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**

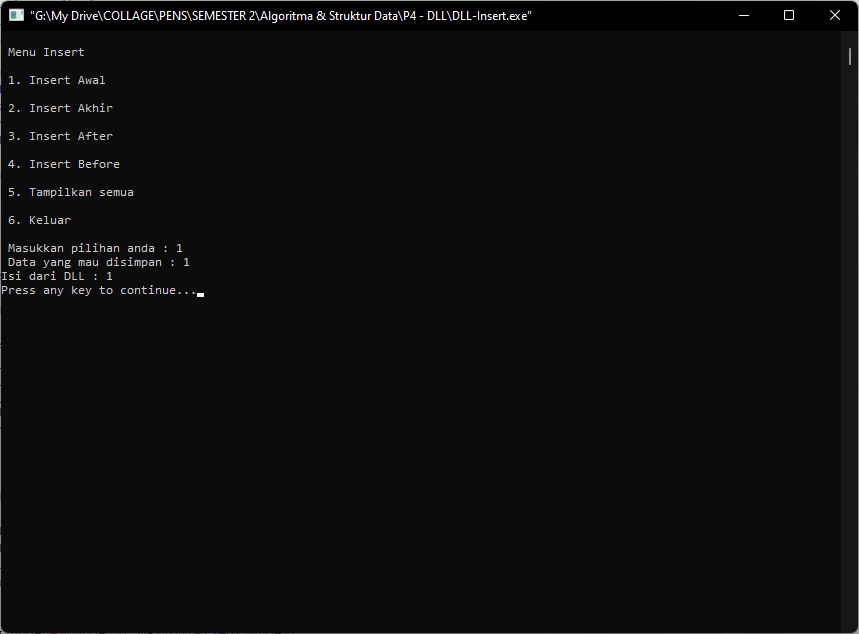
**TAHUN AKADEMIK 2021/2022**

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;  } |

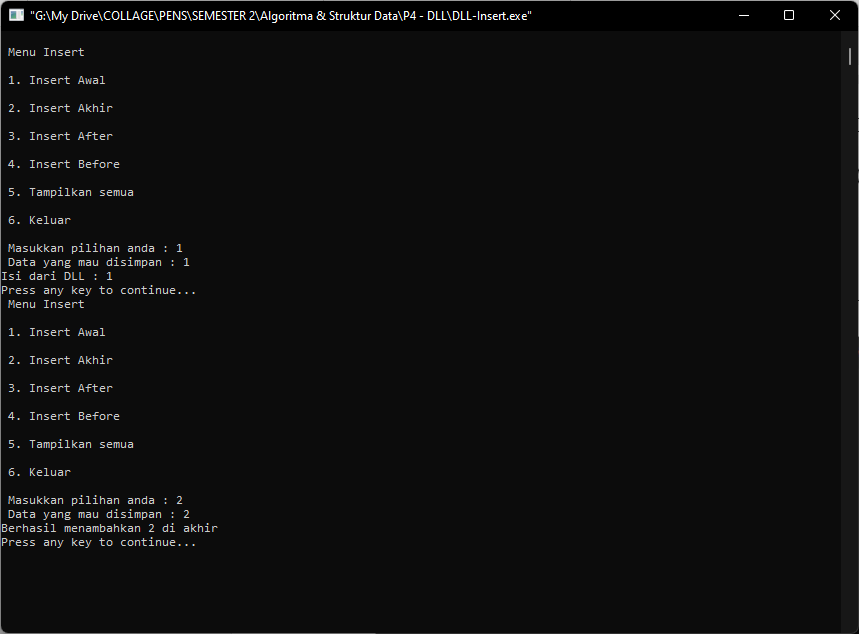


Insert Akhir

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:

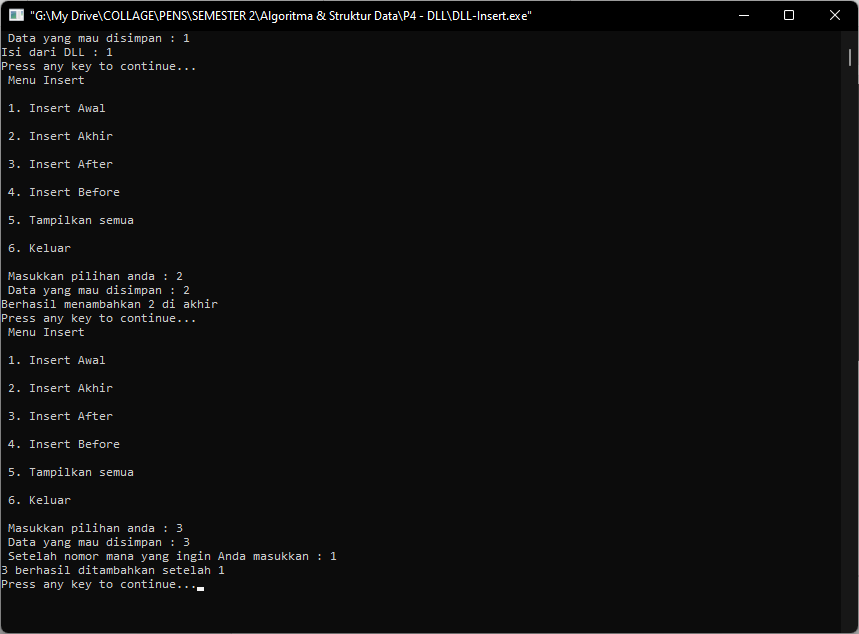


Insert After

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 :

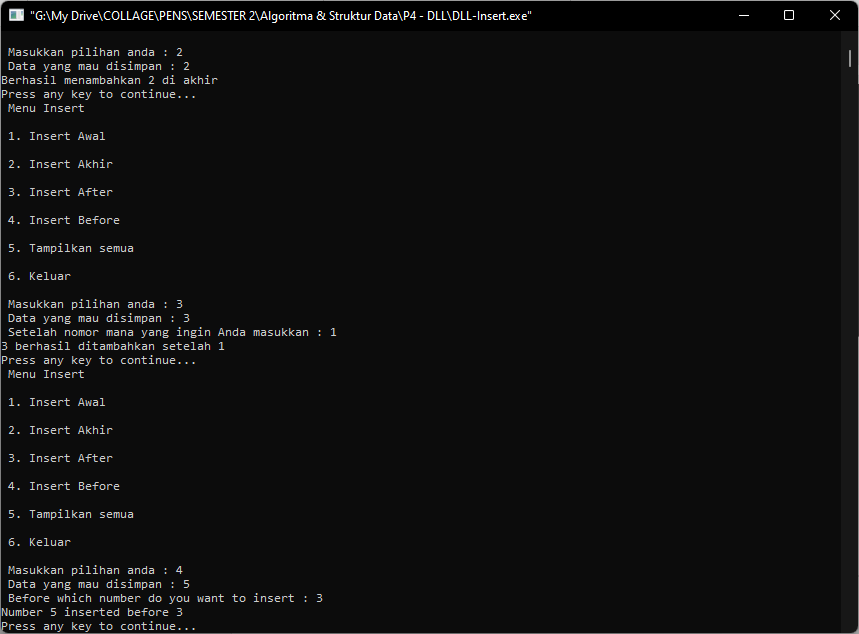


Insert Before

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 :

