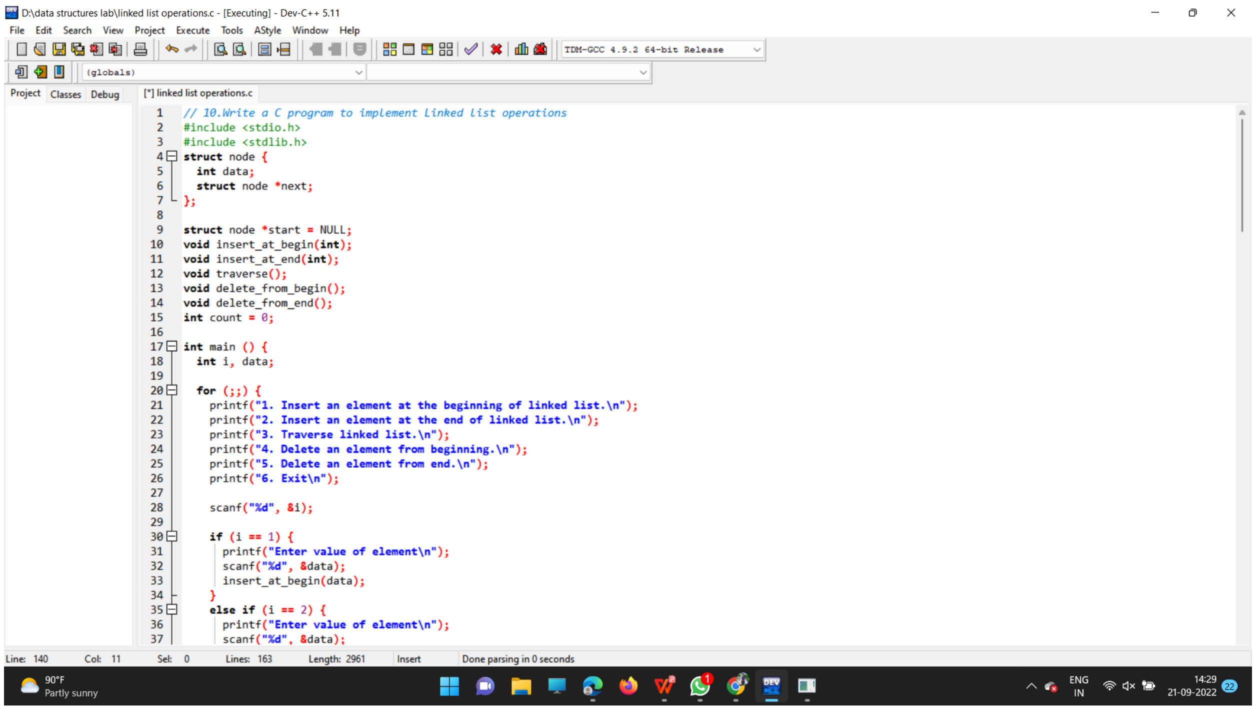
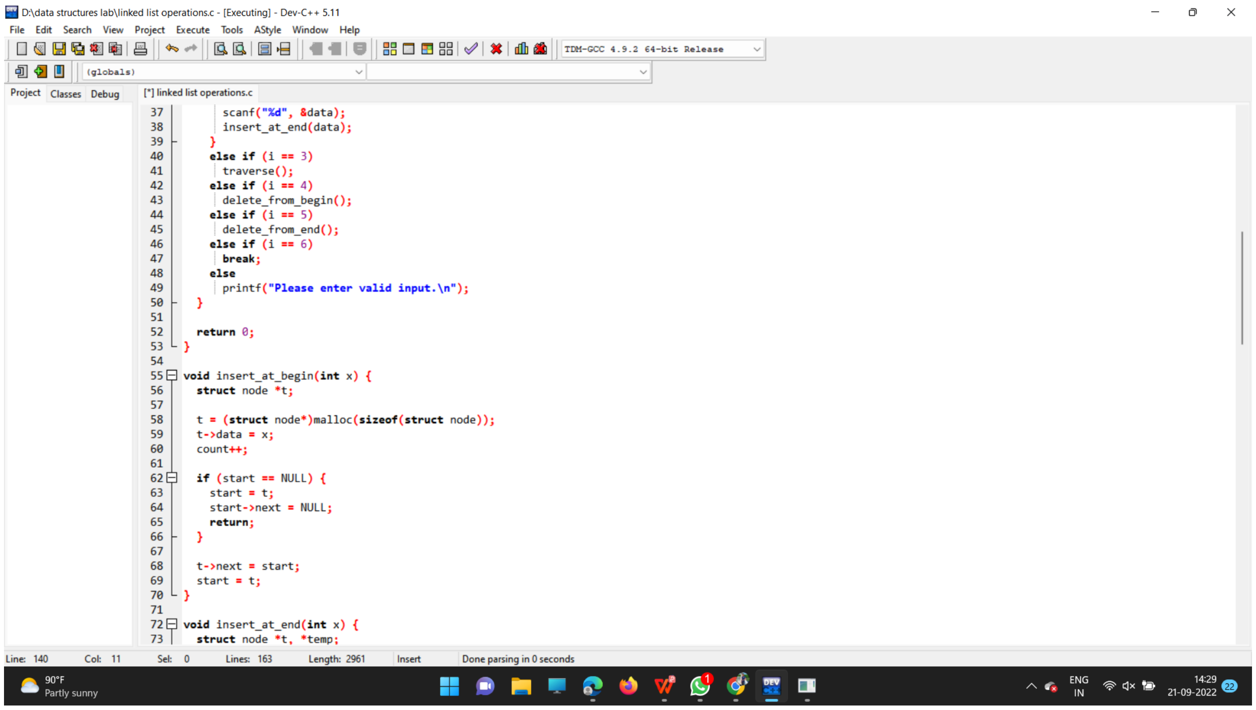
```
// 10.Write a C program to implement Linked list operations
#include <stdio.h>
#include <stdlib.h>
struct node {
  int data;
  struct node *next;
};

struct node *start = NULL;
void insert_at_begin(int);
void insert_at_end(int);
void traverse();
void delete_from_begin();
void delete_from_end();
int count = 0;

int main () {
  int i, data;

  for (;;) {
    printf("1. Insert an element at the beginning of linked list.\n");
    printf("2. Insert an element at the end of linked list.\n");
    printf("3. Traverse linked list.\n");
    printf("4. Delete an element from beginning.\n");
    printf("5. Delete an element from end.\n");
    printf("6. Exit\n");

    scanf("%d", &i);

    if (i == 1) {
      printf("Enter value of element\n");
      scanf("%d", &data);
      insert_at_begin(data);
    }
    else if (i == 2) {
      printf("Enter value of element\n");
      scanf("%d", &data);
      insert_at_end(data);
    }
    else if (i == 3)
      traverse();
    else if (i == 4)
      delete_from_begin();
    else if (i == 5)
```

```
      delete_from_end();
    else if (i == 6)
      break;
    else
      printf("Please enter valid input.\n");
  }

  return 0;
}

void insert_at_begin(int x) {
  struct node *t;

  t = (struct node*)malloc(sizeof(struct node));
  t->data = x;
  count++;

  if (start == NULL) {
    start = t;
    start->next = NULL;
    return;
  }

  t->next = start;
  start = t;
}

void insert_at_end(int x) {
  struct node *t, *temp;

  t = (struct node*)malloc(sizeof(struct node));
  t->data = x;
  count++;

  if (start == NULL) {
    start = t;
    start->next = NULL;
    return;
  }

  temp = start;

  while (temp->next != NULL)
    temp = temp->next;
  temp->next = t;
  t->next   = NULL;
}

void traverse() {
  struct node *t;

  t = start;

  if (t == NULL) {
    printf("Linked list is empty.\n");
    return;
  }

  printf("There are %d elements in linked list.\n", count);

  while (t->next != NULL) {
    printf("%d\n", t->data);
    t = t->next;
  }
  printf("%d\n", t->data); // Print last node
}

void delete_from_begin() {
  struct node *t;
  int n;

  if (start == NULL) {
    printf("Linked list is empty.\n");
    return;
  }

  n = start->data;
  t = start->next;
  free(start);
  start = t;
  count--;

  printf("%d deleted from the beginning successfully.\n", n);
}

void delete_from_end() {
  struct node *t, *u;
  int n;

  if (start == NULL) {
    printf("Linked list is empty.\n");
    return;
  }

  count--;

  if (start->next == NULL) {
    n = start->data;
    free(start);
    start = NULL;
    printf("%d deleted from end successfully.\n", n);
    return;
  }

  t = start;

  while (t->next != NULL) {
    u = t;
    t = t->next;
  }

  n = t->data;
  u->next = NULL;
  free(t);

  printf("%d deleted from end successfully.\n", n);
}
```


D:\data structures lab\linked list operations.c - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
[*] linked list operations.c
90°F
Partly sunny
ENG
IN
14:29
21-09-2022

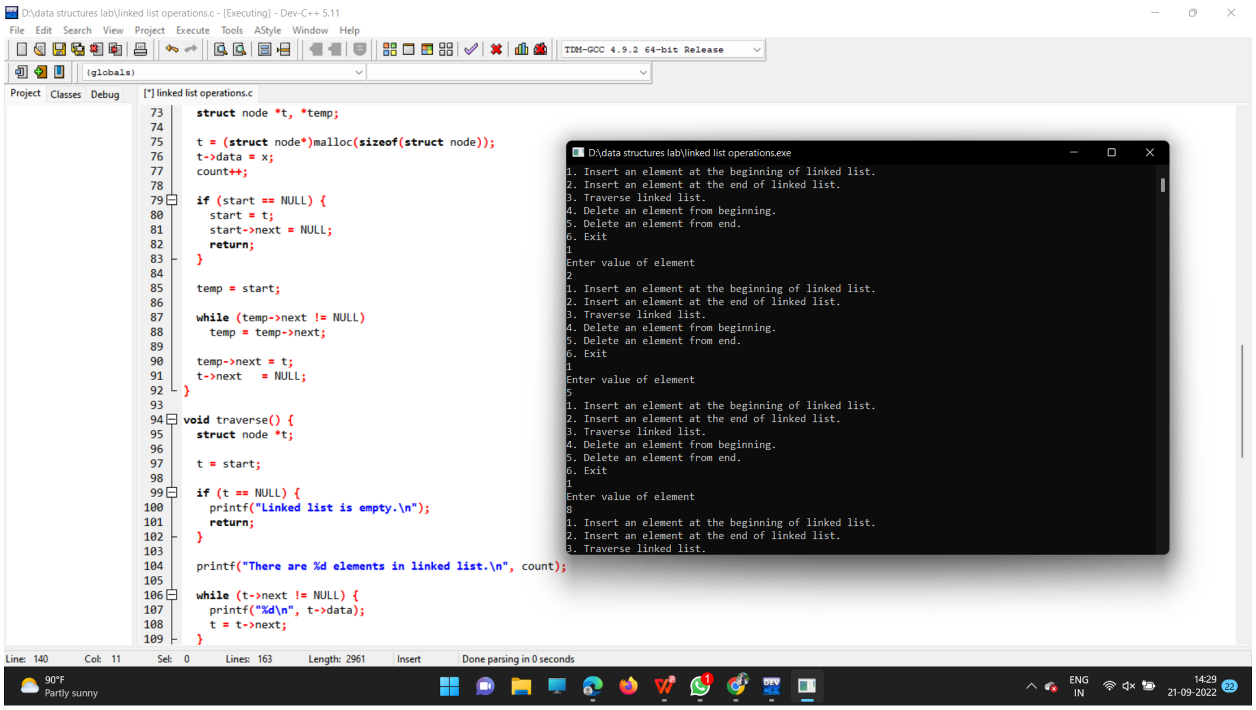
```c
// 10.Write a C program to implement Linked List operations
#include <stdio.h>
#include <stdlib.h>
struct node {
  int data;
  struct node *next;
};

struct node *start = NULL;
void insert_at_begin(int);
void insert_at_end(int);
void traverse();
void delete_from_begin();
void delete_from_end();
int count = 0;

int main () {
  int i, data;

  for (;;) {
    printf("1. Insert an element at the beginning of linked list.\n");
    printf("2. Insert an element at the end of linked list.\n");
    printf("3. Traverse linked list.\n");
    printf("4. Delete an element from beginning.\n");
    printf("5. Delete an element from end.\n");
    printf("6. Exit\n");

    scanf("%d", &i);

    if (i == 1) {
      printf("Enter value of element\n");
      scanf("%d", &data);
      insert_at_begin(data);
    }
    else if (i == 2) {
      printf("Enter value of element\n");
      scanf("%d", &data);
```

Line: 140 Col: 11 Sel: 0 Lines: 163 Length: 2961 Insert Done parsing in 0 seconds


D:\data structures lab\linked list operations.c - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
[*] linked list operations.c
90°F
Partly sunny
ENG
IN
14:29
21-09-2022

```c
      scanf("%d", &data);
      insert_at_end(data);
    }
    else if (i == 3)
      traverse();
    else if (i == 4)
      delete_from_begin();
    else if (i == 5)
      delete_from_end();
    else if (i == 6)
      break;
    else
      printf("Please enter valid input.\n");
  }

  return 0;
}

void insert_at_begin(int x) {
  struct node *t;

  t = (struct node*)malloc(sizeof(struct node));
  t->data = x;
  count++;

  if (start == NULL) {
    start = t;
    start->next = NULL;
    return;
  }

  t->next = start;
  start = t;
}

void insert_at_end(int x) {
  struct node *t, *temp;
```

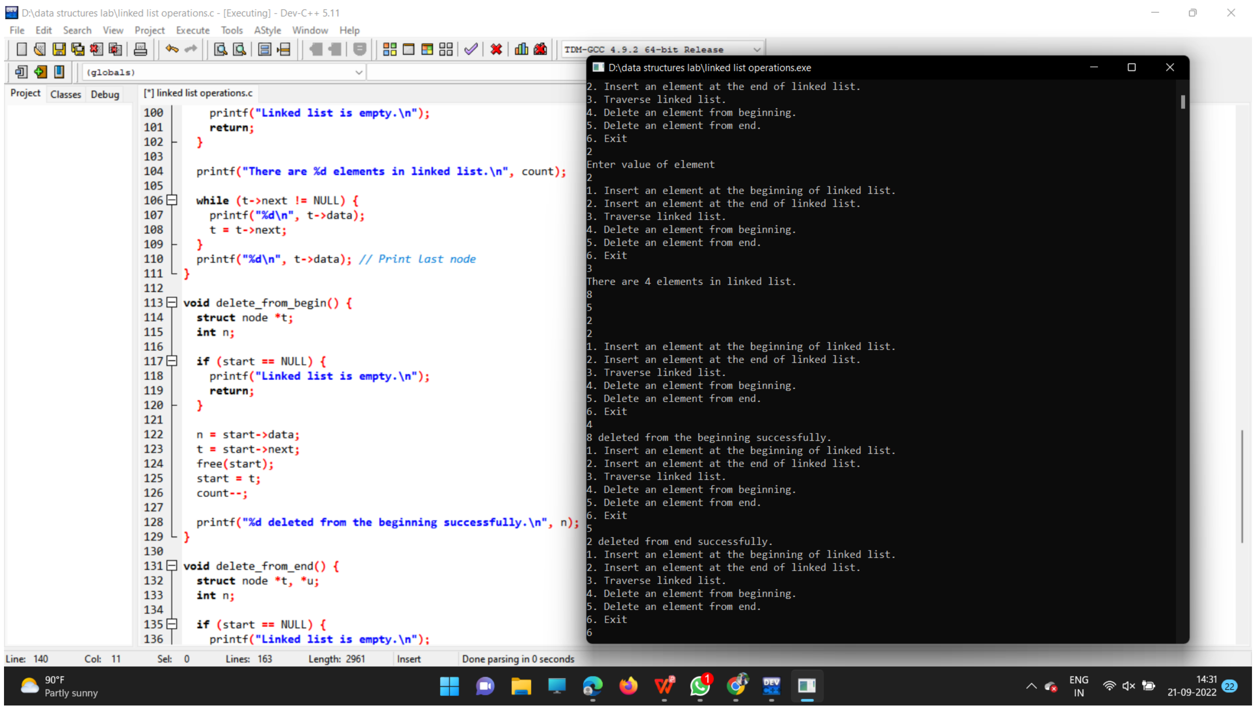
Line: 140 Col: 11 Sel: 0 Lines: 163 Length: 2961 Insert Done parsing in 0 seconds


D:\data structures lab\linked list operations.c - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
[*] linked list operations.c
Line: 140 Col: 11 Sel: 0 Lines: 163 Length: 2961 Insert Done parsing in 0 seconds
90°F
Partly sunny
ENG
IN
14:29
21-09-2022

```c
  struct node *t, *temp;

  t = (struct node*)malloc(sizeof(struct node));
  t->data = x;
  count++;

  if (start == NULL) {
    start = t;
    start->next = NULL;
    return;
  }

  temp = start;

  while (temp->next != NULL)
    temp = temp->next;

  temp->next = t;
  t->next   = NULL;
}

void traverse() {
  struct node *t;

  t = start;

  if (t == NULL) {
    printf("Linked list is empty.\n");
    return;
  }

  printf("There are %d elements in linked list.\n", count);

  while (t->next != NULL) {
    printf("%d\n", t->data);
    t = t->next;
  }
```

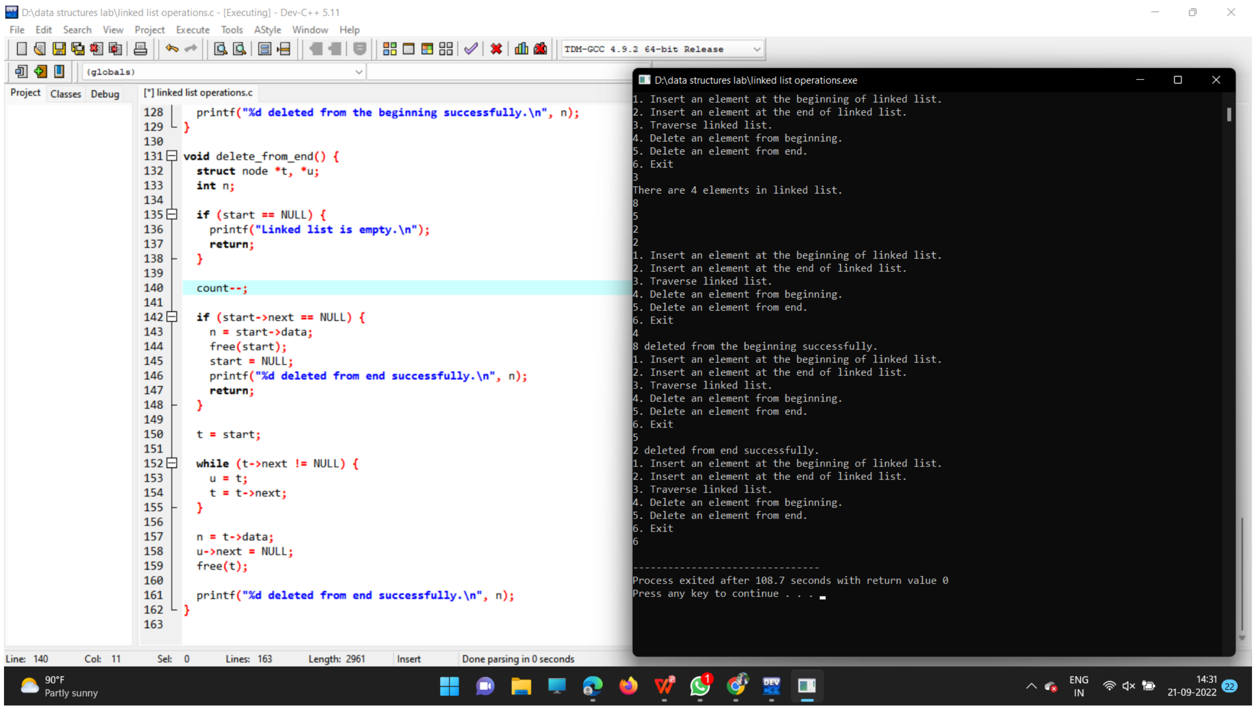
D:\data structures lab\linked list operations.exe

```
1. Insert an element at the beginning of linked list.
2. Insert an element at the end of linked list.
3. Traverse linked list.
4. Delete an element from beginning.
5. Delete an element from end.
6. Exit
1
Enter value of element
2
1. Insert an element at the beginning of linked list.
2. Insert an element at the end of linked list.
3. Traverse linked list.
4. Delete an element from beginning.
5. Delete an element from end.
6. Exit
1
Enter value of element
5
1. Insert an element at the beginning of linked list.
2. Insert an element at the end of linked list.
3. Traverse linked list.
4. Delete an element from beginning.
5. Delete an element from end.
6. Exit
1
Enter value of element
8
1. Insert an element at the beginning of linked list.
2. Insert an element at the end of linked list.
3. Traverse linked list.
```



```c
    printf("Linked list is empty.\n");
    return;
  }

  printf("There are %d elements in linked list.\n", count);

  while (t->next != NULL) {
    printf("%d\n", t->data);
    t = t->next;
  }
  printf("%d\n", t->data); // Print last node
}

void delete_from_begin() {
  struct node *t;
  int n;

  if (start == NULL) {
    printf("Linked list is empty.\n");
    return;
  }

  n = start->data;
  t = start->next;
  free(start);
  start = t;
  count--;

  printf("%d deleted from the beginning successfully.\n", n);
}

void delete_from_end() {
  struct node *t, *u;
  int n;

  if (start == NULL) {
    printf("Linked list is empty.\n");
```

```
2. Insert an element at the end of linked list.
3. Traverse linked list.
4. Delete an element from beginning.
5. Delete an element from end.
6. Exit
2
Enter value of element
2
1. Insert an element at the beginning of linked list.
2. Insert an element at the end of linked list.
3. Traverse linked list.
4. Delete an element from beginning.
5. Delete an element from end.
6. Exit
3
There are 4 elements in linked list.
8
5
2
2
1. Insert an element at the beginning of linked list.
2. Insert an element at the end of linked list.
3. Traverse linked list.
4. Delete an element from beginning.
5. Delete an element from end.
6. Exit
4
8 deleted from the beginning successfully.
1. Insert an element at the beginning of linked list.
2. Insert an element at the end of linked list.
3. Traverse linked list.
4. Delete an element from beginning.
5. Delete an element from end.
6. Exit
5
2 deleted from end successfully.
1. Insert an element at the beginning of linked list.
2. Insert an element at the end of linked list.
3. Traverse linked list.
4. Delete an element from beginning.
5. Delete an element from end.
6. Exit
6
```



```c
  printf("%d deleted from the beginning successfully.\n", n);
}

void delete_from_end() {
  struct node *t, *u;
  int n;

  if (start == NULL) {
    printf("Linked list is empty.\n");
    return;
  }

  count--;

  if (start->next == NULL) {
    n = start->data;
    free(start);
    start = NULL;
    printf("%d deleted from end successfully.\n", n);
    return;
  }

  t = start;

  while (t->next != NULL) {
    u = t;
    t = t->next;
  }

  n = t->data;
  u->next = NULL;
  free(t);

  printf("%d deleted from end successfully.\n", n);
}
```

```
1. Insert an element at the beginning of linked list.
2. Insert an element at the end of linked list.
3. Traverse linked list.
4. Delete an element from beginning.
5. Delete an element from end.
6. Exit
3
There are 4 elements in linked list.
8
5
2
2
1. Insert an element at the beginning of linked list.
2. Insert an element at the end of linked list.
3. Traverse linked list.
4. Delete an element from beginning.
5. Delete an element from end.
6. Exit
4
8 deleted from the beginning successfully.
1. Insert an element at the beginning of linked list.
2. Insert an element at the end of linked list.
3. Traverse linked list.
4. Delete an element from beginning.
5. Delete an element from end.
6. Exit
5
2 deleted from end successfully.
1. Insert an element at the beginning of linked list.
2. Insert an element at the end of linked list.
3. Traverse linked list.
4. Delete an element from beginning.
5. Delete an element from end.
6. Exit
6

--------------------------------
Process exited after 108.7 seconds with return value 0
Press any key to continue . . .
```