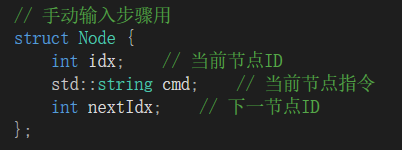
定义如下结构



1. 存储

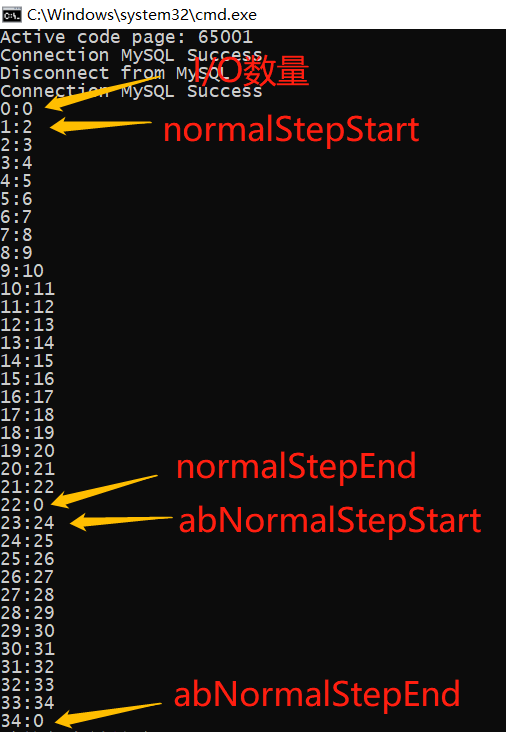
以下是表”cycle\_manual\_step”中的共35个数据，id即idx，operation即cmd，next\_id即nextIdx（next\_id为0为终点，当前有3个终点，一是ioNum用于读取设定的IO数量，二是正常处理终点，三是异常处理终点），mold\_id当前用于区分正常/异常（当前主要是为了获得异常处理步骤的入口Index）

|  |  |  |  |
| --- | --- | --- | --- |
| id | operation | next\_id | mold\_id |
| 22 | 跟踪 | 0 | 1 |
| 21 | #1#显示# | 22 | 1 |
| 20 | #0#显示# | 21 | 1 |
| 19 | #0#2#1#允许关模#0# | 20 | 1 |
| 18 | #1#暂停# | 19 | 1 |
| 17 | #0#暂停# | 18 | 1 |
| 16 | #1#检查# | 17 | 1 |
| 15 | #延时#250#顶出后延时# | 16 | 1 |
| 14 | #1#2#1#顶针退回#0# | 15 | 1 |
| 13 | #1#2#0#顶针退回#0# | 14 | 1 |
| 12 | #1#2#1#顶针退回#0# | 13 | 1 |
| 11 | #1#2#0#顶针退回#0# | 12 | 1 |
| 10 | #0#1#1#允许顶出#0# | 11 | 1 |
| 9 | #0#检查# | 10 | 1 |
| 8 | #延时#25#顶出前延时# | 9 | 1 |
| 7 | #0#2#0#允许关模#0# | 8 | 1 |
| 6 | #1#1#1#开模完成#1# | 7 | 1 |
| 5 | #1#2#0#顶针退回#1# | 6 | 1 |
| 34 | #1#清除# | 0 | 0 |
| 32 | #0#2#1#允许关模#0# | 33 | 0 |
| 33 | #0#清除# | 34 | 0 |
| 31 | #0#1#1#允许顶出#0# | 32 | 0 |
| 30 | #1#3#1#安全门关#0# | 31 | 0 |
| 29 | #1#3#0#安全门关#0# | 30 | 0 |
| 28 | 保存 | 29 | 0 |
| 4 | #0#5#1#绿灯/正常运行#1# | 5 | 1 |
| 27 | #1#显示# | 28 | 0 |
| 26 | #0#显示# | 27 | 0 |
| 3 | #0#2#1#允许关模#1# | 4 | 1 |
| 25 | #0#6#1#红灯/警报#0# | 26 | 0 |
| 24 | #0#2#0#允许关模#0# | 25 | 0 |
| 23 | #0#1#0#允许顶出#0# | 24 | 0 |
| 2 | #0#1#0#允许顶出#1# | 3 | 1 |
| 1 | #1#1#0#开模完成#0# | 2 | 1 |
| 0 | #ioNum#6# | 0 | 1 |

1. 程序中结构构建

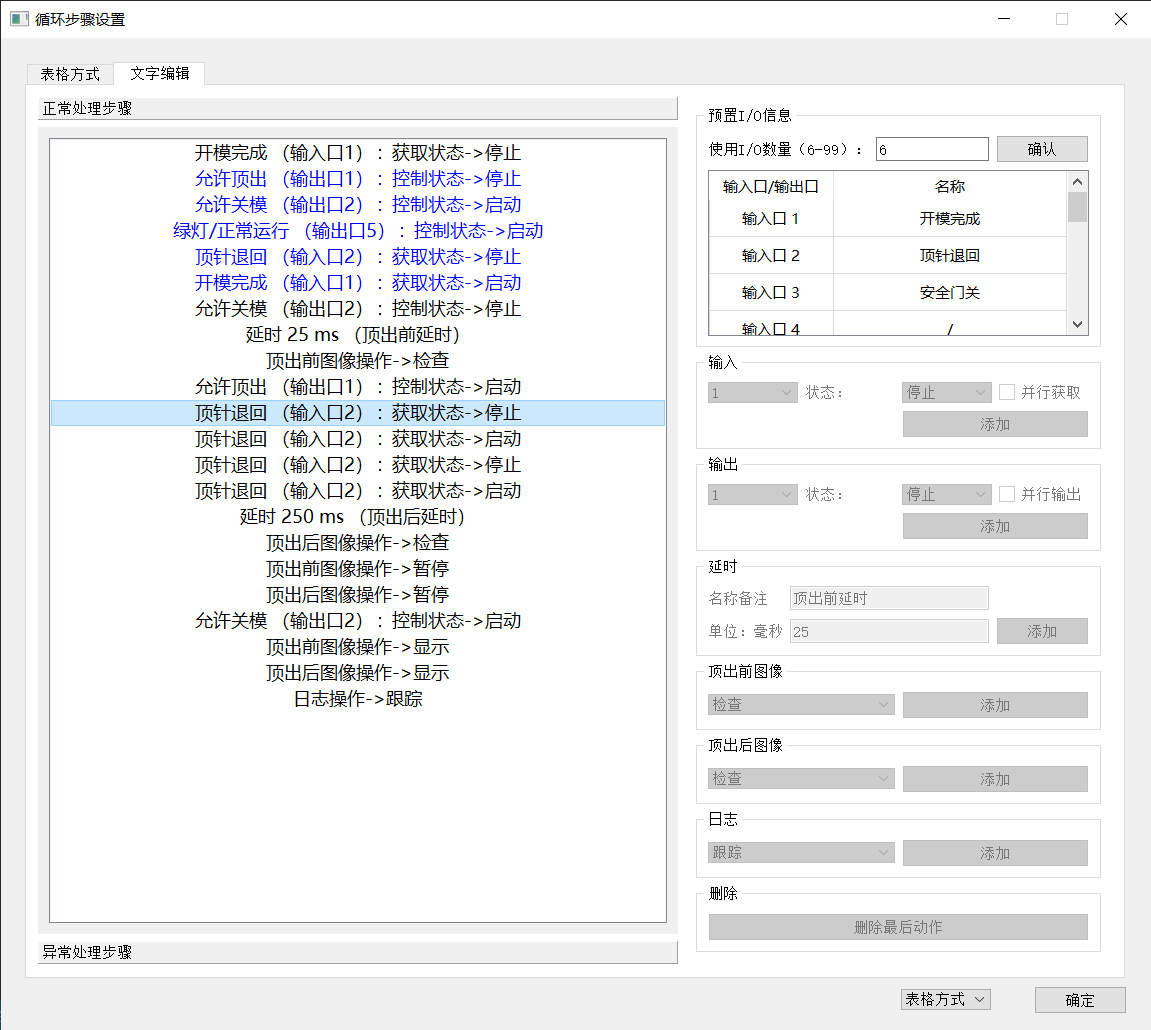


1. manualStep为从表中读取的前三列数（分别sort asc by “mold\_id = 1”时的id与“mold\_id = 0”时的id）汇总；
2. 使用manualStep中的第一列idx与第三列nextIdx构建一张简单的图，每个点的入度出度都为1



1. 设置操作

设置目前我主要考虑的是插入操作，这也是我构建一张简单Graph的原因。



插入操作：用户通过双击左侧指令某一item，程序中获取该item的索引，继而可以通过点击右侧原先一系列插入指令来在当前双击这一行下方进行插入操作，后方原先指令向后移动，程序中表现为数据直接插在总数据manualStep的后方（如从idx = 36开始插入），修改该指令所代表的nextIdx指向为最新的idx（如当前双击第11行指令，原本该指令的nextIdx指向12，改变该nextIdx指向idx为36的节点），而当用户插入完毕后，更新最后的指令nextIdx指向（如插入5条新指令后，最后一条指令的idx为40，nextIdx就可以设置为12），更新manualStep数据，保存至数据库表”cycle\_manual\_step”中。