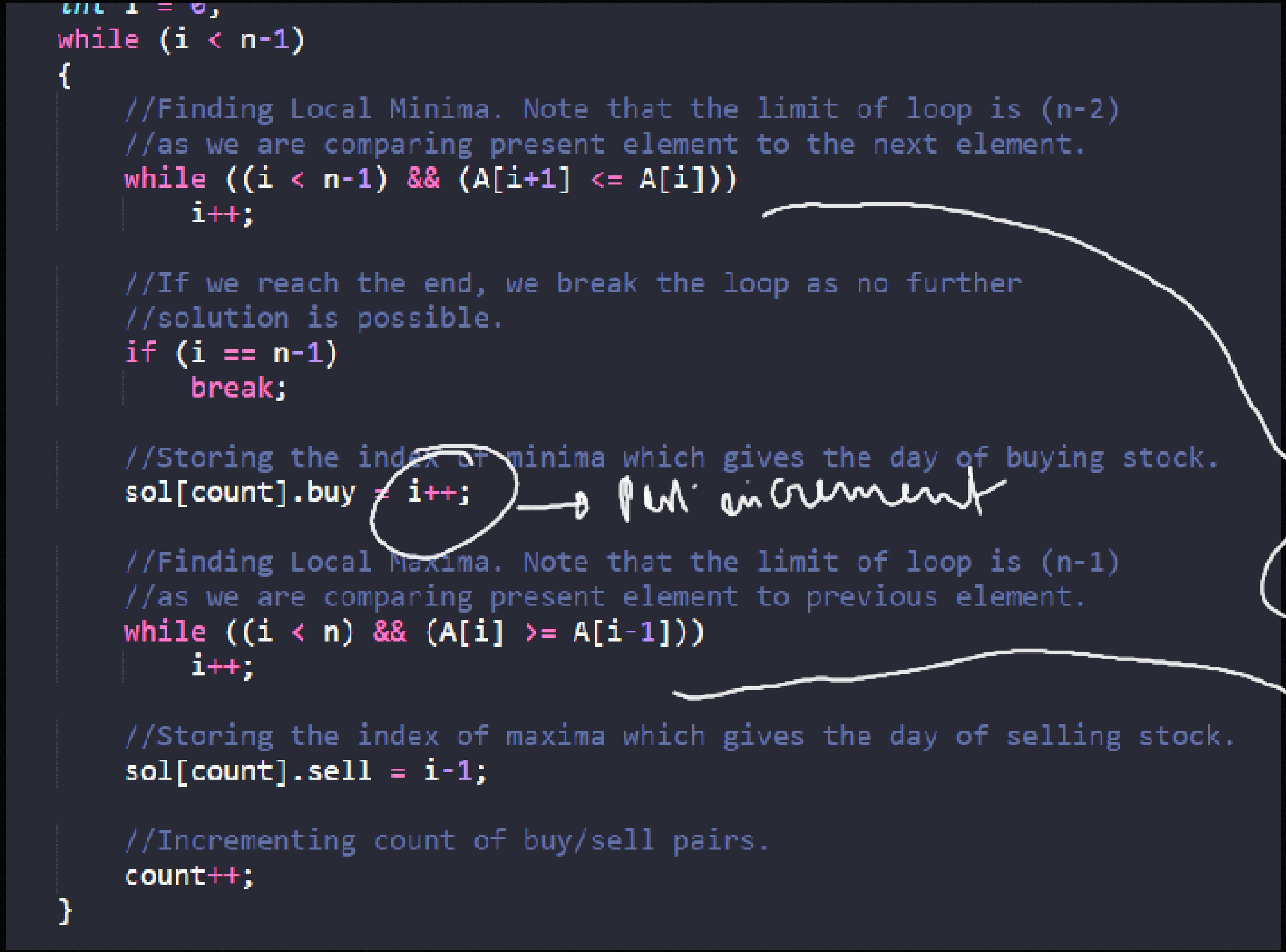
No minima exists.

↳ array in dec. order

first entry of buying on a minimal

Consecutive local maxima for full walk.

∴ i will ensure the iteration are done by part



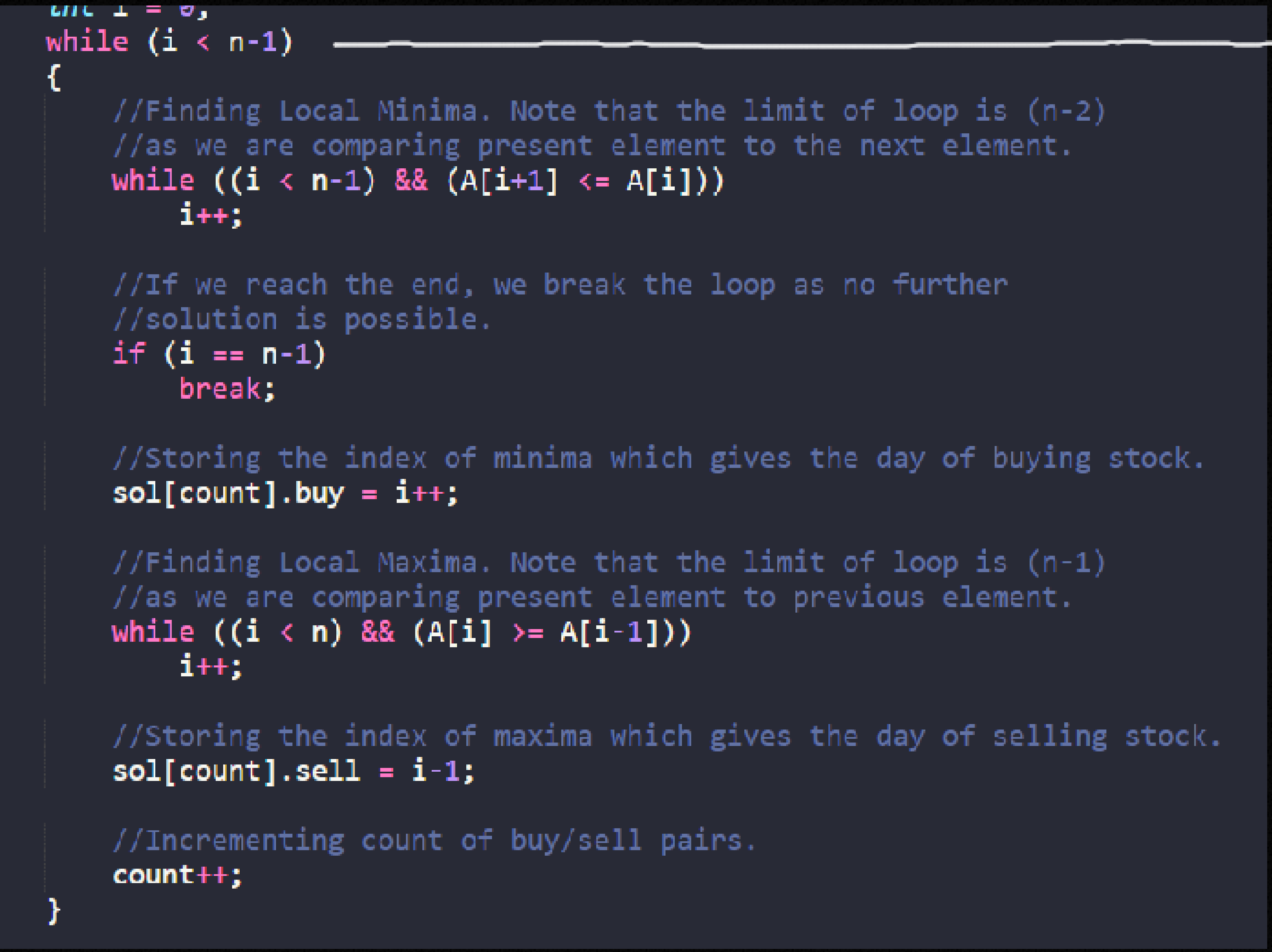
```
    int i = 0;
    while (i < n-1)
    {
        //Finding Local Minima. Note that the limit of loop is (n-2)
        //as we are comparing present element to the next element.
        while ((i < n-1) && (A[i+1] <= A[i]))
            i++;

        //If we reach the end, we break the loop as no further
        //solution is possible.
        if (i == n-1)
            break;

        //Storing the index of minima which gives the day of buying stock.
        sol[count].buy = i++;

        //Finding Local Maxima. Note that the limit of loop is (n-1)
        //as we are comparing present element to previous element.
        while ((i < n) && (A[i] >= A[i-1]))
            i++;

        //Storing the index of maxima which gives the day of selling stock.
        sol[count].sell = i-1;

        //Incrementing count of buy/sell pairs.
        count++;
    }
```

$\xrightarrow{}$ Post increment

① iterate the whole array i.e $i < n-1$ $\rightarrow$ if equal then only one element cond'n would occur -

② till the next element is smaller by for.

③



```
int i = 0;
while (i < n-1)
{
    //Finding Local Minima. Note that the limit of loop is (n-2)
    //as we are comparing present element to the next element.
    while ((i < n-1) && (A[i+1] <= A[i]))
        i++;

    //If we reach the end, we break the loop as no further
    //solution is possible.
    if (i == n-1)
        break;

    //Storing the index of minima which gives the day of buying stock.
    sol[count].buy = i++;

    //Finding Local Maxima. Note that the limit of loop is (n-1)
    //as we are comparing present element to previous element.
    while ((i < n) && (A[i] >= A[i-1]))
        i++;

    //Storing the index of maxima which gives the day of selling stock.
    sol[count].sell = i-1;

    //Incrementing count of buy/sell pairs.
    count++;
}
```

① Iterating the whole array till n.

② Finding local minima

$(i<n-1)$ && $A[i+1] <= A[i]$

i++.

③ $i == n-1 \rightarrow$ array in decreasing order

④ Storing this minima in first buy day, then incremented i.

⑤ Finding local maxima

$(i < n)$ && $(A[i] >= A[i+1])$

i++;

The loop will break when the cond^n will fail i.e $A[i] \not> A[i+1]$

↓

store i-1;

count++;