Задача. Описать подпрограммы push_front() и push_back() для вставки элемента в начало и в конец списка. Описать функцию main(), использующую данные подпрограммы.

Решение

Способ 1. Опишем подпрограммы как функции, возвращающие void. Их аргументами будут адрес объекта, хранящего адрес начала списка, и вставляемый символ.

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Node * link;
typedef struct Node { char elem;
                      link next;
                    }
        node;
typedef link list;

void push_front(list *lp, char c)
/* вставить символ c в начало списка *lp */
{ link p=malloc(sizeof(node));
  p->elem=c;
  p->next=*lp;
  *lp=p;
}

void push_back(list *lp, char c)
/* вставить символ c в конец списка *lp */
{
  while (*lp!=NULL)
       lp=&(*lp)->next;
  *lp=malloc(sizeof(node));
  (*lp)->elem=c;
  (*lp)->next=NULL;
}

int main()
{ list lst=NULL;
  push_back(&lst,'a');
  push_front(&lst,'s');
  push_back(&lst,'t');
  link p=lst;
  while (p!=NULL){
      printf("%d",p->elem);
      p=p->next;
  }
  printf("\n");
/* напечатается sat */
  return 0;
}
```

Недостатком данного способа является использование двойной косвенной адресации. Например, в выражении `(*pp)->next`, находящемся в теле цикла, выполняются две операции разыменования указателя.

Способ 2 Можно обойтись без двойной косвенной адресации, если по-другому организовать подпрограммы: их аргументами будут адрес начала списка и вставляемый элемент, а возвращаемым значением будет адрес начала измененного списка.

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Node * link;
typedef struct Node { char elem;
                      link next;
                    }
        node;
typedef link list;

list push_front(list l, char c)
/* вставить символ c в начало списка *lp */
{ link p=malloc(sizeof(node));
  p->elem=c;
  p->next=l;
  return p;
}


list push_back(list l, char c)
/* вставить символ c в конец списка *lp */
{
  link q=l, p=malloc(sizeof(node));
  p->elem=c;
  p->next=NULL;;
  if (l==NULL)
      return p;
  while (q->next!=NULL)
      q=q->next;
  q->next=p;
  return l;
}

int main()
{ list lst=NULL;
  lst=push_back(lst,'a');
  lst=push_front(lst,'s');
  lst=push_back(lst,'t');
  link p=lst;
  while (p!=NULL){
      printf("%c",p->elem);
      p=p->next;
  }
  printf("\n");
/* напечатается sat */
  return 0;
}
```

Данный способ также более наглядно отражает изменения списка lst – они явно выражены присваиваниями объекту lst новых значений.