**Singly linked list - insert, remove, add, count source code**

This snippet submitted by Girish Amara on 2010-08-25. It has been viewed 452459 times.   
Rating of 7.2 with 1529 votes

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206 | #include<stdio.h>  #include<stdlib.h>    struct node  {      int data;      struct node \*next;  }\*head;        void append(int num)  {      struct node \*temp,\*right;      temp= (struct node \*)malloc(sizeof(struct node));      temp->data=num;      right=(struct node \*)head;      while(right->next != NULL)      right=right->next;      right->next =temp;      right=temp;      right->next=NULL;  }        void add( int num )  {      struct node \*temp;      temp=(struct node \*)malloc(sizeof(struct node));      temp->data=num;      if (head== NULL)      {      head=temp;      head->next=NULL;      }      else      {      temp->next=head;      head=temp;      }  }  void addafter(int num, int loc)  {      int i;      struct node \*temp,\*left,\*right;      right=head;      for(i=1;i<loc;i++)      {      left=right;      right=right->next;      }      temp=(struct node \*)malloc(sizeof(struct node));      temp->data=num;      left->next=temp;      left=temp;      left->next=right;      return;  }        void insert(int num)  {      int c=0;      struct node \*temp;      temp=head;      if(temp==NULL)      {      add(num);      }      else      {      while(temp!=NULL)      {          if(temp->data<num)          c++;          temp=temp->next;      }      if(c==0)          add(num);      else if(c<count())          addafter(num,++c);      else          append(num);      }  }        int delete(int num)  {      struct node \*temp, \*prev;      temp=head;      while(temp!=NULL)      {      if(temp->data==num)      {          if(temp==head)          {          head=temp->next;          free(temp);          return 1;          }          else          {          prev->next=temp->next;          free(temp);          return 1;          }      }      else      {          prev=temp;          temp= temp->next;      }      }      return 0;  }      void  display(struct node \*r)  {      r=head;      if(r==NULL)      {      return;      }      while(r!=NULL)      {      printf("%d ",r->data);      r=r->next;      }      printf("\n");  }      int count()  {      struct node \*n;      int c=0;      n=head;      while(n!=NULL)      {      n=n->next;      c++;      }      return c;  }      int  main()  {      int i,num;      struct node \*n;      head=NULL;      while(1)      {      printf("\nList Operations\n");      printf("===============\n");      printf("1.Insert\n");      printf("2.Display\n");      printf("3.Size\n");      printf("4.Delete\n");      printf("5.Exit\n");      printf("Enter your choice : ");      if(scanf("%d",&i)<=0){          printf("Enter only an Integer\n");          exit(0);      } else {          switch(i)          {          case 1:      printf("Enter the number to insert : ");                   scanf("%d",&num);                   insert(num);                   break;          case 2:     if(head==NULL)                  {                  printf("List is Empty\n");                  }                  else                  {                  printf("Element(s) in the list are : ");                  }                  display(n);                  break;          case 3:     printf("Size of the list is %d\n",count());                  break;          case 4:     if(head==NULL)                  printf("List is Empty\n");                  else{                  printf("Enter the number to delete : ");                  scanf("%d",&num);                  if(delete(num))                      printf("%d deleted successfully\n",num);                  else                      printf("%d not found in the list\n",num);                  }                  break;          case 5:     return 0;          default:    printf("Invalid option\n");          }      }      }      return 0;  } |