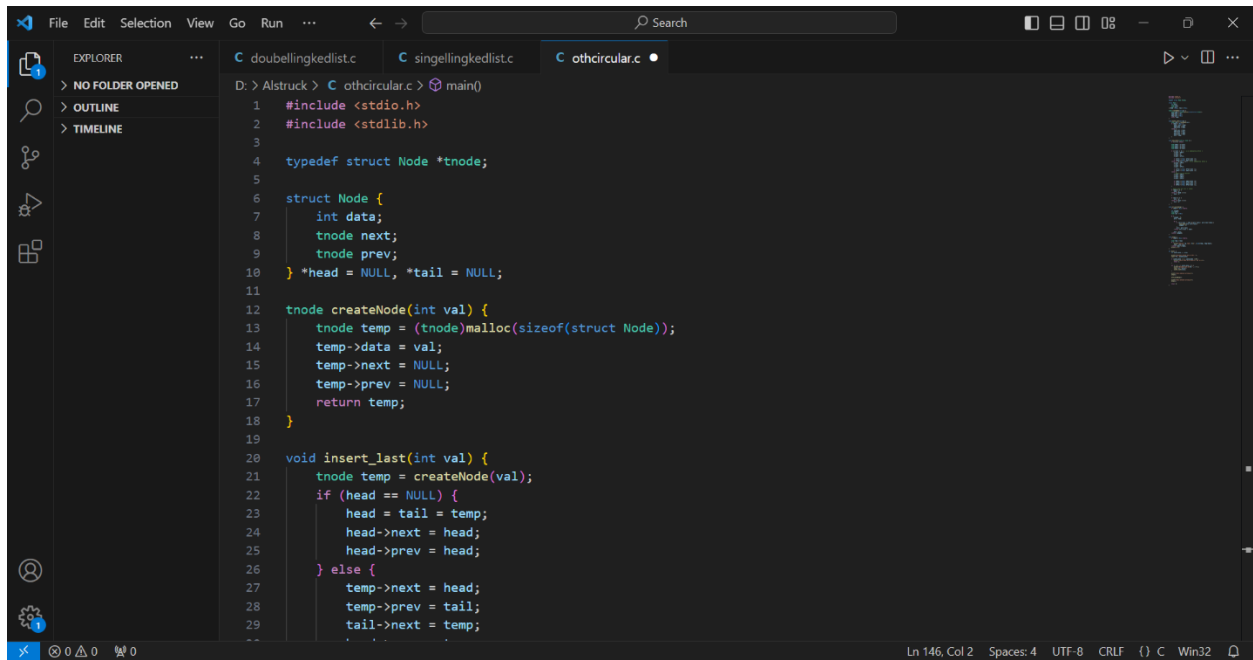


Nama : Rizky Dwi Firmansah

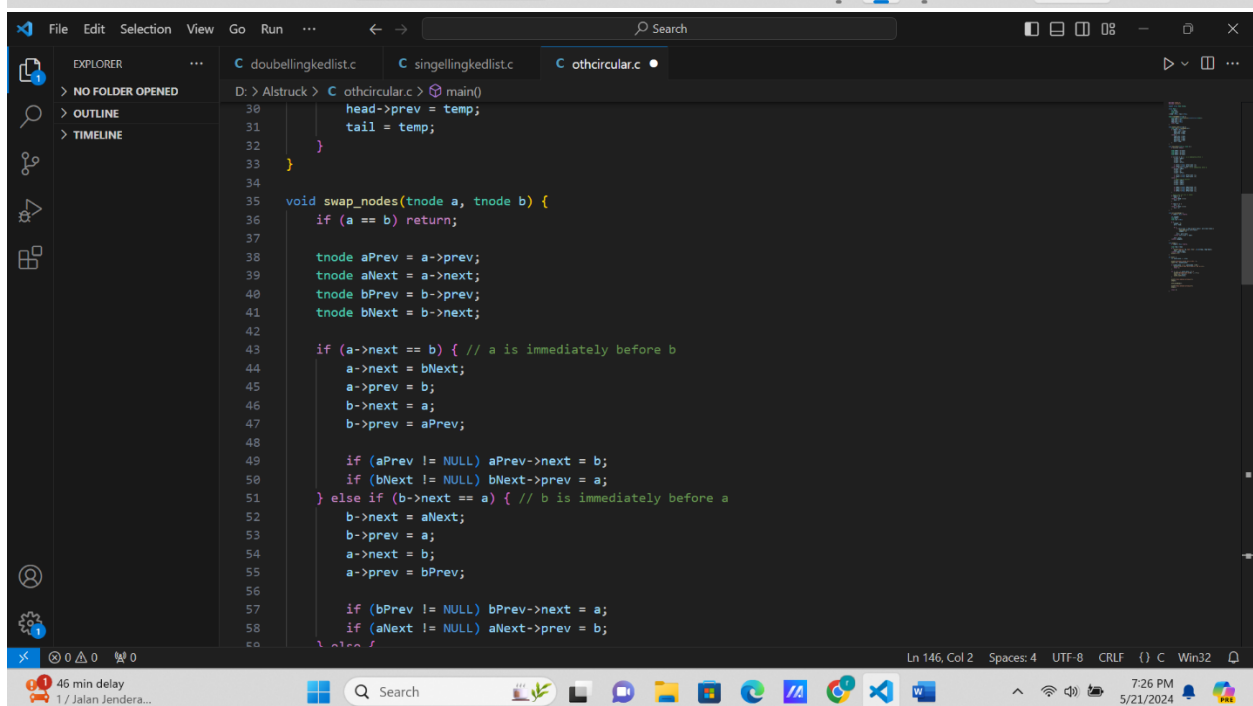
Nim :1203230049

Kelas : IF - 03 - 02

## INPUT



```
File Edit Selection View Go Run ... Search
EXPLORER
> NO FOLDER OPENED
> OUTLINE
> TIMELINE
D: > Alstruck > C othcircular.c > main()
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 typedef struct Node *tnode;
5
6 struct Node {
7     int data;
8     tnode next;
9     tnode prev;
10 } *head = NULL, *tail = NULL;
11
12 tnode createNode(int val) {
13     tnode temp = (tnode)malloc(sizeof(struct Node));
14     temp->data = val;
15     temp->next = NULL;
16     temp->prev = NULL;
17     return temp;
18 }
19
20 void insert_last(int val) {
21     tnode temp = createNode(val);
22     if (head == NULL) {
23         head = tail = temp;
24         head->next = head;
25         head->prev = head;
26     } else {
27         temp->next = head;
28         temp->prev = tail;
29         tail->next = temp;
```



```
30     head->prev = temp;
31     tail = temp;
32 }
33 }
34
35 void swap_nodes(tnode a, tnode b) {
36     if (a == b) return;
37
38     tnode aPrev = a->prev;
39     tnode aNext = a->next;
40     tnode bPrev = b->prev;
41     tnode bNext = b->next;
42
43     if (a->next == b) { // a is immediately before b
44         a->next = bNext;
45         a->prev = b;
46         b->next = a;
47         b->prev = aPrev;
48
49         if (aPrev != NULL) aPrev->next = b;
50         if (bNext != NULL) bNext->prev = a;
51     } else if (b->next == a) { // b is immediately before a
52         b->next = aNext;
53         b->prev = a;
54         a->next = b;
55         a->prev = bPrev;
56
57         if (bPrev != NULL) bPrev->next = a;
58         if (aNext != NULL) aNext->prev = b;
59     }
60 }
```

```
> NO FOLDER OPENED
> OUTLINE
> TIMELINE

D: > Alstruck > C othcircular.c > main()

59     } else {
60         // Nodes are not adjacent
61         a->next = bNext;
62         a->prev = bPrev;
63         b->next = aNext;
64         b->prev = aPrev;
65
66         if (aNext != NULL) aNext->prev = b;
67         if (aPrev != NULL) aPrev->next = b;
68         if (bNext != NULL) bNext->prev = a;
69         if (bPrev != NULL) bPrev->next = a;
70     }
71
72     // Update head and tail if needed
73     if (head == a) {
74         head = b;
75     } else if (head == b) {
76         head = a;
77     }
78
79     if (tail == a) {
80         tail = b;
81     } else if (tail == b) {
82         tail = a;
83     }
84 }
85
86 void sort_ascending() {
87     if (head == NULL) return;
```

```
> NO FOLDER OPENED
> OUTLINE
> TIMELINE

D: > Alstruck > C othcircular.c > main()

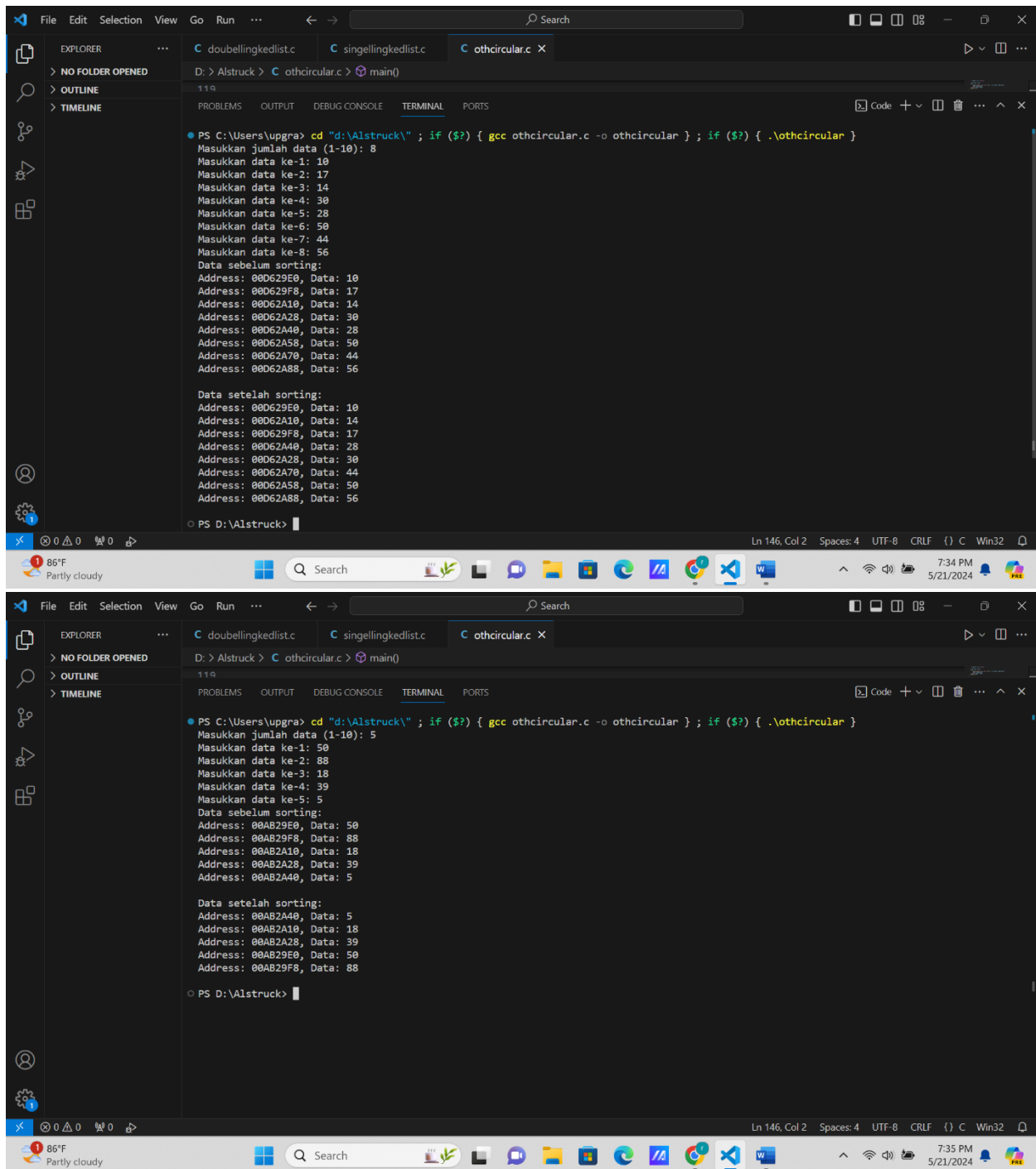
89     int swapped;
90     tnode ptr1;
91     tnode lptr = NULL;
92
93     do {
94         swapped = 0;
95         ptr1 = head;
96
97         do {
98             if (ptr1->next != head && ptr1->data > ptr1->next->data) {
99                 swap_nodes(ptr1, ptr1->next);
100                 swapped = 1;
101             }
102             ptr1 = ptr1->next;
103         } while (ptr1->next != head);
104
105         lptr = ptr1;
106     } while (swapped);
107 }
108
109 void cetak() {
110     if (head == NULL) return;
111
112     tnode temp = head;
113     do {
114         printf("Address: %p, Data: %d\n", (void*)temp, temp->data);
115         temp = temp->next;
116     } while (temp != head);
117     printf("\n");
118 }
```

```
> NO FOLDER OPENED
> OUTLINE
> TIMELINE

D: > Alstruck > C othcircular.c > main()

119
120 int main() {
121     int jumlah_data, i, nilai;
122
123     printf("Masukkan jumlah data (1-10): ");
124     scanf("%d", &jumlah_data);
125
126     if (jumlah_data < 1 || jumlah_data > 10) {
127         printf("Jumlah data harus antara 1 dan 10.\n");
128         return 1;
129     }
130
131     for (i = 0; i < jumlah_data; i++) {
132         printf("Masukkan data ke-%d: ", i + 1);
133         scanf("%d", &nilai);
134         insert_last(nilai);
135     }
136
137     printf("Data sebelum sorting:\n");
138     cetak();
139
140     sort_ascending();
141
142     printf("Data setelah sorting:\n");
143     cetak();
144
145     return 0;
146 }
```

## OUTPUT



```
PS C:\Users\upgra> cd "d:\Alstruck\" ; if ($?) { gcc othcircular.c -o othcircular } ; if ($?) { .\othcircular }
Masukkan jumlah data (1-10): 8
Masukkan data ke-1: 10
Masukkan data ke-2: 17
Masukkan data ke-3: 14
Masukkan data ke-4: 30
Masukkan data ke-5: 28
Masukkan data ke-6: 50
Masukkan data ke-7: 44
Masukkan data ke-8: 56
Data sebelum sorting:
Address: 000629E0, Data: 10
Address: 000629F8, Data: 17
Address: 00062A10, Data: 14
Address: 00062A28, Data: 30
Address: 00062A40, Data: 28
Address: 00062A58, Data: 50
Address: 00062A70, Data: 44
Address: 00062A88, Data: 56
Data setelah sorting:
Address: 000629E0, Data: 10
Address: 00062A10, Data: 14
Address: 000629F8, Data: 17
Address: 00062A40, Data: 28
Address: 00062A28, Data: 30
Address: 00062A70, Data: 44
Address: 00062A58, Data: 50
Address: 00062A88, Data: 56
PS D:\Alstruck>
```

```
PS C:\Users\upgra> cd "d:\Alstruck\" ; if ($?) { gcc othcircular.c -o othcircular } ; if ($?) { .\othcircular }
Masukkan jumlah data (1-10): 5
Masukkan data ke-1: 50
Masukkan data ke-2: 88
Masukkan data ke-3: 18
Masukkan data ke-4: 39
Masukkan data ke-5: 5
Data sebelum sorting:
Address: 00AB29E0, Data: 50
Address: 00AB29F8, Data: 88
Address: 00AB2A10, Data: 18
Address: 00AB2A28, Data: 39
Address: 00AB2A40, Data: 5
Data setelah sorting:
Address: 00AB2A40, Data: 5
Address: 00AB2A10, Data: 18
Address: 00AB2A28, Data: 39
Address: 00AB29E0, Data: 50
Address: 00AB29F8, Data: 88
PS D:\Alstruck>
```

## PENJELASAN

```
3
4 typedef struct Node *tnode;
5
6 struct Node {
7     int data;
8     tnode next;
9     tnode prev;
10 } *head = NULL, *tail = NULL;
11
```

1. `typedef struct Node *tnode;`: Mendefinisikan tipe baru `tnode` sebagai pointer ke `struct Node`.
2. `struct Node`: Mendefinisikan struktur `Node` yang memiliki tiga anggota: `int data`, `tnode next`, dan `tnode prev`.
3. `head` dan `tail`: Pointer global untuk menunjuk ke elemen pertama dan terakhir dari linked list.

```
11
12 tnode createNode(int val) {
13     tnode temp = (tnode)malloc(sizeof(struct Node));
14     temp->data = val;
15     temp->next = NULL;
16     temp->prev = NULL;
17     return temp;
18 }
19
```

1. `tnode createNode(int val)`: Fungsi untuk membuat node baru dengan nilai `val`.
2. `malloc(sizeof(struct Node))`: Mengalokasikan memori untuk node baru.
3. `temp->data = val`: Mengisi data node dengan `val`.
4. `temp->next = NULL` dan `temp->prev = NULL`: Menginisialisasi pointer `next` dan `prev` ke `NULL`.
5. `return temp`: Mengembalikan pointer ke node baru.

```
19
20 void insert_last(int val) {
21     tnode temp = createNode(val);
22     if (head == NULL) {
23         head = tail = temp;
24         head->next = head;
25         head->prev = head;
26     } else {
27         temp->next = head;
28         temp->prev = tail;
29         tail->next = temp;
30         head->prev = temp;
31         tail = temp;
32     }
33 }
34
```

1. `void insert_last(int val)`: Fungsi untuk menambahkan node baru dengan nilai `val` di akhir linked list.
2. `createNode(val)`: Membuat node baru.
3. `if (head == NULL)`: Jika list kosong, node baru menjadi `head` dan `tail`, dengan pointer `next` dan `prev` menunjuk pada diri sendiri.
4. `else`: Jika list tidak kosong, node baru ditambahkan di akhir list dan `tail` diperbarui.

```
35 void swap_nodes(tnode a, tnode b) {
36     if (a == b) return;
37
38     tnode aPrev = a->prev;
39     tnode aNext = a->next;
40     tnode bPrev = b->prev;
41     tnode bNext = b->next;
42
43     if (a->next == b) { // a is immediately before b
44         a->next = bNext;
45         a->prev = b;
46         b->next = a;
47         b->prev = aPrev;
48
49         if (aPrev != NULL) aPrev->next = b;
50         if (bNext != NULL) bNext->prev = a;
51     } else if (b->next == a) { // b is immediately before a
52         b->next = aNext;
53         b->prev = a;
54         a->next = b;
55         a->prev = bPrev;
56
57         if (bPrev != NULL) bPrev->next = a;
58         if (aNext != NULL) aNext->prev = b;
59     } else {
60         // Nodes are not adjacent
61         a->next = bNext;
62         a->prev = bPrev;
63         b->next = aNext;
64         b->prev = aPrev;
65
66         if (aNext != NULL) aNext->prev = b;
67         if (aPrev != NULL) aPrev->next = b;
68         if (bNext != NULL) bNext->prev = a;
69         if (bPrev != NULL) bPrev->next = a;
70     }
71
72     // Update head and tail if needed
73     if (head == a) {
74         head = b;
75     } else if (head == b) {
76         head = a;
77     }
78
79     if (tail == a) {
80         tail = b;
81     } else if (tail == b) {
82         tail = a;
83     }
84 }
```

- 1 void swap\_nodes(tnode a, tnode b): Fungsi untuk menukar dua node a dan b.
- 2 if (a == b) return;; Jika a dan b adalah node yang sama, keluar dari fungsi.
- 3 tnode aPrev = a->prev, dll.: Menyimpan pointer prev dan next dari a dan b.
- 4 if (a->next == b): Jika a dan b bersebelahan, perbarui pointer untuk menukar keduanya.
- 5 else if (b->next == a): Jika b dan a bersebelahan, perbarui pointer untuk menukar keduanya.
- 6 else: Jika a dan b tidak bersebelahan, perbarui pointer untuk menukar mereka.
- 7 if (head == a): Perbarui head jika head adalah salah satu dari node yang ditukar.
- 8 if (tail == a): Perbarui tail jika tail adalah salah satu dari node yang ditukar.

```

85
86 void sort_ascending() {
87     if (head == NULL) return;
88
89     int swapped;
90     tnode ptr1;
91     tnode lptr = NULL;
92
93     do {
94         swapped = 0;
95         ptr1 = head;
96
97         do {
98             if (ptr1->next != head && ptr1->data > ptr1->next->data) {
99                 swap_nodes(ptr1, ptr1->next);
100                 swapped = 1;
101             }
102             ptr1 = ptr1->next;
103         } while (ptr1->next != head);
104
105         lptr = ptr1;
106     } while (swapped);
107 }

```

- 1 void sort\_ascending(): Fungsi untuk mengurutkan linked list dalam urutan naik.
- 2 if (head == NULL) return;; Jika list kosong, keluar dari fungsi.
- 3 int swapped;; Variabel untuk melacak apakah ada pertukaran yang dilakukan.
- 4 tnode ptr1;; Pointer untuk iterasi melalui list.
- 5 do { ... } while (swapped);: Lakukan loop hingga tidak ada pertukaran yang dilakukan dalam satu pass melalui list.
- 6 if (ptr1->next != head && ptr1->data > ptr1->next->data): Jika data node saat ini lebih besar dari data node berikutnya, tukar node tersebut dan set swapped ke 1.

```

109 void cetak() {
110     if (head == NULL) return;
111
112     tnode temp = head;
113     do {
114         printf("Address: %p, Data: %d\n", (void*)temp, temp->data);
115         temp = temp->next;
116     } while (temp != head);
117     printf("\n");

```

- 1 void cetak(): Fungsi untuk mencetak data dari linked list.
- 2 if (head == NULL) return;; Jika list kosong, keluar dari fungsi.
- 3 tnode temp = head;; Pointer untuk iterasi melalui list.
- 4 do { ... } while (temp != head);: Iterasi melalui seluruh list dan cetak alamat dan data dari setiap node.

```
> TIMELINE
120 int main() {
121     int jumlah_data, i, nilai;
122
123     printf("Masukkan jumlah data (1-10): ");
124     scanf("%d", &jumlah_data);
125
126     if (jumlah_data < 1 || jumlah_data > 10) {
127         printf("Jumlah data harus antara 1 dan 10.\n");
128         return 1;
129     }
130
131     for (i = 0; i < jumlah_data; i++) {
132         printf("Masukkan data ke-%d: ", i + 1);
133         scanf("%d", &nilai);
134         insert_last(nilai);
135     }
136
137     printf("Data sebelum sorting:\n");
138     cetak();
139
140     sort_ascending();
141
142     printf("Data setelah sorting:\n");
143     cetak();
144
145     return 0;
146 }
```

- 1 int main(): Fungsi utama program.
- 2 int jumlah\_data, i, nilai;: Variabel untuk menyimpan jumlah data, indeks loop, dan nilai data.
- 3 printf("Masukkan jumlah data (1-10): ");: Minta pengguna untuk memasukkan jumlah data.
- 4 scanf("%d", &jumlah\_data);: Baca jumlah data dari input pengguna.
- 5 if (jumlah\_data < 1 || jumlah\_data > 10): Jika jumlah data tidak antara 1 dan 10, cetak pesan error dan keluar dari program.
- 6 for (i = 0; i < jumlah\_data; i++): Loop untuk membaca nilai data dan menambahkannya ke list.
- 7 insert\_last(nilai);: Tambahkan nilai ke akhir list.
- 8 printf("Data sebelum sorting:\n");: Cetak data sebelum sorting.
- 9 cetak();: Panggil fungsi cetak untuk mencetak data.
- 10 sort\_ascending();: Urutkan data dalam urutan naik.
- 11 printf("Data setelah sorting:\n");: Cetak data setelah sorting.
- 12 return 0;: Kembalikan nilai 0, menandakan program selesai dengan sukses.