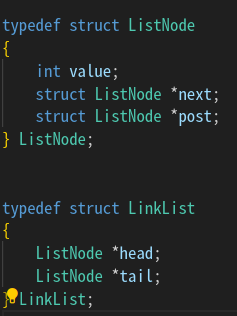
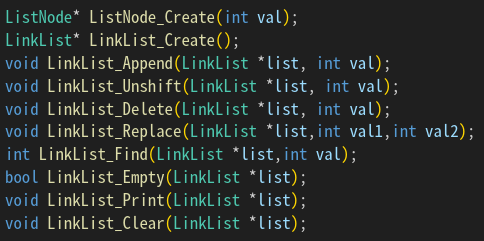
**数据结构可视化项目报告**

一、模块设计

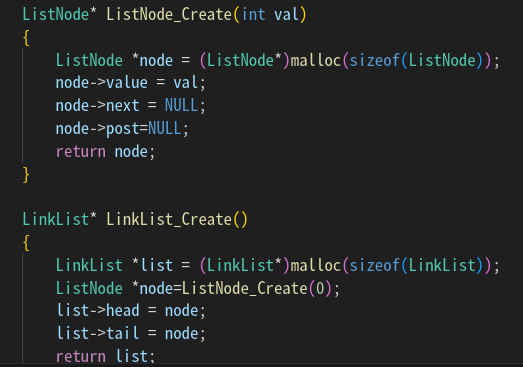
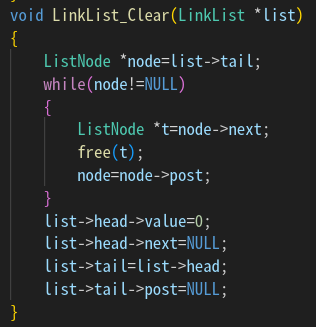
1、数据结构模块

 我在本次项目中选择对链表这一数据结构及其操作进行可视化。首先在头文件linklist.h中，定义了与链表相关的存储结构，分别是结点ListNode和链表整体的结构体LinkList，需要注意的是链表的头结点存储了链表有效数据的长度；头文件中还声明了与链表的操作相关的各种函数，包括对链表的增、删、改、查、输出等。



存储结构 操作相关函数

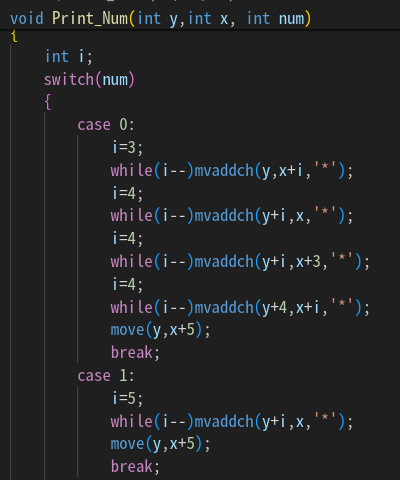
之后在linklist.c文件中对头文件声明的各个函数进行具体实现。其中较为关键的部分是：在链表的初始化创建函数中，采用动态分配空间的方式创建链表，这样可以防止之后的操作反复创建链表消耗过多的空间；在链表的清除函数中也采用相似的方法不断释放每个结点占用的空间。

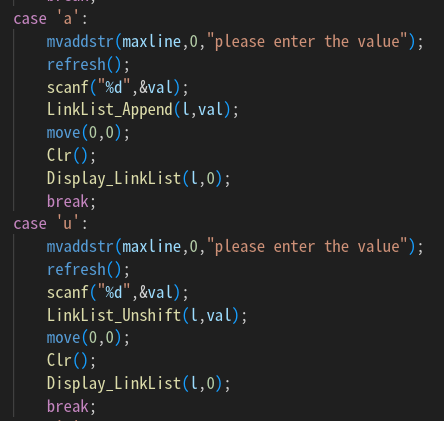
初始化创建函数 清除函数

2、可视化模块

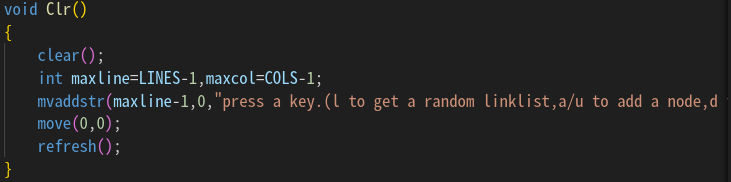
该模块主要通过ncurses库对数据结构的可视化进行了具体实现，使用星号表示像素点对数据进行显示。

其中对于数字的描绘是模块的核心，通过代码控制每个数字都从左上角开始描绘，描绘结束后将光标移至下一数字的起始处。



本模块的另一个较重要的思路是，每次对链表数据进行修改如增加、删除等，需要将屏幕上的链表也更新时，都采用将新的链表整体重新输出的方式，而非只对发生改变的结点单独进行输出。这样做的好处是不需要进行频繁的光标移动和细节调整，所有的改动在内存中完成后再整体在屏幕上显示，提高了效率。

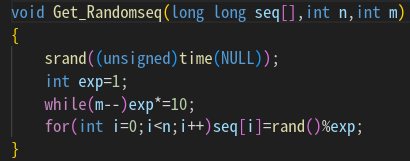
每一个改动操作后都对整个链表重新输出

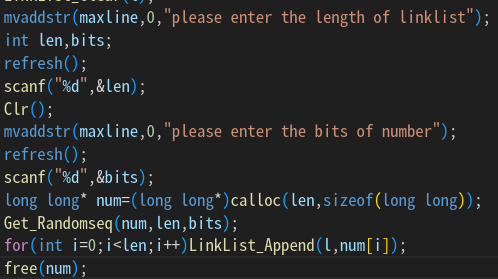
需要注意的是，为了每次链表数据有改变时都将新的链表整体输出，在输入字符进行操作时需要先将屏幕对应位置清空，防止之后覆盖时新旧链表在屏幕上发生重叠，而不改变数据的查找等操作则不需要进行清空。

清屏函数

3、随机序列模块

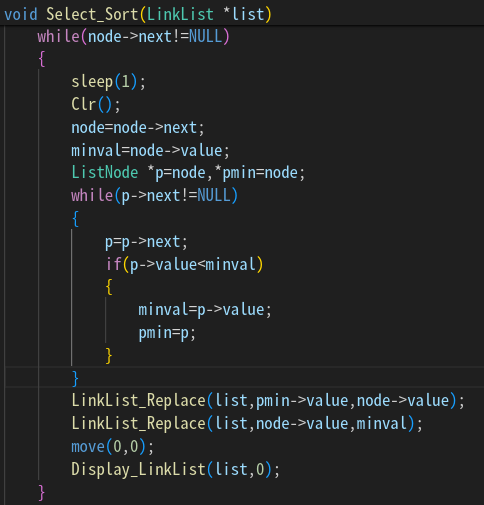
该模块通过时间种子产生随机数，并可通过键盘输入指定序列的长度及随机数的位数，函数的参数seq同样通过calloc动态分配内存获得，节省空间；产生的序列通过append方式加入链表中。

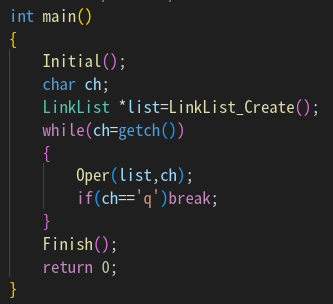




4、排序模块

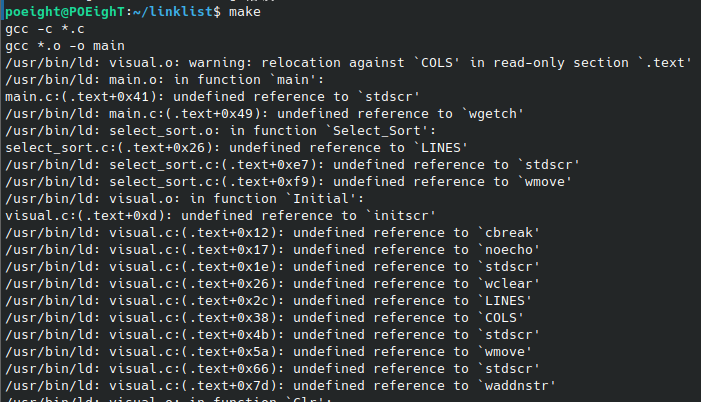
由于数据要求不高，故本项目对链表的排序采用较简单的选择排序算法。在该模块中，我选择在链表上对数据进行排序而非直接在内存上对随机序列进行排序，这是因为我想要在选择排序的过程中把每一次循环得到的最小值标记出来，体现出选择排序的动态过程，直接在链表上操作便于该过程的可视化。

5、主函数

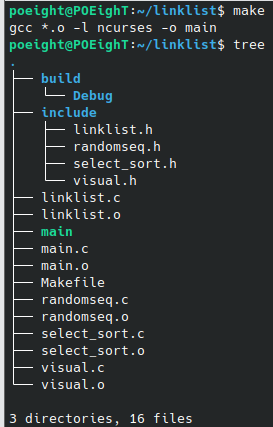
整个项目的主函数就是不断的从键盘读取输入，根据输入的字符控制程序进行各种操作。

二、项目构建

1、目标程序构建

本项目包含Makefile，可通过make快速构建得到目标程序main。需要注意的是在由.o文件生成目标文件时需要使用-l ncurses参数使用ncurses库进行编译，否则程序将无法识别项目中的ncurses库函数。

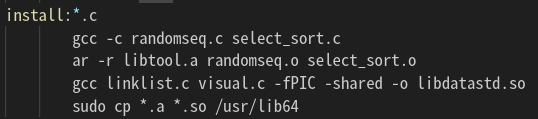
未使用ncures库编译



使用ncurses库编译

2、生成动态、静态链接库

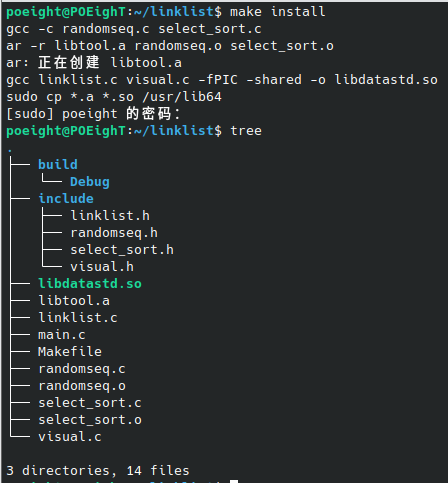
使用make install生成动态、静态链接库并将其安装到linux库路径/usr/lib64.



3、使用库编译文件

使用make installtest通过两个库编译main文件（也可以编译其他项目中的文件）。





三、最终效果

项目最终效果已录制视频提交。