

```
#include <iostream>
using namespace std;

int iterative (int arr[], int a, int b) {
    int x = arr[b];
    int y = (a - 1);
    for (int i = a; i <= b - 1; i++) {
        if (arr[i] <= x) {
            y++;
            swap(arr[y], arr[i]);
        }
    }
    swap(arr[y + 1], arr[b]);
    return (y + 1);
}

void quick_sort(int arr[], int a, int b) {
    int stack[b-a+1];
    int top = -1;
    stack[++top]=a;
    stack[++top]=b;

    while(top >= 0){
        b = stack[top--];
        a = stack[top--];
        int p = iterative(arr, a, b);
        if(p-1>a){
            stack[++top] = a;
            stack[++top] = p - 1;
        }
        if(p+1<b){
            stack[++top] = p + 1;
            stack[++top] = b;
        }
    }
}
```

```

int rekursif (int ar[], int start, int end) {
    int pivot = ar[end];
    int partitionIndex = start;
    for (int i=start; i<end; i++) {
        if (ar[i] < pivot) {
            swap (ar[i], ar[partitionIndex]);
            partitionIndex++;
        }
    }
    swap (ar[partitionIndex], ar[end]);
    return partitionIndex;
}

```

```

void quickSort (int ar[], int start, int end) {
    if (start < end) {
        int partitionIndex = iterative (ar,start,end);
        quickSort (ar,start,partitionIndex-1);
        quickSort (ar,partitionIndex+1,end);
    }
}

```

```

int main(){

    int n, pilih;

    cout << "Masukkan Jumlah Data : " ; cin >> n ;
    int a[n];

    for (int x=0 ; x<=n-1 ; x++) {
        cout << "Masukkan Angka ke-" << x+1 << " : " ; cin >> a[x];
    }

    cout << endl;
    cout << "Urutkan Data : " << endl;
    cout << "1. Iterative Quicksort : " << endl;
    cout << "2. Recursive Quicksort : " << endl;
    cout << "Masukkan Pilihan [1..2] : " ; cin >> pilih;
    cout<<endl;

    if ( pilih == 1) {
        cout << "Pengurutan Data Dengan Iterative Quicksort" << endl;
        cout << "===== " << endl;
        cout << "Data Sebelum Diurutkan : ";

        for(int x=0; x<n; x++) {

```

```

        cout << a[x] << " ";
    }

    cout << endl;

    quickSort(a,0,n-1);
    cout << "Data Setelah Diurutkan : ";

    for(int x=0; x<n; x++){
        cout<<a[x]<<" ";
    }

    cout<<endl;
}

else if ( pilih == 2) {
    cout << "Pengurutan Data Dengan Rekursive Quicksort" << endl;
    cout << "===== " << endl;
    cout << "Data Sebelum Diurutkan : ";

    for(int x=0; x<n; x++) {
        cout << a[x] << " ";
    }

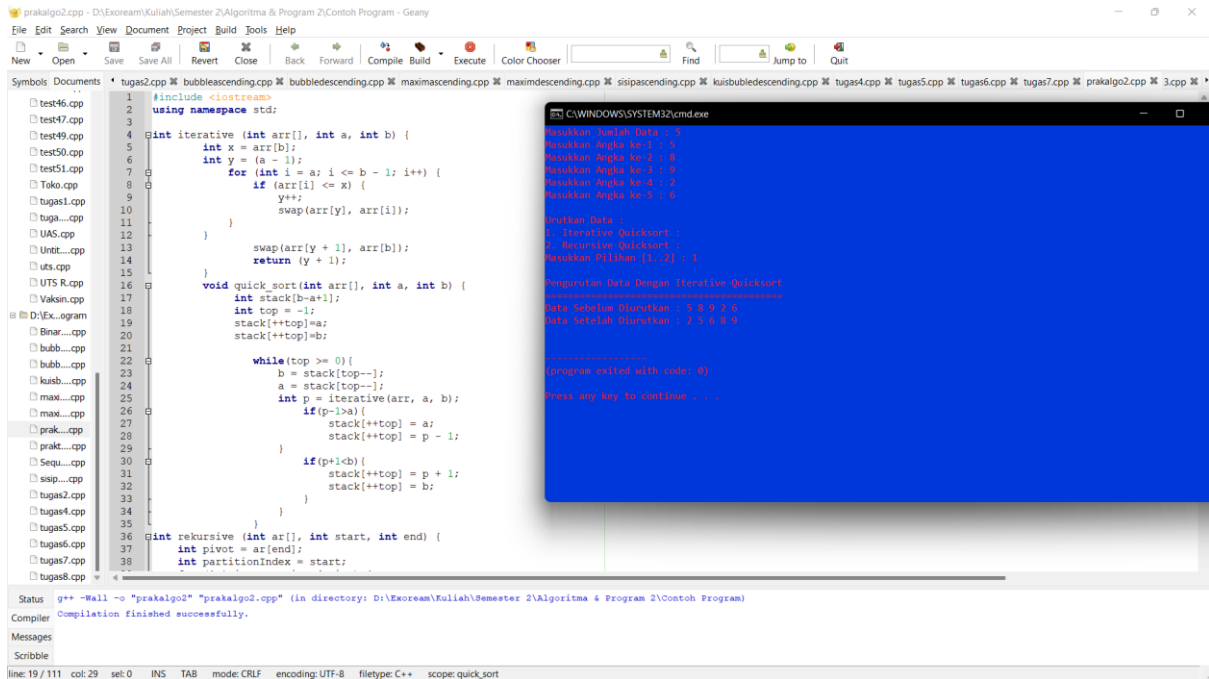
    cout << endl;
    quickSort (a,0,n-1);
    cout << "Data Setelah Diurutkan : ";

    for(int x=0;x<n;x++) {
        cout<<a[x]<<" ";
    }

    cout<<endl;
}
}

```

Iterative Quicksort



The screenshot shows a C++ IDE with a project named "prakalgo2.cpp". The code implements an iterative quicksort algorithm. The output window shows the program's execution, including input prompts, data sorting, and the final sorted array.

```
1 #include <iostream>
2 using namespace std;
3
4 int iterative (int arr[], int a, int b) {
5     int x = arr[b];
6     int y = (a - 1);
7     for (int i = a; i <= b - 1; i++) {
8         if (arr[i] <= x) {
9             y++;
10            swap(arr[y], arr[i]);
11        }
12    }
13    swap(arr[y + 1], arr[b]);
14    return (y + 1);
15 }
16
17 void quick_sort(int arr[], int a, int b) {
18     int top = -1;
19     stack[++top] = a;
20     stack[++top] = b;
21
22     while (top >= 0) {
23         b = stack[top--];
24         a = stack[top--];
25         int p = iterative(arr, a, b);
26         if (p > a) {
27             stack[++top] = a;
28             stack[++top] = p - 1;
29         }
30         if (p < b) {
31             stack[++top] = p + 1;
32             stack[++top] = b;
33         }
34     }
35 }
36
37 int rekursi (int ar[], int start, int end) {
38     int pivot = ar[end];
39     int partitionIndex = start;
```

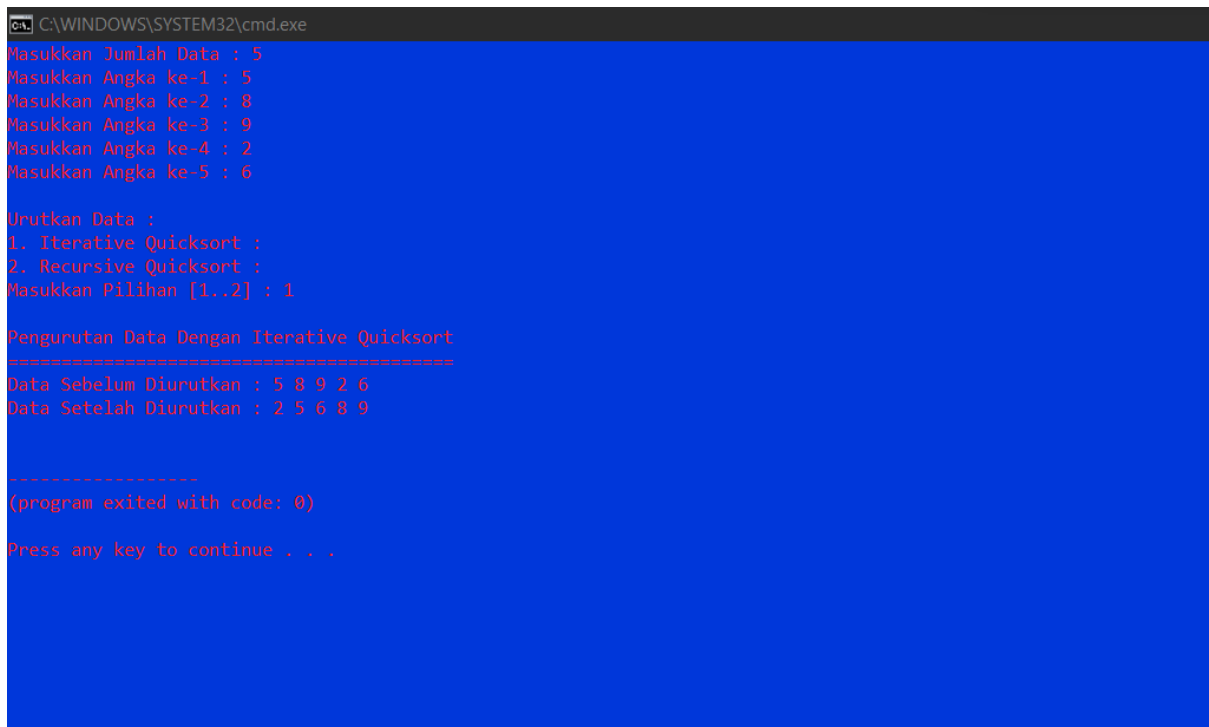
Output:

```
Masukkan Jumlah Data : 5
Masukkan Angka ke-1 : 5
Masukkan Angka ke-2 : 8
Masukkan Angka ke-3 : 9
Masukkan Angka ke-4 : 2
Masukkan Angka ke-5 : 6

Urutkan Data :
1. Iterative Quicksort :
2. Recursive Quicksort :
Masukkan Pilihan [1..2] : 1

Pengurutan Data Dengan Iterative Quicksort
=====
Data Sebelum Diurutkan : 5 8 9 2 6
Data Setelah Diurutkan : 2 5 6 8 9

-----
(program exited with code: 0)
Press any key to continue . . .
```



The screenshot shows a Windows command prompt window with the following text:

```
C:\WINDOWS\SYSTEM32\cmd.exe

Masukkan Jumlah Data : 5
Masukkan Angka ke-1 : 5
Masukkan Angka ke-2 : 8
Masukkan Angka ke-3 : 9
Masukkan Angka ke-4 : 2
Masukkan Angka ke-5 : 6

Urutkan Data :
1. Iterative Quicksort :
2. Recursive Quicksort :
Masukkan Pilihan [1..2] : 1

Pengurutan Data Dengan Iterative Quicksort
=====
Data Sebelum Diurutkan : 5 8 9 2 6
Data Setelah Diurutkan : 2 5 6 8 9

-----
(program exited with code: 0)
Press any key to continue . . .
```

Recursive Quicksort

The screenshot shows a C++ IDE with a file explorer on the left containing various test and task files. The main editor displays the implementation of a recursive quicksort algorithm. The code includes a partitioning function, a recursive sorting function, and a main function that takes user input for the number of elements and the elements themselves. A status bar at the bottom indicates the compilation was successful.

```
1 #include <iostream>
2 using namespace std;
3
4 int iterative (int arr[], int a, int b) {
5     int x = arr[b];
6     int y = (a - 1);
7     for (int i = a; i <= b - 1; i++) {
8         if (arr[i] <= x) {
9             y++;
10            swap(arr[y], arr[i]);
11        }
12    }
13    swap(arr[y + 1], arr[b]);
14    return (y + 1);
15 }
16
17 void quick sort (int arr[], int a, int b) {
18     int stack[b-a+1];
19     int top = -1;
20     stack[++top] = a;
21     stack[++top] = b;
22
23     while (top >= 0) {
24         b = stack[top--];
25         a = stack[top--];
26         int p = iterative(arr, a, b);
27         if (p > a) {
28             stack[++top] = a;
29             stack[++top] = p - 1;
30         }
31         if (p < b) {
32             stack[++top] = p + 1;
33             stack[++top] = b;
34         }
35     }
36 }
37
38 int rekursif (int ar[], int start, int end) {
39     int pivot = ar[end];
40     int partitionIndex = start;
41
42     for (int i = start; i < end; i++) {
43         if (ar[i] < pivot) {
44             swap(ar[i], ar[partitionIndex]);
45             partitionIndex++;
46         }
47     }
48     swap(ar[partitionIndex], ar[end]);
49
50     if (partitionIndex > start)
51         rekursif(ar, start, partitionIndex - 1);
52     if (partitionIndex < end)
53         rekursif(ar, partitionIndex + 1, end);
54 }
```

Execution output in the command prompt window:

```
Masukkan Jumlah Data : 4
Masukkan Angka ke-1 : 15
Masukkan Angka ke-2 : 8
Masukkan Angka ke-3 : 19
Masukkan Angka ke-4 : 5

Urutkan Data :
1. Iterative Quicksort :
2. Recursive Quicksort :
Masukkan Pilihan [1..2] : 2

Pengurutan Data Dengan Rekursif Quicksort
=====
Data Sebelum Diurutkan : 15 8 19 5
Data Setelah Diurutkan : 5 8 15 19

-----
(program exited with code: 0)
Press any key to continue . . .
```

This block provides a detailed view of the command prompt output from the previous image. It shows the user input for the number of elements (4) and the elements themselves (15, 8, 19, 5). It then displays the sorting process using recursive quicksort, showing the data before and after sorting. The program exits with code 0, and the user is prompted to press any key to continue.

```
C:\WINDOWS\SYSTEM32\cmd.exe

Masukkan Jumlah Data : 4
Masukkan Angka ke-1 : 15
Masukkan Angka ke-2 : 8
Masukkan Angka ke-3 : 19
Masukkan Angka ke-4 : 5

Urutkan Data :
1. Iterative Quicksort :
2. Recursive Quicksort :
Masukkan Pilihan [1..2] : 2

Pengurutan Data Dengan Rekursif Quicksort
=====
Data Sebelum Diurutkan : 15 8 19 5
Data Setelah Diurutkan : 5 8 15 19

-----
(program exited with code: 0)
Press any key to continue . . .
```