专业：微电子科学与技术
姓名：焦天晟
学号：3220105664
日期：2023.04.14

浙江大学实验报告

课程名称：C 程序设计专题　　指导老师：翁恺　　成绩：

实验名称：作业 1：　ADIF 数据处理

一、实验题目要求：

实现一个 GUI 窗口内的单行的文本输入编辑器。

支持左右方向键、回退和删除键、ins 切换插入和覆盖状态，编辑过程中有光标闪烁，插入和覆盖状态的光标形状不同。

回车后，结束输入，将输入的内容在标准输出输出。

二、实验思路

这是一个简易图形程序设计，需要在一个 GUI 界面上显示字符和光标和简易编辑功能。

首先由 GUI 想到要用 ACLlib 绘图函数库，首先是学习 sample 中的 char.c 程序来初步了解 ACLlib 中字符输出的功能，并遍历 ACLlib 库中的库函数来了解如何更准确地利用库函数。

初步了解后确定了要利用如下一些函数：

```
1.  setCaretSize(ca,25);
2.  setCaretPos(0,0);
3.  showCaret();
```

来实现光标；

```
1.  setTextColor(BLACK);
2.  setTextSize(20);
3.  paintText(i*11,0,s)
```

来实现文字输入输出。

```
1.  registerKeyboardEvent(key);
2.  registerCharEvent(print);
```

来处理键盘事件。

另外，我考虑使用双向链表来实现字符的删除和插入。

除 Setup 函数外，另外写 key 函数来实现键盘事件处理和 print 函数来实现字符的输出。

三、实验代码解释

```
#include "acllib.h"
#include <string.h>
#include <stdio.h>
```

程序文件头，包含 ACLlib 库，字符处理库和标准输入输出；

```
void print(char c);
void key(int key,int ev);
```

函数声明；

```
typedef struct _node
{
    char ch;
    struct _node* next;
    struct _node* prev;
}Node;

Node *head = NULL,*tail = NULL, *node_ = NULL;
int  care = 2, careposition = 0, flag = 0 , insert = 0;
```

结构体及基本参数初始化；

```
int Setup()
{
    // 初始化控制台
    initConsole();

    // 初始化窗口
    initWindow("editor",DEFAULT,DEFAULT,900,500);

    // 创建一个双向链表，并将其头指针存储在 head 变量中
    head = (Node*)malloc(sizeof(Node));
    head->next = NULL;
    head->prev = NULL;
```

```
    head->ch = 0;

    // 将当前节点指向头节点，表示初始时光标在文件开头
    node_ = head;

    // 开始绘图，并设定光标大小、位置
    beginPaint();
    setCaretSize(care,25);   // 光标大小
    setCaretPos(0,0);        // 指定光标初始位置

    // 显示光标并设置字体颜色、大小
    showCaret();
    setTextColor(BLACK);
    setTextSize(18);

    // 注册键盘事件和字符事件
    registerKeyboardEvent(key);     //键盘事件
    if(!flag){
        registerCharEvent(print);   // 字符事件
    }
    endPaint();

    flag = 0;

    // 返回 0 表示执行成功
    return 0;
}

void print(char c)
{
   //定义变量
   register unsigned int i = 0,j = 0;
   Node *p = NULL,*q = NULL;
```

```
    beginPaint();
    clearDevice();  // 清空显示设备，准备重新绘制
    if(c == 8){    // 如果输入的是退格键，则删除前面的一个字符
        if(node_ != head){
            node_->prev->next = node_->next; // 让当前节点的上一节点指向当前节点的下一节点
            if(node_->next)
                node_->next->prev = node_->prev; //让当前节点的下一节点指向当前节点的上一节点
            p = node_;
            node_ = p->prev; //将当前节点指向之前的那个节点
            free(p); //释放本次删除的节点的内存

            //如果删完当前节点后出现空链表，需要特殊处理
            if(!(node_->next)) tail = node_;
          }
     }
     else if(c<32||c==127);
     else{
             //以下三种情况表示需要插入新字符到链表中
             if(!(head->next)){
                 head->next = (Node*)malloc(sizeof(Node));
                 tail = head->next;
                 head->next->prev = head;
                 head->next->next= NULL;
                 tail->ch = c;
                 node_ = tail;//初始化
             }
             else if(node_ == tail){
                 p = (Node*)malloc(sizeof(Node));
                 tail->next = p;
                 p->next = NULL;
```

```
            p->prev = tail;
            tail = p;
            tail->ch = c;
            node_ = node_->next;
        }
        else{
            if(care == 2){
                p = (Node*)malloc(sizeof(Node));
                p->next = node_->next;
                node_->next->prev = p;
                node_->next = p;
                p->prev = node_;
                node_ = p;
                node_->ch = c;
            }
            else{
                node_->next->ch = c;
                node_ = node_->next;
            }
        }
}
// 接下来，我们将从头节点遍历整个链表，将各个字符打印到界面上
p = head->next; //从头节点的下一个节点开始找
char s[2] = {0};
static char *classes="WSU";
while(p){
    s[0] = p->ch;
    paintText(i*10,0,s); //使用两个参数：文本在窗口中的横坐标和纵坐标
    i++;
    if(p == node_) j = i; // 如果找到当前节点，就记录它的位置
    p = p->next;
 }
 careposition = j*10; // 计算光标最新需要出现的 X 坐标
```

```
        setCaretPos(careposition,0);  // 设置光标的新位置
        showCaret();  // 显示光标
        endPaint();   //绘图完成，结束绘制
}

void key(int key,int ev)
{
    Node *p = NULL; // 定义指向节点类型 Node 的指针变量 p，并初始化为空
    char x[2] = {0}; // 定义字符数组 x，用于存放要打印的字符和字符串终止符'\0'，并初始化为全零
    if(ev == 0){ // 按键按下事件响应代码块
        flag = 1; // 标记 flag 置为 1，表示对本次输入事件做出响应
        switch(key){
            case 37: // 如果按下了左箭头键，则将当前光标节点的前一个节点设为当前光标；同时移动光标位置和状态
                if(node_->prev){
                    node_ = node_->prev;
                    careposition-=10;
                    setCaretPos(careposition,0);
                    showCaret();
                }
                break;
            case 39: // 如果按下了右箭头键，则将当前光标节点的后一个节点设为当前光标；同时移动光标位置和状态
                if(node_->next){
                    node_ = node_->next;
                    careposition+=10;
                    setCaretPos(careposition,0);
                    showCaret();
                }
                break;
        }
    }
```

```
    else if(ev == 1){ // 按键释放事件响应代码块
        flag = 1; // 标记 flag 置为 1，表示对本次输入事件做出响应
        switch(key){
            case 46: // 如果按下了删除键
                if(node_->next){ // 如果当前光标节点不是最后一个节点，则删除其后的第一个节点
                    p = node_->next;
                    node_->next = p->next;
                    if(p->next)
                        p->next->prev = node_;
                    free(p); // 释放被删除节点占用的内存空间
                    if(!(node_->next)) tail = node_; // 如果此时当前光标节点的下一个节点为空，则更新尾指针
                }
                p = head->next; // 将指针 p 指向链表的第一个节点，即头结点的下一个节点
                beginPaint(); // 开始渲染窗口界面
                clearDevice(); // 清空设备屏幕内容
                int i = 0; // 定义变量 i 用于控制每个字符在窗口中显示的位置
                while(p){ // 遍历链表中的每个节点
                    x[0] = p->ch; // 从链表节点中读取字符并赋值给 x 数组的第一个元素
                    x[1] = '\0'; // 为 x 数组增加字符串终止符
                    paintText(11*i,0,x); // 在设备屏幕上渲染该字符
                    i=i+1;
                    p = p->next; // 指向下一个节点
                }
                showCaret(); // 显示光标
                endPaint(); // 结束渲染窗口界面
                break;
            case 45: // 如果按下了切换光标大小键，则根据当前光标的大小判断切换后的大小，并更新光标状态
                if(care == 2)
```

```
                    care = 10;
                else care = 2;
                setCaretSize(care,25);
                showCaret();
                break;
            case 13: // 如果按下了回车键，则将链表中的每个字符输出到终端上，同时清空链表以备下一次输入
                beginPaint(); // 开始渲染窗口界面
                p = head->next; // 将指针 p 指向链表的第一个节点，即头结点的下一个节点
                while(p){ // 遍历链表中的每个节点
                    printf("%c",p->ch); // 输出该节点存储的字符到终端上
                    p = p->next; // 指向下一个节点
                    if(p) free(p->prev); // 释放上一个节点占用的内存空间
                }
                free(tail); // 释放尾节点的内存空间
                tail = NULL; // 尾指针置为空
                head->next = NULL; // 头节点的下一个节点置为 NULL
                head->prev = NULL; // 头节点的前一个节点也置为 NULL
                head->ch = 0; // 头节点存储的字符设为 0
                node_ = head; // 当前光标节点指向头节点
                printf("\n"); // 输出一个换行符到终端上
                clearDevice();  // 清空设备屏幕内容
                careposition = 0; // 将光标位置初始化为 0
                setCaretPos(careposition,0); // 显示光标
                showCaret(); // 更新光标状态并显示光标
                endPaint(); // 结束渲染窗口界面
        }
    }
}
```

四、实验体会与心得：

这次大程中我首次采用外部库文件进行编辑，这对一个大一对变写大程序接触不多的我来说是一个挑战。这次大程中我学会了阅读，学习和使用不熟悉的外部库函数进行代码的编辑，也增加了对多文件编程的理解与能力。

其次，这次大程我应用了链表进行数据处理，更加深刻地了解到了链表可以给数据处理带来的方便，也加深了对链表的理解。

最后，通过这次大程我也更加熟悉了 Makefile 的编写与应用，对今后的学习与编程有非常大的帮助。