浙江大学实验报告

专业：混合班
姓名：徐圣泽
学号：3190102721
日期：2020.4.23

课程名称：C 程序设计专题　　指导老师：翁恺　　实验名称：函数绘图

一、实验要求

（1）实现一个GUI窗口内的单行的文本输入编辑器。

（2）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。

（3）回车后，结束输入，将输入的内容在标准输出输出。

二、实验思路与过程描述

（1）链表的运用

贯穿本程序的一个不可或缺的组成部分便是链表。无论是各键盘键（左右方向键、回退键、删除键、ins切换键、回车键）的功能实现还是文本内容的输入输出，都需要用到链表。在此之前`p1`的`Linkedlist`程序作业中写过的各个关于链表的实现不同功能的函数，在这里都很好地发挥了其作用。

左右方向键，判断目前链表结点的位置并在一定的条件下进行位置改变，这个位置在程序中可以用来计算和表达光标所处的位置。

回退键和删除键分别是删除当前所在位置前面和后面的结点（删除内容并重新串联链表），其中前者的光标位置不发生改变，后者的光标位置前移一个单位。

ins切换键用于切换状态，这两种状态分别对应着两种不同的功能——插入和覆盖，这两种状态也对应着不同的光标形态。插入是在当前位置于这个链表中插入一个新的结点，其光标形态与普通状态没有差别（其实这个部分本身便对应着普通的状态）。覆盖需要判断当前位置，如果在链表末尾，便也即简单地在链表末尾加入新的结点；如果不是末尾，便要改写当前所在位置的后面一个结点的值（用输入的值替换后面结点的值），但是链表本身的结构没有改变，覆盖状态下的光标形态与之前不同。

回车键用于将在窗口中输入的文本内容在标准输出中输出，这个过程需要遍历已有的链表并逐个打印每个结点中的值，同时将现有链表清空，以便创建新的链表。

（2）回调函数

这个程序中另外一个很重要的组成部分便是回调函数。

```
void KeyListener(int key,int event);
void CharListener(char key);
void MouseListener(int mx,int my,int button,int event);
registerCharEvent(CharListener);
registerKeyboardEvent(KeyListener);
registerMouseEvent(MouseListener);
```

利用几个回调函数，判断鼠标和键盘的操作，并调用各功能函数，完成功能的实现。

注：在本程序的编写过程中，特别感谢赵同学的实验作业，实验思路受到赵同学代码的启发，并根据类似的原理编写了以下代码。

三、实验代码解释

（1）全局变量

在这个程序中，需要定义一些全局变量，在后面的函数中改变这些变量的值，控制相应的操作。

```
static int x;//x代表光标当前所在结点位置，初始位置为0
static List list={NULL,NULL};//创建一个空链表
static int flag=1;//flag初始定义为1，有0和1两种情况，分别代表由ins键切换的插入和覆盖两种状态
```

（2）`callback`部分

这个部分由三个回调函数构成：

```
void KeyListener(int key,int event);
```

```
case 37:Left(&x);break;
case 39:Right(&x,&list);break;
case 8: Backspace(&x,&list);break;
case 45:Insert(&flag);break;
case 46:Delete(&x,&list);break;
case 13:Enter(&x,&list);break;
```

在这个函数中，需要用`if`语句判断`event`的值，如果等于`0`或者`1`二者之一，便执行操作（否则按键一次函数会被调用两次）。然后通过`switch`语句判断按键并调用相应的函数。

```
void CharListener(char key);
```

这个函数用来存储或修改链表中结点的值。

首先需要判断键盘按键是否为几个特定的功能键，如果不是，则继续下面的操作，如果是，则直接调用相应的函数。

```
if(x==0) list_insert(&list,key);//链表头部
else if(0<x&&x<len){//链表中间
    Node* p=(Node*)malloc(sizeof(Node));
    Node* q=list_find(&list,x);
    p->value=key;
    p->next=q->next;
    q->next=p;
}
else list_append(&list,key);//链表尾部
```

如果此时的`flag`值为1，说明此时的状态为插入，判断结点位置，共三种情况，在链表头、中间、尾插入新的结点。

```
if(x==len) list_append(&list,key);//在链表末尾
else list_set(&list,x,key);//不在链表末尾
```

如果此时`flag`的值为0，说明状态为覆盖，判断当前结点位置是否为最后一个，如果是，则在末尾加入，如果不是，利用`list_set`函数修改当前结点位置的值。

```
void MouseListener(int mx,int my,int button,int event)
```

这个函数用来判断鼠标的动作，判断条件是`if(button==1&&event==0&&my<=tsize)`。如果鼠标在文本行中点击，判断点击位置，并将光标移至合理范围内鼠标点击的位置。

```
int len=list_size(&list);
int i=mx/twidth;
if(i<len){
    x=i;
}
else x=len;
```

（3）绘制部分

这部分绘制光标和链表中的文本内容：

```
void paintCaret(int x,int flag){
    int caretwidth;
    if(flag) caretwidth=2;
    else caretwidth=twidth;
    setCaretSize(caretwidth,theight);
    setCaretPos(x*twidth,0);
    showCaret();
}
```

判断`flag`为何值，如果为1，则此时状态为插入，光标宽度为2；如果为0，则此时状态为覆盖，光标宽度与字符宽度相同，为`twdith`。

```
void showtext(List *list);
```

首先需要用一个矩形覆盖上一次编辑的内容`setBrushColor(WHITE);rectangle(0,0,width,height);`继而遍历链表，依次绘制每一个结点中存储的字符，这里需要定义一个字符串数组，用来存储每个字符，然后需要用到函数绘图中曾使用过的的`sprintf`函数，再利用`paintText`在指定位置绘制出字符。

（4）功能键

①方向功能键

这两个函数是判断目前光标所在的位置（这个位置实际上也就是光标所在结点的位置），根据条件决定是否将光标左移或右移。左方向键判断条件：是否位于第一个结点处；右方向键判断条件：是否位于最后一个结点处。

其中函数传入的参数是光标结点位置的地址，以此来改变光标的位置。

②退格键

这个函数传入的参数是`int *px`和`List *list`，分别是关于光标位置的地址和链表表头的地址，这两个参数在后面的函数中也会不断出现。`Backspace`函数判断条件与左方向键相同，删除当前光标所在位置的结点的值，并重新串联链表，光标位置前移一个单位。这里用到了链表的`remove`函数。在链表更新后，利用`showtext`函数重新在窗口中绘制出文本内容。

③删除键

`Delete`函数的判断条件与右方向键相同，删除当前光标所在位置的后面一个结点的值，并重新串联链表，同样用到`remove`和`showtext`函数，但在这个函数中，光标所在的位置没有改变。

④`Insert`切换键

```
void Insert(int *flag){
    if(*flag==1) *flag=0;
    else *flag=1;
}
```

全局变量`flag`的两个值`1`和`0`分别对应着两种状态：插入和覆盖。通过这个函数修改`flag`的值，以此来控制鼠标状态的变化。

⑤回车键

```
void Enter(int *px,List *list){
    static int i=1;
    printf("第%d行:",i++);
    Node *p=list->head;
    for(p=list->head;p;p=p->next){
        printf("%c",p->value);
    }
    printf("\n");
    list_free(list);
    *px=0;
    showtext(list);
}
```

在这个函数中，定义了一个局部静态变量`i`，每次调用`Enter`函数时`i`的值发生改变（增加`1`），`i`的数值代表已编辑的行数。当按下键盘`Enter`键时，在标准输出窗口依次输出链表各结点中存储的字符，并在输出结束时切换至下一行，清空链表，光标位置置零。

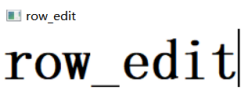
（5）链表函数

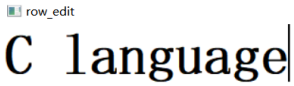
```
void list_free(List *list);
void list_append(List *list, char c);
void list_insert(List *list, char c);
void list_set(List *list, int index, char c);
int list_get(List *list, int index);
int list_size(List *list);
Node* list_find(List *list, int index);
void list_remove(List *list, int index);
```

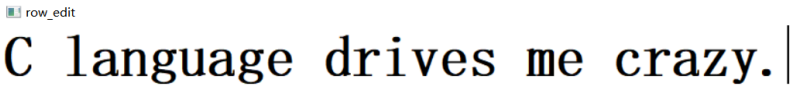
这几个关于链表的函数都在之前的学习过程中有过详细的叙述和解释，尽管某些函数的返回值类型和传入的参数类型发生了一些变化，但原理都与之前相同。

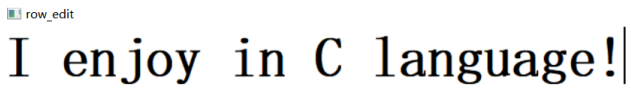
（6）成果展示

以下是几次输入和输出的过程展示。










```
第1行:row_edit
第2行:C language
第3行:C language drives me crazy.
第4行:I enjoy in C language!
第5行:END.
```

四、实验体会和心得

在本次实验中，主要利用了链表和回调函数，将各个键盘功能键的操作实现转变为对链表结点的操作。每次加入新的结点、删除某个位置的结点、或者修改某个位置的结点的值，实际上就是对链表进行操作，继而再重复调用`showtext`函数，绘制出新的状态下对应的“图象”。

除了利用链表解决问题之外，我还想到了一种实现思路——利用字符串数组存储字符。如果使用这种方法，则需要定义一个变量记录当前所在的位置，并对数组中某个元素的值进行存储、删去或修改。但在尝试的过程中，我发现如果定义多个函数，则函数传参的过程较为复杂，但如果全部写在回调函数中，则一个函数的组成过于冗长，不容易检查发现错误。

在这个实验作业中，还有几个想改进之处：

①更多键盘功能的实现，如组合按键，各种快捷键等，这部分我没有深入了解，原理是相似的，但是实现过程肯定更加复杂，不清楚能否用`acllib`库实现。

②根据输入的内容多少和窗口的大小调整字体的大小。在本程序中，如果输入的字符串长度超过了窗口的大小，则无法看见输入的内容。如果想要调整字体大小，则`showtext`函数和`paintCaret`函数都需要随之改变，各个函数之间的联系也需要更加紧密，但在这个过程中，宏定义和最初定义的全局变量的值，都需要根据情况而作出改变，函数传入的参数也需要发生变化，工作量会大大增加。

③鼠标移动过程中形态的变化。鼠标在编辑状态和普通状态下的形态切换，或者在鼠标周围有较明显的显示鼠标当前位置的范围提示。但我尝试后发现，如果需要不断切换状态，则鼠标每移动一次，就需要在窗口中重新绘制一次，但传入参数的过程我并不十分了解，于是最终没能实现。

这个思路并不是我最初时想到的，因为参考了赵同学的实验作业，所以整个实验的思路与原理与他是较为相似的，在某些函数功能的实现和链表函数做了一定的修改，其余都是大同小异，换汤不换药。但的确，在实验的过程中，我对整个程序功能实现的过程有了熟练的把握，同时也复习了链表的知识，收货非常大。

学习他人的代码也是一种学习，从别人的程序中发现自己的程序有哪里不足并加以改进，更能提升自己，但这些进步都要建立在完全吸收和掌握的前提下，如果能再加以改进便是锦上添花。我想，这便是老师每次设置推荐作业的意义。但我希望，在今后的作业中，能够更加积极主动地思考出一个新的做法，将前后所学知识串联，为自己所用。