

**实验报告**

**学院名称： 信息产业学院**

**专业名称： 计算机科学与技术**

**课程名称： 数据结构**

**班 级： 211060104 学号：21106010413**

**学生姓名： 刘钟泽**

**指导老师： 高俊**

2023年5月5日

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 学生姓名 | 刘钟泽 | | 学号 | 21106010413 | 实验成绩 |  |
| 实验项目名称 | | 单链表的基本操作 | | | | |
| 实验地点 | | 博学楼C207 | | 实验日期 | 5月5日 | |
| 1. 实验目的和要求 2. **实验目的：** 3. 理解和掌握单链表的类型定义方法和结点生成方法。 4. 掌握建立单链表和显示单链表元素的算法。 5. 掌握单链表的插入和删除算法。 6. **实验要求：** 7. 能够熟练地掌握单链表的原理及其数据结构定义，并能够使用指针对单链表进行基本操作； 8. 能够运用C语言编写程序，实现单链表的创建、遍历、插入和删除等基本操作； 9. 能够对程序进行测试，检验程序的正确性和健壮性，及时发现和排除程序存在的问题； 10. 能够编写实验报告，简明、准确地总结实验流程和实验结果，归纳实验心得和感受。 11. 实验内容和原理 12. **实验内容：**   （1） 设计程序，创建一个单链表，并从屏幕显示单链表元素列表；  （2） 设计程序，删除链表某位置上的元素，并保持链表原有的顺序不变；  （3） 设计程序，在指定的位置插入一个元素。   1. 实验原理**：**   本实验主要涉及单链表的创建、遍历、插入和删除等操作。单链表是由若干个节点组成的链式结构，每个节点包含一个数据域和一个指向下一个节点的指针域。在单链表中，头节点是第一个节点，尾节点是最后一个节点，除尾节点外的每个节点的指针域都指向下一个节点，而尾节点的指针域则置为 NULL。 **实验过程**  1. 首先定义一个单链表的结构体，包含数据域和指针域； 2. 实现一个函数createList()，用于创建单链表，用户输入需要创建的节点个数和每个节点的值，按照输入顺序逐个创建节点，并将各个节点串联起来； 3. 实现一个函数printList()，用于从屏幕显示单链表的元素列表，从头节点开始沿着指针域依次输出各个节点的数据域； 4. 实现一个函数deleteNode()，用于删除链表中的一个节点，用户输入要删除的节点位置（从0开始），根据输入的位置找到要删除的节点，将要删除的节点从单链表中删除； 5. 实现一个函数insertNode()，用于在指定位置上插入一个节点，用户需要输入插入节点的值和要插入的位置（从0开始），根据输入的位置找到插入节点的前一个节点，将要插入的节点插入到指定位置。 6. 以上步骤都通过调用主函数来完成，最终通过在屏幕上显示单链表的操作结果来验证程序解决问题的正确性。  实验代码 #include <stdio.h>  #include <stdlib.h>  typedef struct Node {      int data;      struct Node\* next;  } Node;  // 创建一个单链表  Node\* createList(int n) {      Node \*head = NULL, \*tail = NULL;      int i, val;        for (i = 0; i < n; i++) {          printf("请输入第%d个元素的值：", i + 1);          scanf("%d", &val);            Node\* node = (Node\*) malloc(sizeof(Node));          node->data = val;          node->next = NULL;            if (head == NULL) {              head = tail = node;          } else {              tail->next = node;              tail = node;          }      }        return head;  }  // 从屏幕显示单链表元素列表  void printList(Node\* head) {      if (head == NULL) {          printf("链表为空\n");          return;      }        Node\* p = head;      printf("[链表元素列表]：");      while (p != NULL) {          printf(" %d", p->data);          p = p->next;      }      printf("\n");  }  // 删除链表某位置上的元素  void deleteNode(Node\*\* head\_ref, int position) {      if (\*head\_ref == NULL) {          printf("链表为空，无法删除\n");          return;      }        Node\* cur = \*head\_ref;      if (position == 0) { // 删除头节点          \*head\_ref = cur->next;          free(cur);          printf("已删除链表头节点\n");      } else {          Node\* prev = NULL;          int i;          for (i = 0; cur != NULL && i < position; i++) {              prev = cur;              cur = cur->next;          }            if (cur == NULL) {              printf("无法删除链表第%d个元素\n", position);          } else {              prev->next = cur->next;              free(cur);              printf("已删除链表第%d个元素\n", position);          }      }  }  // 在指定的位置插入一个元素  void insertNode(Node\*\* head\_ref, int position, int val) {      Node\* node = (Node\*) malloc(sizeof(Node));      node->data = val;      node->next = NULL;        if (\*head\_ref == NULL) {          if (position == 0) {              \*head\_ref = node;              printf("已在链表头部插入元素：%d\n", val);          } else {              printf("链表为空，无法在链表第%d个位置插入元素：%d\n", position, val);          }      } else if (position == 0) { // 插入头节点          node->next = \*head\_ref;          \*head\_ref = node;          printf("已在链表头部插入元素：%d\n", val);      } else {          Node\* cur = \*head\_ref;          Node\* prev = NULL;          int i;          for (i = 0; cur != NULL && i < position; i++) {              prev = cur;              cur = cur->next;          }            if (cur == NULL && position > i) { // 无法插入              printf("无法在链表第%d个位置插入元素：%d\n", position, val);          } else { // 插入其他位置              node->next = cur;              prev->next = node;              printf("已在链表第%d个位置插入元素：%d\n", position, val);          }      }  }  int main() {      int n;      printf("请输入单链表的元素个数：");      scanf("%d", &n);        Node \*head = createList(n);      printList(head);        int position;      printf("请输入要删除的节点的位置（从0开始）：");      scanf("%d", &position);      deleteNode(&head, position);      printList(head);        int val;      printf("请输入要插入的元素的值：");      scanf("%d", &val);      printf("请输入要插入元素的位置（从0开始）：");      scanf("%d", &position);      insertNode(&head, position, val);      printList(head);        return 0;  } 实验结果 | | | | | | |