Algoritma & Struktur Data

M4 DLL Insert

Dosen Pengampu

Dr. Tita Karlita S.Kom, M.Kom

Disusun Oleh :

Nama : M. Faza Nur Husain

Nrp : 3121550004

D3 PJJ AK TEKNIK INFORMATIKA

POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

TAHUN AKADEMIK 2021/2022

Source Code dll insert awal, akhir, after dan before

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

/* Node Stucture */
typedef struct node_t {
  int data;
  struct node_t *next;
} Node;

/* Function Declarations */
Node * insert_top(int, Node *);
Node * insert_bottom(int, Node *);
Node * insert_after(int, int, Node *);
Node * insert_before(int, int, Node *);
void print(Node *);
int count(Node *);

/* Add a new node to the top of a list */
Node * insert_top(int num, Node *head) {
 Node *new_node;
 new_node = (Node *) malloc(sizeof(Node));
 new_node->data = num;
 new_node->next= head;
 head = new_node;
return head;
}


/* Add a new node to the bottom of a list */
Node * insert_bottom(int num, Node *head) {
  Node *current_node = head;
  Node *new_node;
 while ( current_node != NULL && current_node->next != NULL) {
   current_node = current_node->next;
  }

  new_node = (Node *) malloc(sizeof(Node));
  new_node->data = num;
  new_node->next= NULL;
  if (current_node != NULL)
    current_node->next = new_node;
  else
     head = new_node;
return head;
}


/* Add a new node after an element in the list */
Node * insert_after(int num, int prev_num, Node *head) {
  Node *current_node = head;
```

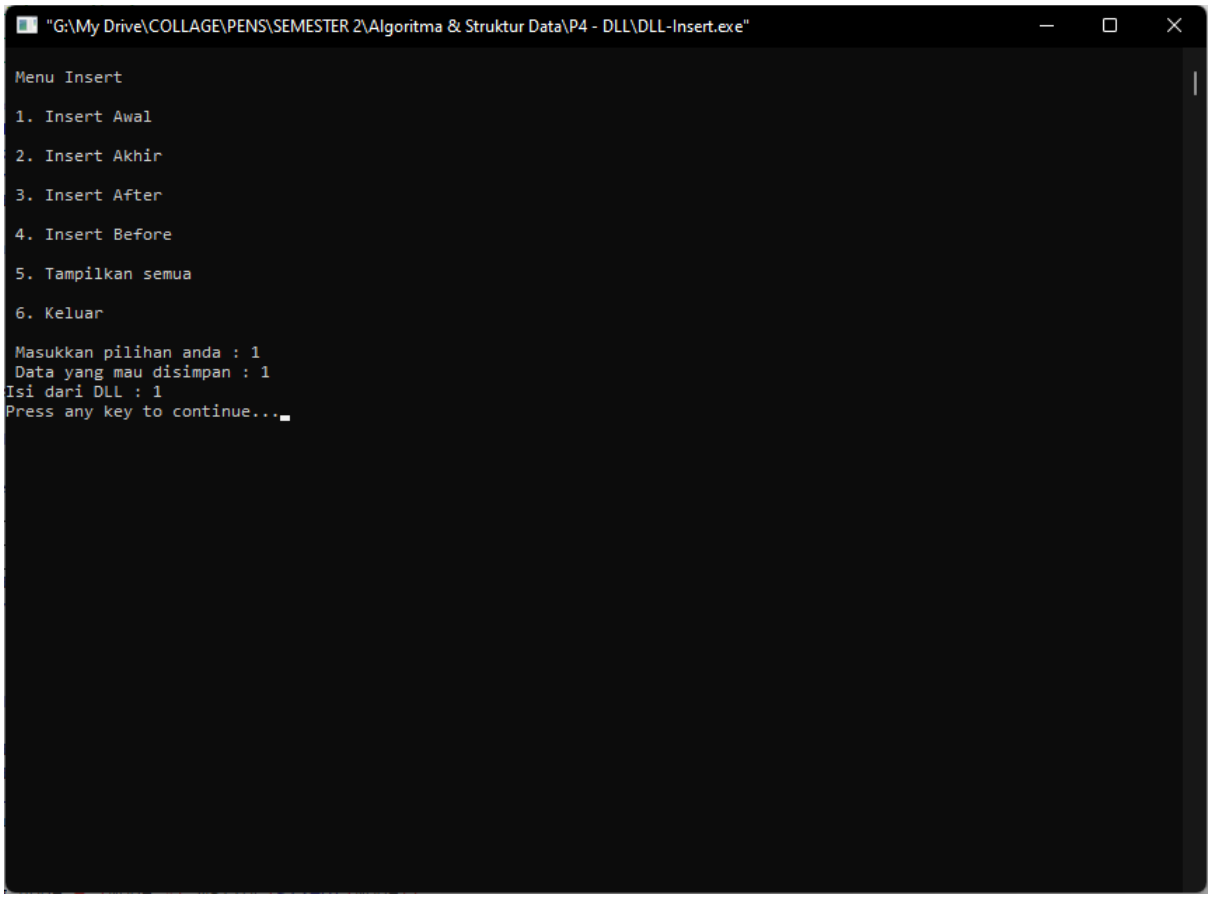
```
  Node *new_node;
  while ( current_node->data != prev_num) {
      current_node = current_node->next;
  }
  new_node = (Node *) malloc(sizeof(Node));
  new_node->data = num;
  new_node->next= current_node->next;
  current_node->next = new_node;
return head;
}


/* Add a new node before an element in the list */
  Node * insert_before(int num, int next_num, Node *head) {
  Node *current_node = head;
  Node *new_node;
  while ( current_node->next->data != next_num) {
    current_node = current_node->next;
  }
  new_node = (Node *) malloc(sizeof(Node));
  new_node->data = num;
  new_node->next= current_node->next;
  current_node->next = new_node;
return head;
}

/* Print all the elements in the linked list */
void print(Node *head) {
  Node *current_node = head;
  while ( current_node != NULL) {
    printf("%d ", current_node->data);
    current_node = current_node->next;
  }
}

/* Program main */
int main()
{
   Node *head = NULL;
   int num, prev_num, next_num;
   int option;
   char * temp;
   char ch;
   /* Display Menu */
   while(1) {

     printf("\n Menu Insert \n");
     printf("\n 1. Insert Awal\n");
     printf("\n 2. Insert Akhir\n");
     printf("\n 3. Insert After\n");
     printf("\n 4. Insert Before\n");
     printf("\n 5. Tampilkan semua\n");
     printf("\n 6. Keluar \n");
```

```
     printf("\n Masukkan pilihan anda : ");
     if (scanf("%d", &option) != 1) {
        printf(" *Error: Input Salah,silahkan coba lagi.\n");
        scanf("%s", &temp); /*clear input buffer */
        continue;
     }

     switch (option) {
      case 1:        /* Add to top*/
          printf(" Data yang mau disimpan : ");
          if (scanf("%d", &num) != 1) {
              printf(" *Error: Input Salah,silahkan coba lagi.\n");
              scanf("%s", &temp);   /*clear input buffer */
              continue;
          }
          head = insert_top(num, head);
          printf("Isi dari DLL : %d", num);
          printf("\nPress any key to continue...");
          getch();
          break;

      case 2:    /* add to bottom */
          printf(" Data yang mau disimpan : ");
          if (scanf("%d", &num) != 1) {
              printf(" *Error: Input Salah,silahkan coba lagi. \n");
              scanf("%s", &temp);
              continue;
          }
          head = insert_bottom(num, head);
          printf("Berhasil menambahkan %d di akhir", num);
          printf("\nPress any key to continue...");
          getch();
          break;

      case 3:    /* Insert After */
          printf(" Data yang mau disimpan : ");
          if (scanf("%d", &num) != 1) {
              printf(" *Error: Input Salah,silahkan coba lagi.\n");
              scanf("%s", &temp);
              continue;
          }

          printf(" Setelah nomor mana yang ingin Anda masukkan : ");
          if (scanf("%d", &prev_num) != 1) {
              printf(" *Error: Input Salah,silahkan coba lagi.\n");
              scanf("%s", &temp);
              continue;
          }
          if (head != NULL) {
              head = insert_after(num, prev_num, head);
              printf("%d berhasil ditambahkan setelah %d", num,
prev_num);
          }else {
```

```
            printf("The list is empty", num, prev_num);
        }
            printf("\nTekan apa aja untuk melanjutkan...");
            getch();
            break;

    case 4:    /* Insert Before */
         printf(" Data yang mau disimpan : ");
         if (scanf("%d", &num) != 1) {
            printf(" *Error: Input Salah,silahkan coba lagi. \n");
            scanf("%s", &temp);
            continue;
        }

        printf(" Before which number do you want to insert : ");
        if (scanf("%d", &prev_num) != 1) {
            printf(" *Error: Input Salah,silahkan coba lagi.\n");
            scanf("%s", &temp);
            continue;
        }

      if (head != NULL) {
           head = insert_before(num, prev_num, head);
           printf("Number %d inserted before %d", num, prev_num);
       }else {
           printf("The list is empty", num, prev_num);
       }
           printf("\nTekan apa saja untuk melanjutkan...");
          getch();
          break;

    case 5: /* Show all elements */
        printf("\nElements in the list: \n [ ");
        print(head);
        printf("]\n\nTekan apa saja untuk melanjutkan...");
        getch();
        break;

    case 6:  /* Exit */
        return(0);
        break;

    default:
        printf("Pilihan salah, silahkan coba lagi.");
        getch();

    } /* End of Switch */
  } /* End of While */

return(0);
}
```

Insert Awal

```
Node * insert_top(int num, Node *head) {
 Node *new_node;
 new_node = (Node *) malloc(sizeof(Node));
 new_node->data = num;
 new_node->next= head;
 head = new_node;
return head;
}
```

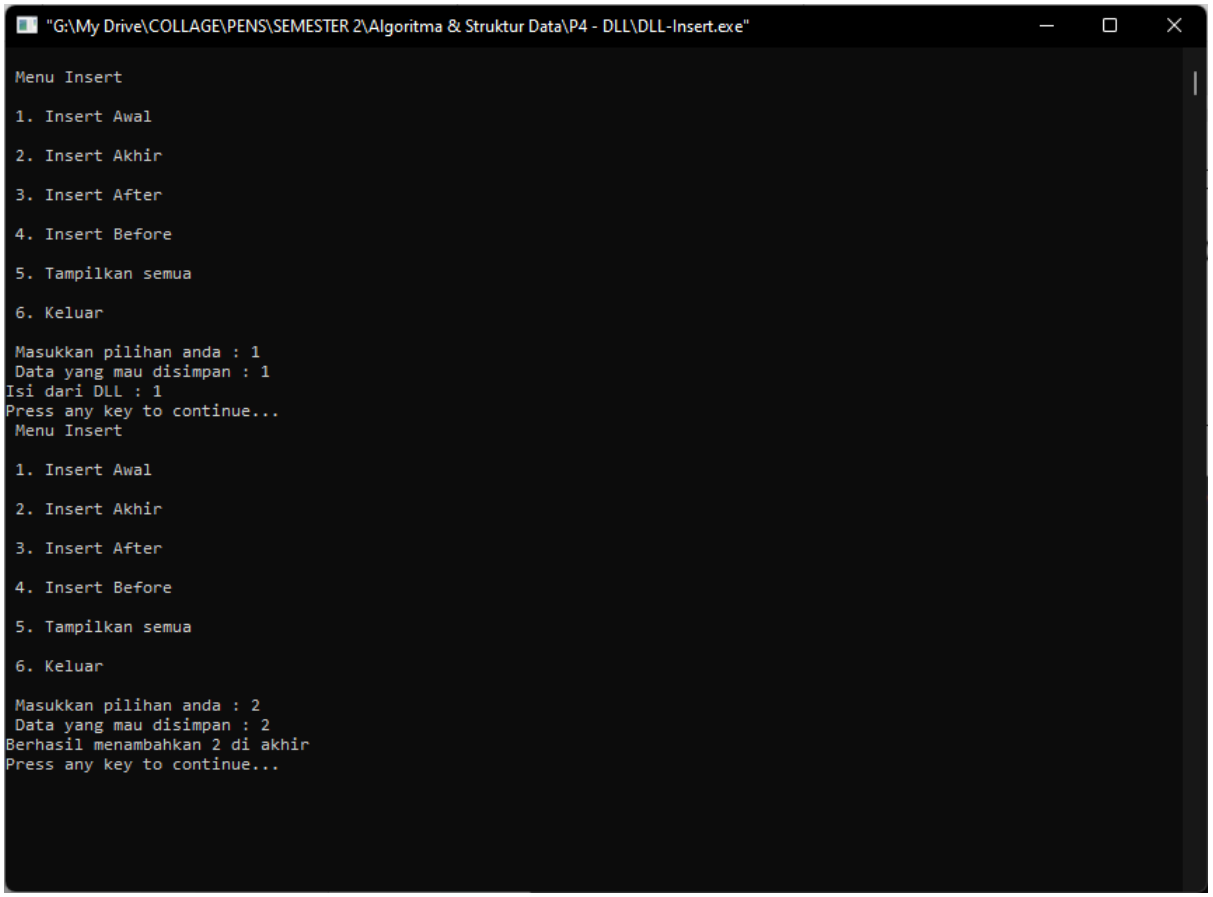


Insert Akhir

Source Code :

```
Node * insert_bottom(int num, Node *head) {
  Node *current_node = head;
  Node *new_node;
 while ( current_node != NULL && current_node->next != NULL) {
   current_node = current_node->next;
  }

  new_node = (Node *) malloc(sizeof(Node));
  new_node->data = num;
  new_node->next= NULL;
  if (current_node != NULL)
    current_node->next = new_node;
  else
     head = new_node;
return head;
}
```

Output:

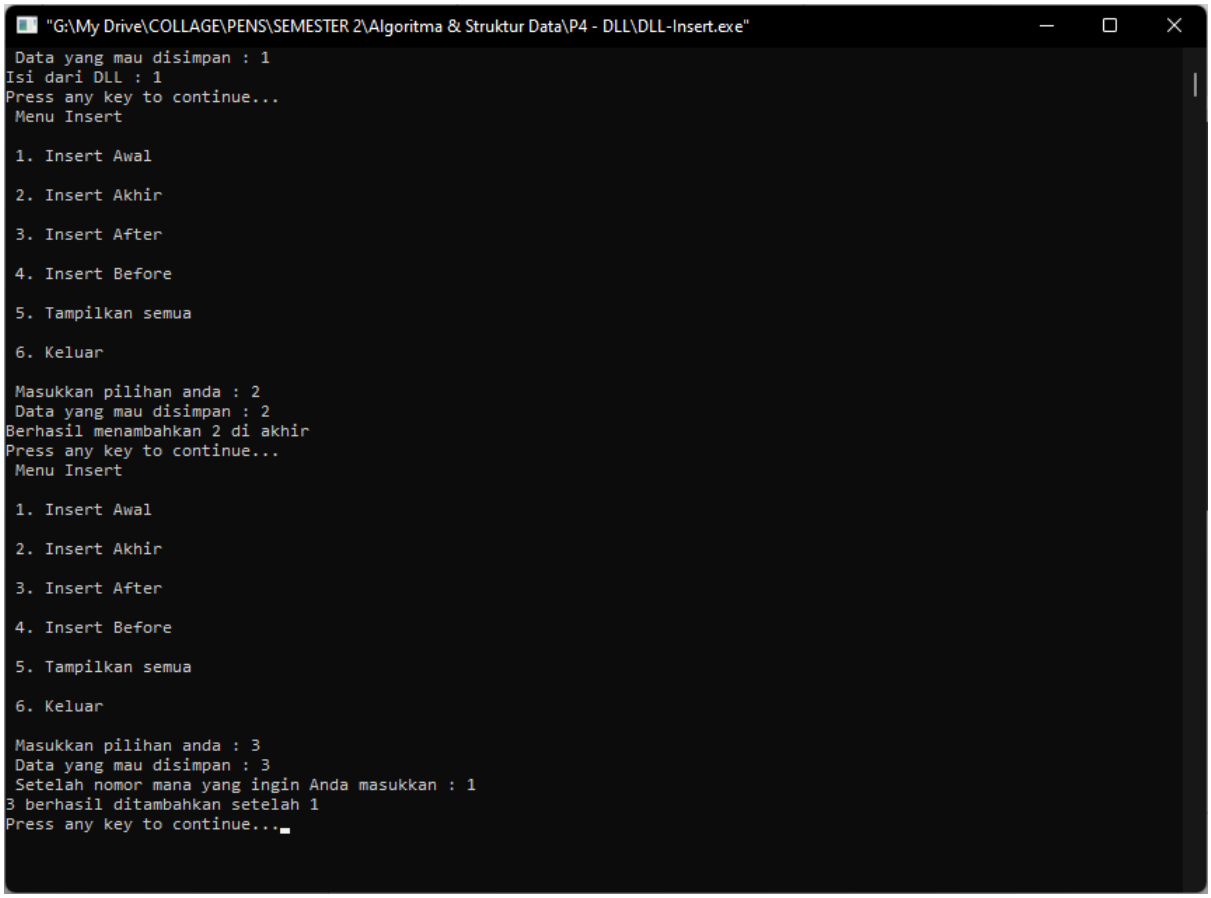


Insert After

Source code :

```
Node * insert_after(int num, int prev_num, Node *head) {
  Node *current_node = head;
  Node *new_node;
  while ( current_node->data != prev_num) {
      current_node = current_node->next;
  }
  new_node = (Node *) malloc(sizeof(Node));
  new_node->data = num;
  new_node->next= current_node->next;
  current_node->next = new_node;
return head;
}
```

Output :

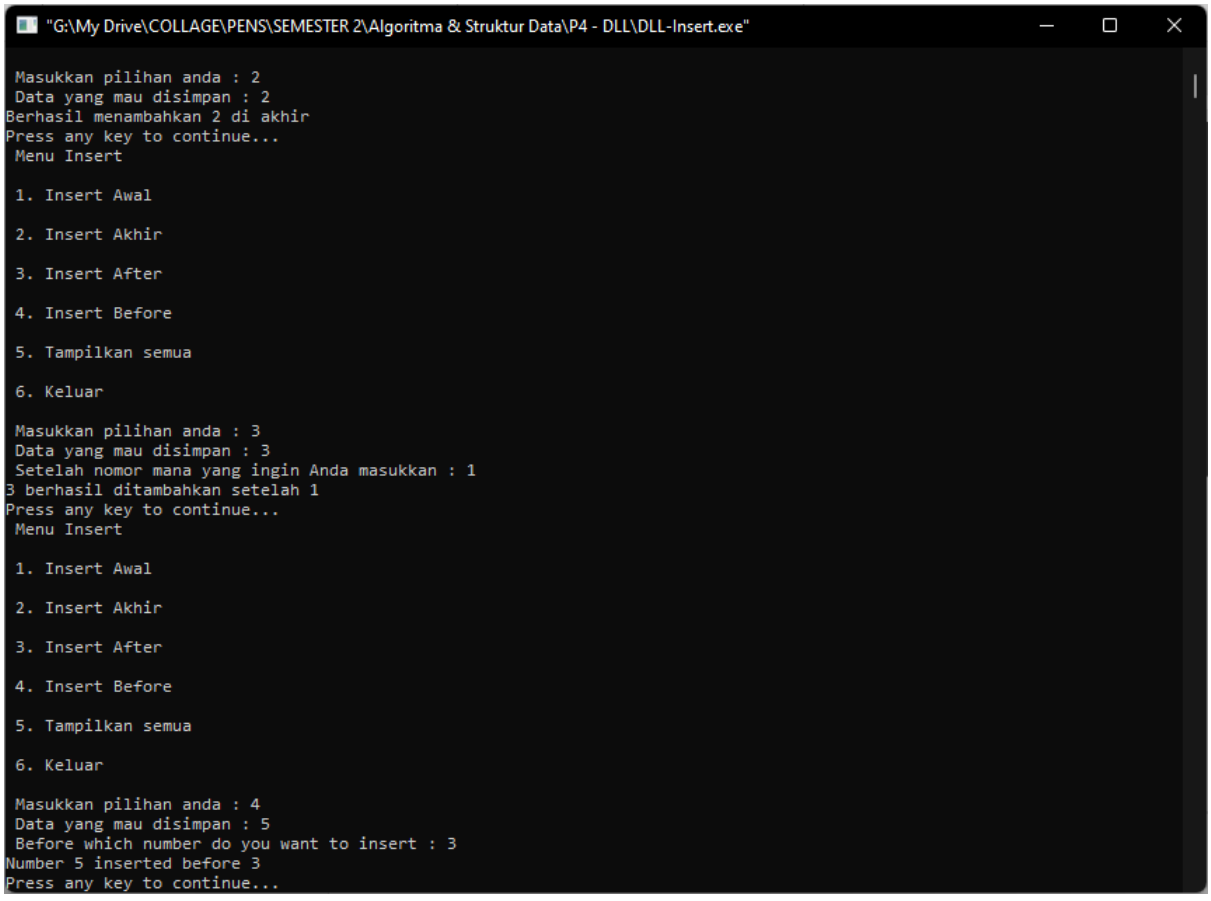


Insert Before

Source code :

```
Node * insert_before(int num, int next_num, Node *head) {
  Node *current_node = head;
  Node *new_node;
  while ( current_node->next->data != next_num) {
    current_node = current_node->next;
  }
  new_node = (Node *) malloc(sizeof(Node));
  new_node->data = num;
  new_node->next= current_node->next;
  current_node->next = new_node;
return head;
}
```

Output :



Data yang berhasil disimpan :