

- 1.自我介绍
- 2.数据结构课程大纲
- 3.数据结构学习注意事项
- 4.数据结构相关概念
 - 4.1什么是数据
 - 4.2什么是结构
 - 4.3什么是数据结构

- 5.算法
 - 5.1什么是算法
 - 5.2算法的特点
 - 5.3什么样的算法是好的算法
 - 5.4算法的时间复杂度

6.数据结构总览

- 7.顺序表
 - 7.1什么是顺序表
 - 7.2顺序表的结构
 - 7.3顺序表的特点
 - 7.4顺序表常见的操作
 - 7.4顺序表的代码
 - seqlist.h
 - seqlist.c
 - main.c

8.顺序表的优缺点

9.使用vscode对代码进行调试

1.自我介绍

代 daizs_bj@hqyj.com 2863238200

2.数据结构课程大纲

- 1. 数据结构相关概念
- 2. 顺序表
- 3. 单链表
- 4. 单向循环链表
- 5. 双向链表
- 6. 双向循环链表
- 7. 栈
- 8. 队列
- 9. 树
- 10. 图（选学）
- 11. 冒泡排序
- 12. 选择排序
- 13. 归并排序
- 14. 快速排序
- 15. 哈希查找

3.数据结构学习注意事项

- 1. c语言基础
- 2. 数据结构学完主要是使用嵌入式学科中使用
- 3. 数据结构课程逻辑思维需要比较高
- 4. 多写代码，多调试代码
- 5. 使用vscode完成代码的编写
 - https://note.youdao.com/ynotes/index.html?id=ea4796f0aa4344e11bae0d5bdfe32cc8&type=notebook&_time=1652253468820
- 6. DC23032-数据结构笔记链接：
 - <http://note.youdao.com/noteshare?id=0a505339e5cb0f696aad8660246ea87b>

4.数据结构相关概念

4.1什么是数据

数据就是描述某种信息的一种载体，能够被计算机识别存储并参与运算。

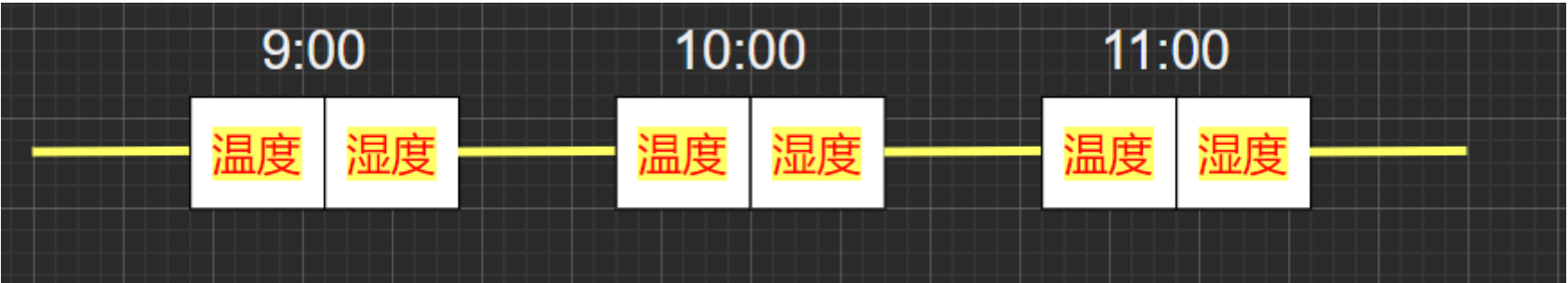
4.2什么是结构

结构：它就是描述数据与数据之间逻辑关系，它分为链式关系，层次关系，网状关系。

链式关系：

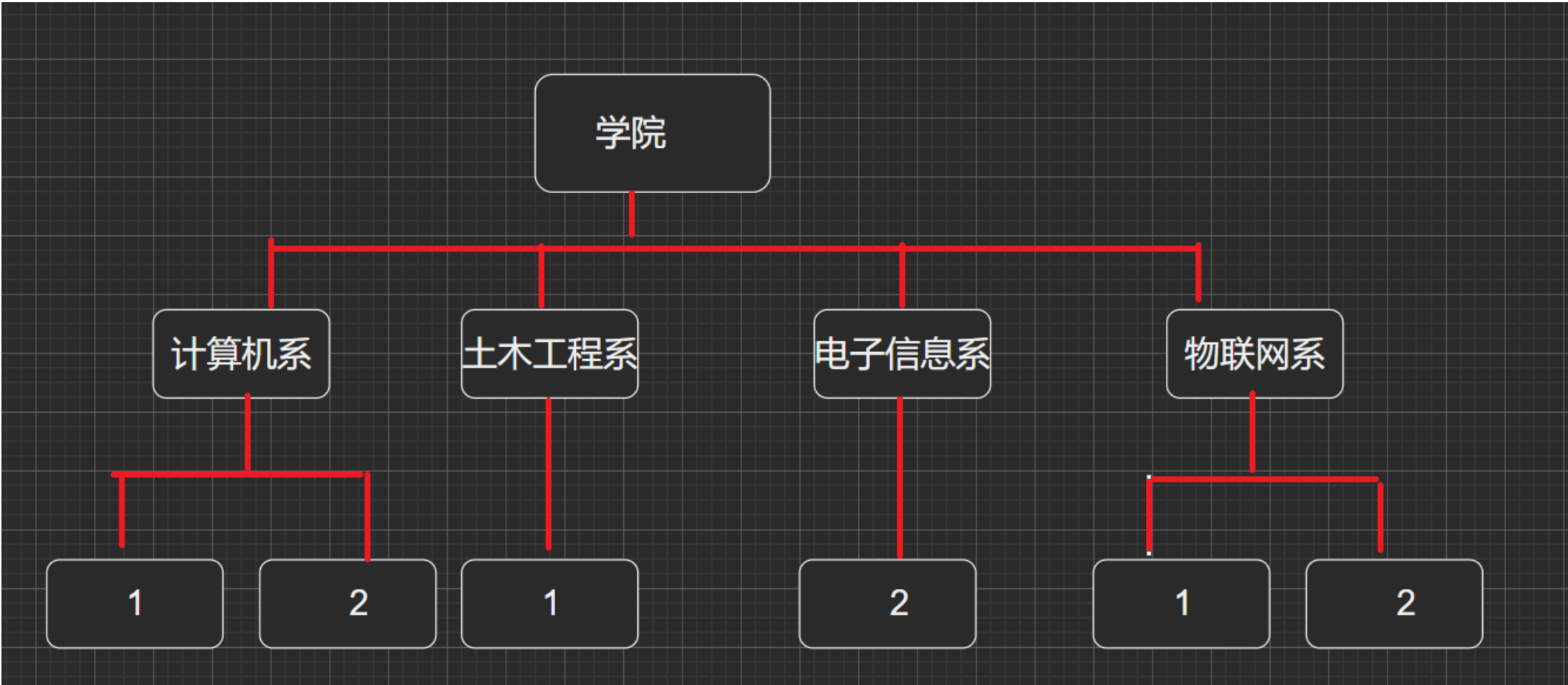
链式结构中有一个节点头，节点头没有前驱，之后后继节点，尾结点没有后继只有前驱

其余的节点只能有一个前驱，之后有一个后继。



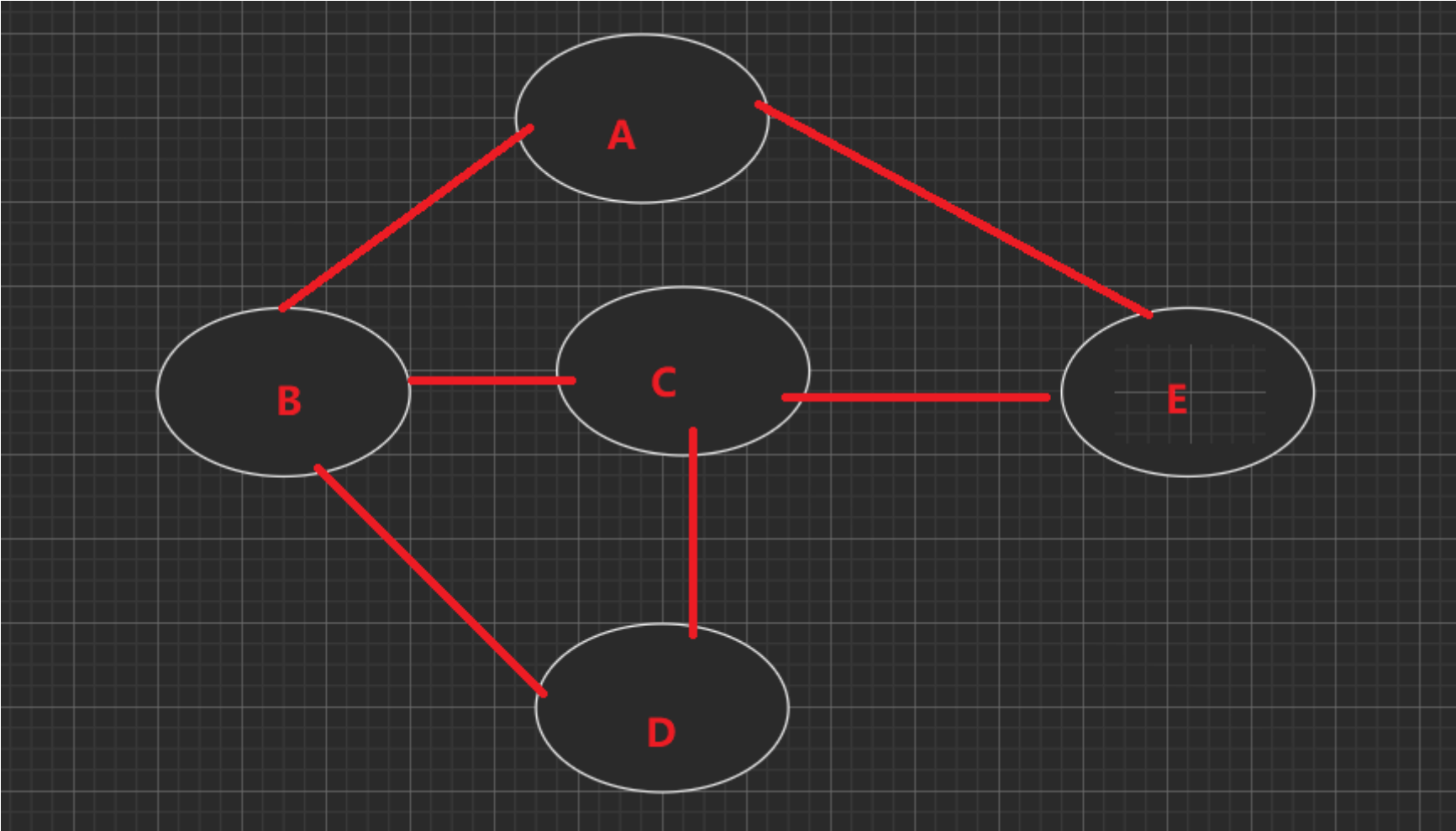
层次关系：

头节点没有前驱，只有后继，尾节点没有后继只有前驱。其余的节点只能有一个前驱，可以有一个或多个后继



网状关系：

每个节点都可以有一个或者多个前驱，也可以有一个和或者多个后继



4.3什么是数据结构

数据结构：指数据之间的相互关系，包含下面三方面的内容：

- 逻辑结构：表示数据运算之间的抽象关系（如邻接关系、从属关系等），按每个元素可能具有的直接前趋数和直接后继数将逻辑结构分为“线性结构”和“非线性结构”两大类。
- 存储结构：逻辑结构在计算机中的具体实现方法，分为顺序存储方法、链接存储方法。
索引存储方法、散列存储方法（哈希）。
- 数据运算：对数据进行的操作，如插入、删除、查找、排序等。

5.算法

5.1什么是算法

算法（Algorithm）是一个有穷规则（或语句、指令）的有序集合。它确定了解决某一问题的一个运算序列。对于问题的初始输入，通过算法有限步的运行，产生一个或多个输出。

5.2算法的特点

- (1) **有穷性** —— 算法执行的步骤（或规则）是有限的；
- (2) **确定性** —— 每个计算步骤无二义性；
- (3) **可行性** —— 每个计算步骤能够在有限的时间内完成；
- (4) **输入** —— 算法有零个或多个外部输入；
- (5) **输出** —— 算法有一个或多个输出。

这里要说明的是，算法与程序既有联系又有区别。二者都是为完成某个任务，或解决某个问题而编制的规则（或语句）的有序集合，这是它们的共同点。区别在于：其一，算法与计算机无关，但程序依赖于具体的计算机语言。其二，算法必须是有穷尽的，但程序可能是无穷尽的。其三，算法可忽略一些语法细节问题，重点放在解决问题的思路，但程序必须严格遵循相应语言工具的语法。算法转换成程序后才能在计算机上运行。另外，在设计算法时，一定要考虑它的确定性，即算法的每个步骤都是无二义性的（即一条规则不能有两种以上的解释）

5.3什么样的算法是好的算法

- 1. 空间复杂度低
- 2. 时间复杂度低
- 3. 好的算法便于理解和维护（尽量精简）

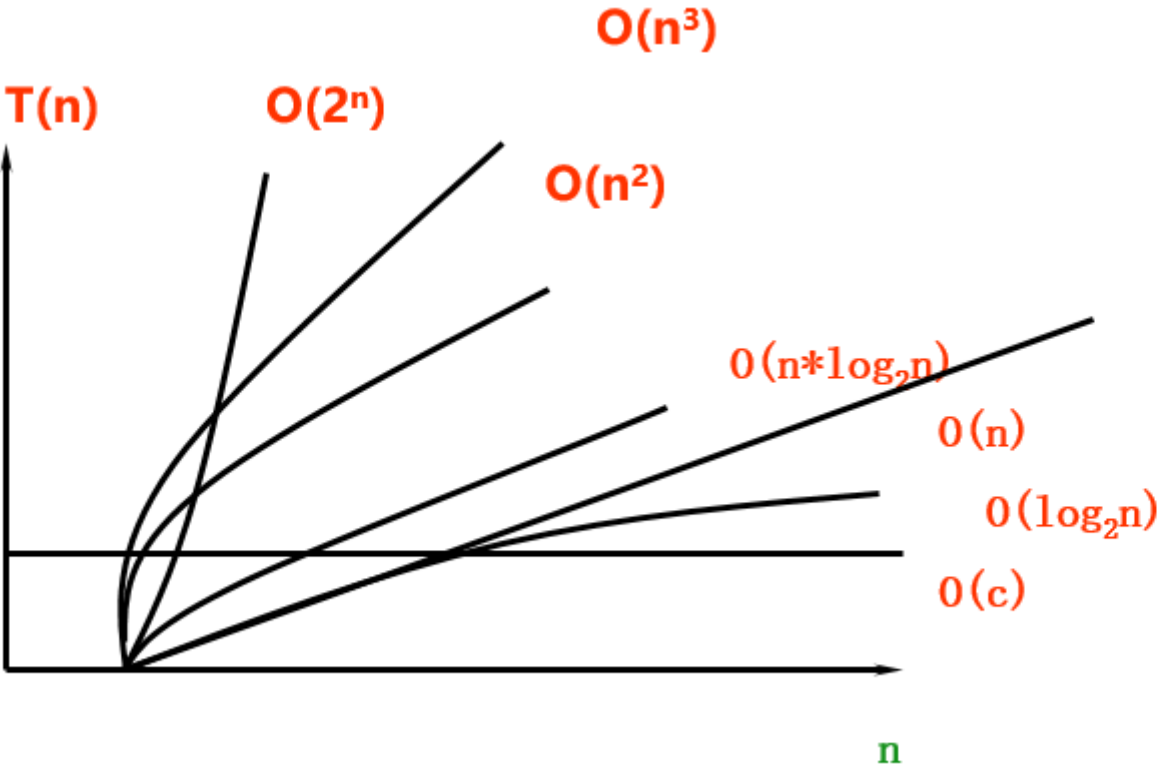
5.4算法的时间复杂度

T(n) =O(f(n))

T:程序运行的时间

O():时间复杂度表示方法

f(n):语句执行的步骤



- 常量阶时间复杂度O(1)

```
1 void swap(int *p,int *q)
2 {
3     int tmp;
4     tmp = *q;
5     *q = *p;
6     *p = tmp;
7 }
```

- 线性阶时间复杂度O(n)

```
1 for(int i=1;i<=n;i++){
2     sum+=i;
3 }
```

- 平方或者立方阶时间复杂度O(n²) O(n³)

```
1 for(int i=0;i<N-1;i++){
2     for(int j=0;j<N-i-1;j++){
3         if(arr[j]>arr[j+1]){
4             swap(&arr[j],&arr[j+1]);
5         }
6     }
7 }
```

如果3层嵌套的循环时间复杂度就是 $O(n^3)$

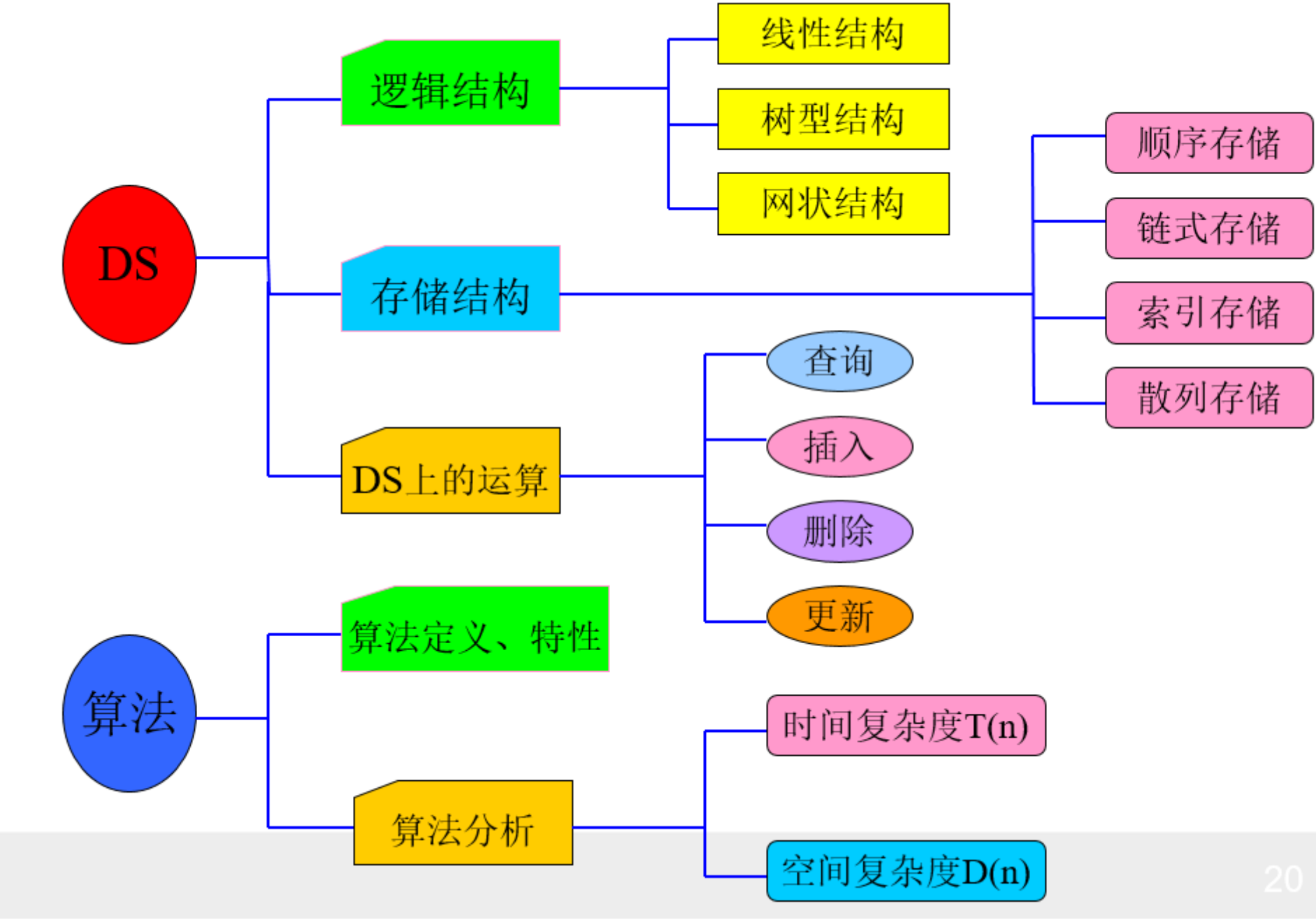
- 对数阶时间复杂度 $O(\log_2 n)$

```
1 int i=1
2 while(i<n){
3     i = i*2;
4 }
5
6 2^x = n
7 x = log2n
```

- 线性对数阶 $O(n\log_2 n)$

```
1 for(int j=0;j<n;j++){
2     int i=1
3     while(i<n){
4         i = i*2;
5     }
6 }
7
```

6.数据结构总览

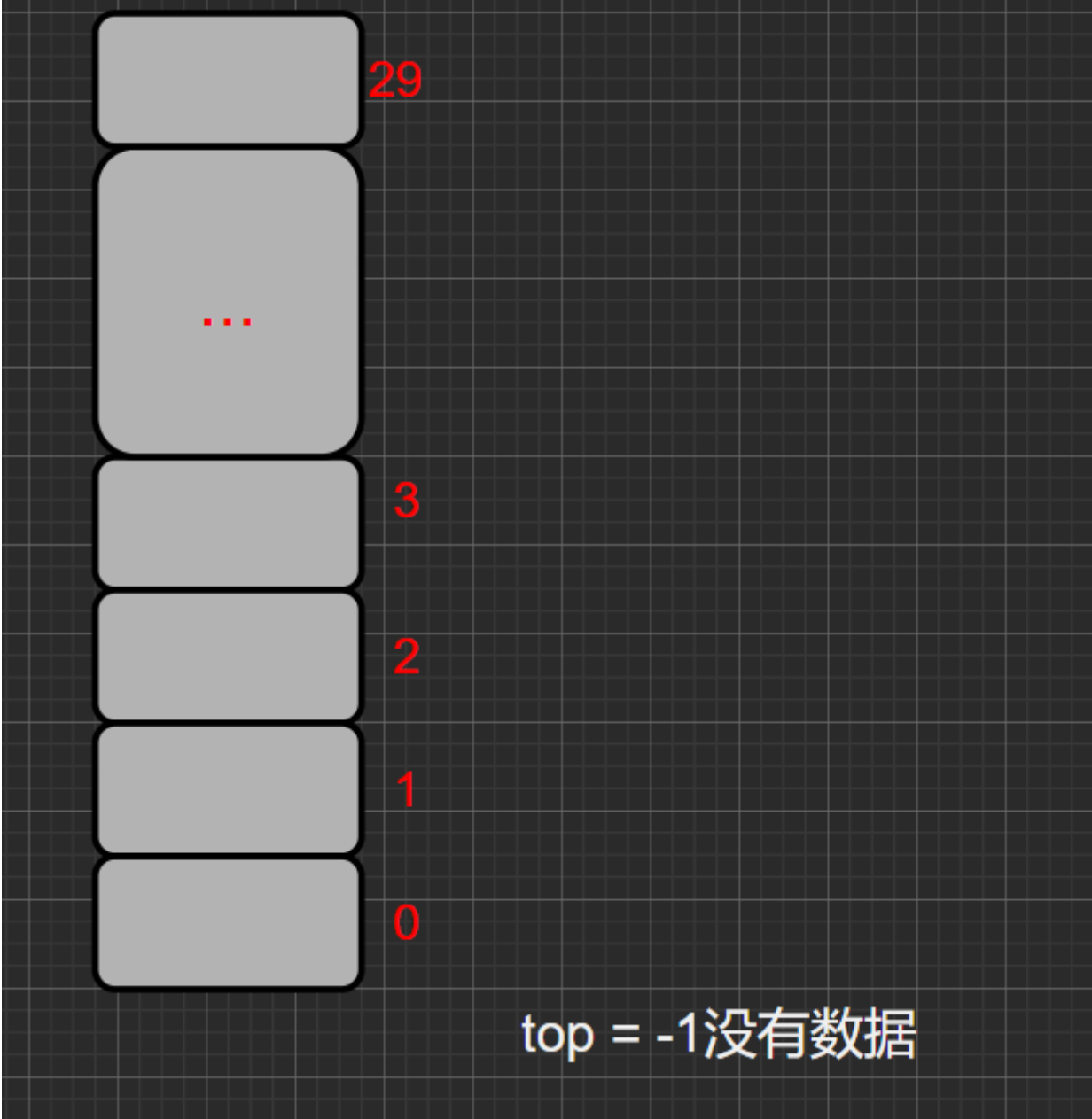


7.顺序表

7.1什么是顺序表

顺序表：顺序表就是线性表的顺序存储

7.2顺序表的结构



```
1 #define N 30
2 #define datatype int
3 typedef struct{
4     datatype data[N];
5     int top;
6 }seqlist_t;
```

7.3顺序表的特点

data[0] :顺序表的头，没有前驱，只有后继
data[29]:顺序表的尾，没有后继，只有前驱
data[n] :只有一个前驱节点和一个后继节点

7.4顺序表常见的操作

1. 创建顺序表
2. 判满
3. 插入节点数据
4. 顺序表的遍历
5. 判空
6. 删除节点数据
7. 根据位置查询数据
8. 根据数据查询位置
9. 根据位置修改数据
10. 根据数据修改数据
11. 任意位置插入数据
12. 任意位置删除数据
13. 删除相同的数据
14. 数据按照小大排序

7.4顺序表的代码

seqlist.h

```
1 #ifndef __SEQLIST_H__
2 #define __SEQLIST_H__
3 #include <stdio.h>
4 #include <stdlib.h>
5
6 #define N 30
```

```
7 #define datatype int
8 typedef struct {
9     datatype data[N]; //定义数组，里面有30个数据
10     int top;          //最后一个数据成员在数组中的下标
11 } seqlist_t;
12
13 seqlist_t* SeqListCreate(void);
14 int SeqListIsFull(seqlist_t *h);
15 int SeqListInsertHead(seqlist_t*h,datatype data);
16 void SeqListShow(seqlist_t *h);
17 int SeqListIsEmpty(seqlist_t *h);
18 datatype SeqListDeleteTail(seqlist_t *h);
19 datatype SeqListCheckDataByPos(seqlist_t *h,int pos);
20 int SeqListCheckPosByData(seqlist_t *h,datatype data);
21 int SeqListUpdateDataByPos(seqlist_t *h,int pos,datatype data);
22 int SeqListUpdateDataByData(seqlist_t *h,datatype odata,datatype ndata);
23 int SeqListInsertDataByPos(seqlist_t *h,int pos,datatype data);
24 datatype SeqListDeleteDataByPos(seqlist_t *h,int pos);
25 void SeqListDeleteSameData(seqlist_t *h);
26 void SeqListBubbleSort(seqlist_t *h);
27 #endif
```

seqlist.c

```
1 #include "seqlist.h"
2 /*
3  *功能： 创建顺序表
4  *参数：
5  *   @::无
6  *返回值：成功返回顺序表的首地址h，失败返回NULL
7  */
8 seqlist_t* SeqListCreate(void)
9 {
10     seqlist_t* h;
11     // 1.给h指针分配内存空间
12     h = (seqlist_t*)malloc(sizeof(*h));
13     if (h == NULL) {
14         printf("%s:malloc error\n", __func__);
15         return NULL;
16     }
17
18     // top=-1代表顺序表中没有数据
19     h->top = -1;
20
21     // 将h地址返回
22     return h;
23 }
24
25 /*
26 *功能： 判断顺序表是否是空
27 *参数：
28 *   @:h:顺序表指针
29 *返回值：满返回1，否则返回0
30 */
31 int SeqListIsFull(seqlist_t* h)
32 {
33     // 如果满返回1，否则返回0
34     return (h->top + 1) == N ? 1 : 0;
35 }
36
37 /*
38 *功能： 向顺序表中插入数据
39 *参数：
40 *   @:h:顺序表的指针
41 *   @:data:向顺序表中插入的数据
42 *返回值：成功返回0，失败返回-1
43 */
44 int SeqListInsertHead(seqlist_t* h, datatype data)
45 {
46     // 判满
47     if (SeqListIsFull(h)) {
48         printf("SeqList Is Full\n");
49         return -1;
50     }
51     // 将数据插入顺序表
52     h->data[++h->top] = data;
53
54     // 成功返回0
55     return 0;
56 }
57
58 /*
59 *功能： 顺序表的遍历
```

```
60  *参数:
61  *   @:h:顺序表的指针
62  *返回值:无
63  */
64 void SeqListShow(seqlist_t* h)
65 {
66     for (int i = 0; i <= h->top; i++) {
67         printf("%d ", h->data[i]);
68     }
69     putchar(10);
70 }
71
72 /*
73  *功能: 判断顺序表是否是空
74  *参数:
75  *   @:h:顺序表的指针
76  *返回值:如果是空返回真1, 如果不是空就返回假0
77  */
78 // 注释 ctrl+/
79 // 代码对齐 alt+shift+f
80 int SeqListIsEmpty(seqlist_t* h)
81 {
82     return h->top == -1 ? 1 : 0;
83 }
84
85 /*
86  *功能: 删除顺序表中数据 (尾部)
87  *参数:
88  *   @:h:顺序表的地址
89  *返回值:成功返回删除的数据, 失败返回 (datatype)-1
90  */
91 datatype SeqListDeleteTail(seqlist_t* h)
92 {
93     // 1.判空
94     if (SeqListIsEmpty(h)) {
95         printf("%s is empty\n", __func__);
96         return (datatype)-1;
97     }
98     // 删除数据
99     return h->data[h->top--];
100 }
101
102 /*
103  *功能: 根据位置查询数据
104  *参数:
105  *   @:h:顺序表指针
106  *   @:pos:位置下标
107  *返回值:成功返回查询到的数据, 失败返回 (datatype)-1
108  */
109 datatype SeqListCheckDataByPos(seqlist_t* h, int pos)
110 {
111     // 1.判断pos是否合法
112     if (pos < 0 || pos > h->top) {
113         printf("%s pos error\n", __func__);
114         return (datatype)-1;
115     }
116     // 2.返回对应位置的数据
117     return h->data[pos];
118 }
119
120 /*
121  *功能: 根据数据查询位置
122  *参数:
123  *   @:h:顺序表的指针
124  *   @:data:数据
125  *返回值:成功返回下标, 失败返回-1
126  */
127 int SeqListCheckPosByData(seqlist_t* h, datatype data)
128 {
129     for (int i = 0; i <= h->top; i++) {
130         if (data == h->data[i]) {
131             return i;
132         }
133     }
134     printf("%s data is not exist\n", __func__);
135     return -1;
136 }
137
138 /*
139  *功能: 根据位置更新数据
140  *参数:
141  *   @:h:顺序表的指针
142  *   @:pos:位置
143  *   @:data:数据
144  *返回值:成功返回0, 失败返回-1
```

```
144  */
145  int SeqListUpdateDataByPos(seqlist_t* h, int pos, datatype data)
146  {
147      // 1.判断位置
148      if (pos < 0 || pos > h->top) {
149          printf("%s pos error\n", __func__);
150          return -1;
151      }
152      // 2.修改对应位置的数据
153      h->data[pos] = data;
154
155      // 3.返回
156      return 0;
157  }
158
159  /*
160   *功能： 根据数据更新数据
161   *参数：
162   *   @:h:顺序表的指针
163   *   @:odata:旧的数据
164   *   @:ndata:新的数据
165   *返回值:成功返回0，失败返回-1
166   */
167  int SeqListUpdateDataByData(seqlist_t* h,
168      datatype odata, datatype ndata)
169  {
170      // 找到表中odata的位置，进行数据更新
171      for (int i = 0; i <= h->top; i++) {
172          if (odata == h->data[i]) {
173              h->data[i] = ndata;
174              return 0;
175          }
176      }
177      printf("%s data not found\n", __func__);
178      return -1;
179  }
180
181  /*
182   *功能： 任意位置插入数据
183   *参数：
184   *   @:h:顺序表的指针
185   *   @:pos:位置
186   *   @:data:数据
187   *返回值:成功返回0，失败返回-1
188   */
189  int SeqListInsertDataByPos(seqlist_t* h, int pos, datatype data)
190  {
191      // 1.判断pos合法性
192      if (pos < 0 || pos > (h->top + 1)) {
193          printf("%s pos error\n", __func__);
194          return -1;
195      }
196      // 2.判满
197      if (SeqListIsFull(h)) {
198          printf("%s is full\n", __func__);
199          return -1;
200      }
201      // 3.循环将数据向后搬移
202      for (int i = h->top; i >= pos; i--) {
203          h->data[i + 1] = h->data[i];
204      }
205      // 4.插入数据
206      h->data[pos] = data;
207
208      // 5.让top自加
209      h->top++;
210
211      return 0;
212  }
213
214  /*
215   *功能： 任意位置删除数据
216   *参数：
217   *   @:h:顺序表的指针
218   *   @:pos:位置
219   *返回值:成功返回删除的数据，失败返回（datatype）-1
220   */
221  datatype SeqListDeleteDataByPos(seqlist_t* h, int pos)
222  {
223      datatype data;
224      // 1.判断pos的合法性
225      if (pos < 0 || pos > h->top) {
226          printf("%s pos is error\n", __func__);
227          return (datatype)-1;
228      }
229      // 2.判空
```



```

228     if (SeqListIsEmpty(h)) {
229         printf("%s seqlist is empty\n", __func__);
230         return (datatype)-1;
231     }
232     // 3.数据循环向前搬移
233     data = h->data[pos];
234     for (int i = pos; i < h->top; i++) {
235         h->data[i] = h->data[i + 1];
236     }
237     // 4.将top的值--
238     h->top--;
239
240     return data;
241 }
242 /*
243 *功能: 删除顺序表中重复的数据
244 *参数:
245 *   @:h: 顺序表的指针
246 *返回值: 无
247 */
248 void SeqListDeleteSameData(seqlist_t *h)
249 {
250     for(int i=0;i<h->top;i++){
251         for(int j=i+1;j<=h->top;j++){
252             if(h->data[i]==h->data[j]){
253                 SeqListDeleteDataByPos(h,j);
254                 j--;
255             }
256         }
257     }
258 }
259 void SeqListBubbleSort(seqlist_t *h)
260 {
261     for(int i=0;i<h->top;i++){
262         for(int j=0;j<h->top-i;j++){
263             if(h->data[j]>h->data[j+1]){
264                 h->data[j] ^= h->data[j+1];
265                 h->data[j+1] ^= h->data[j];
266                 h->data[j] ^= h->data[j+1];
267             }
268         }
269     }
270 }

```

main.c

```

1  #include "seqlist.h"
2
3  int main(int argc, const char* argv[])
4  {
5      seqlist_t* h;
6
7      h = SeqListCreate();
8      if (h == NULL) {
9          printf("SeqListCreate error\n");
10         return -1;
11     }
12
13     SeqListInsertHead(h,7);
14     SeqListInsertHead(h,3);
15     SeqListInsertHead(h,4);
16     SeqListInsertHead(h,3);
17     SeqListInsertHead(h,3);
18     SeqListInsertHead(h,2);
19     SeqListInsertHead(h,6);
20
21     // printf("pos = %d\n",SeqListCheckPosByData(h,444));
22     // SeqListUpdateDataByPos(h,0,666);
23
24     // SeqListShow(h);
25
26     // SeqListUpdateDataByData(h,7,111);
27     // SeqListShow(h);
28
29     // SeqListInsertDataByPos(h,3,555);
30     // SeqListShow(h);
31     // printf("delete = %d\n",SeqListDeleteDataByPos(h,-1));
32     SeqListShow(h);
33     SeqListDeleteSameData(h);
34     SeqListShow(h);
35     SeqListBubbleSort(h);
36     SeqListShow(h);
37     return 0;

```

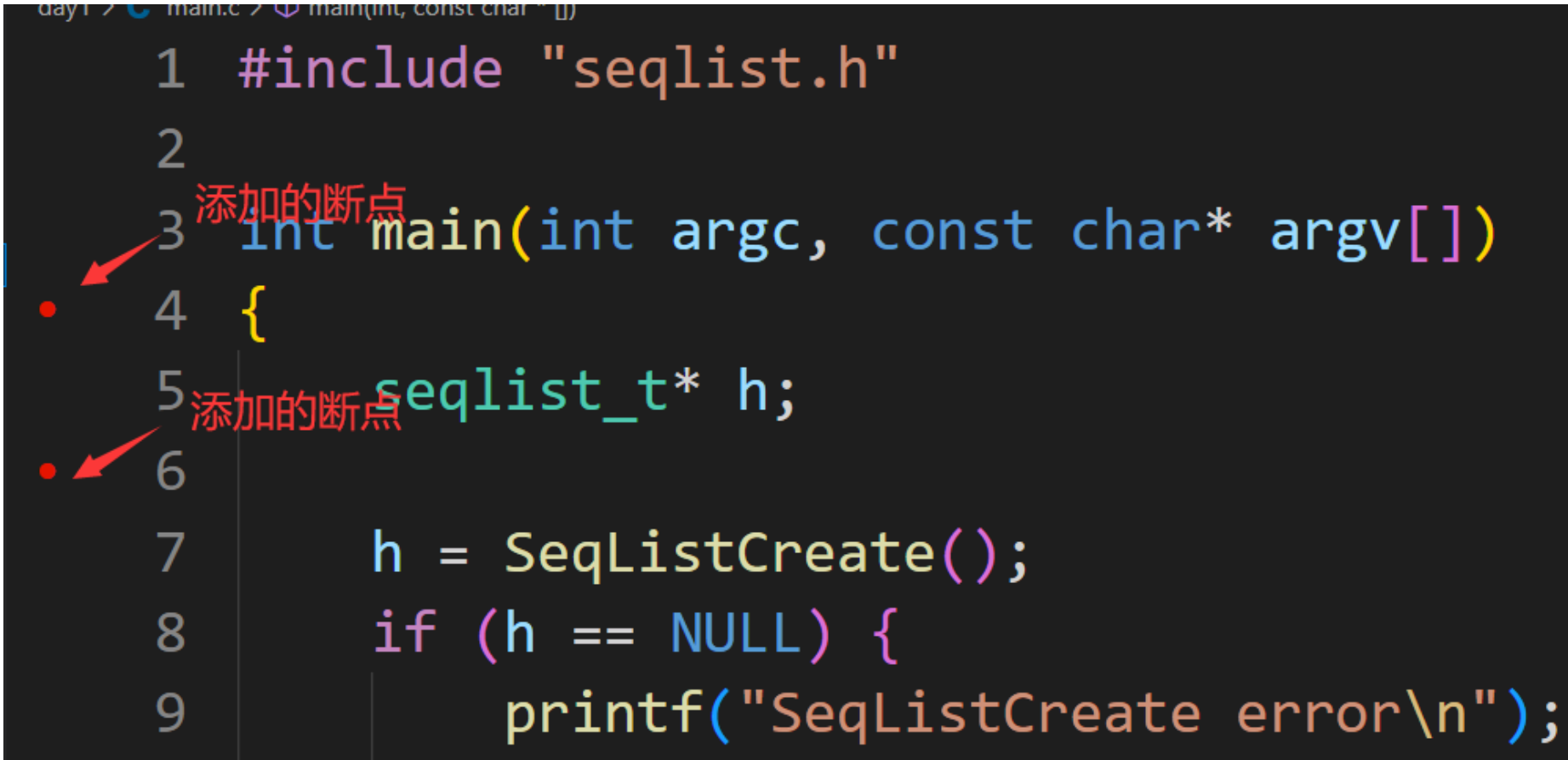
8. 顺序表的优缺点

顺序表优点：因为顺序表的数据是通过数组实现的，内存上连续，并且对数据操作的时候直接下标就可以访问数据的内部成员的。

顺序表的缺点：顺序表中数据定义的是数组，大小是固定的，不方便后序的扩展。另外在进行任意位置插入和任意位置删除的时候时间复杂度高。

9. 使用vscode对代码进行调试

1. 先对调试的代码加断点

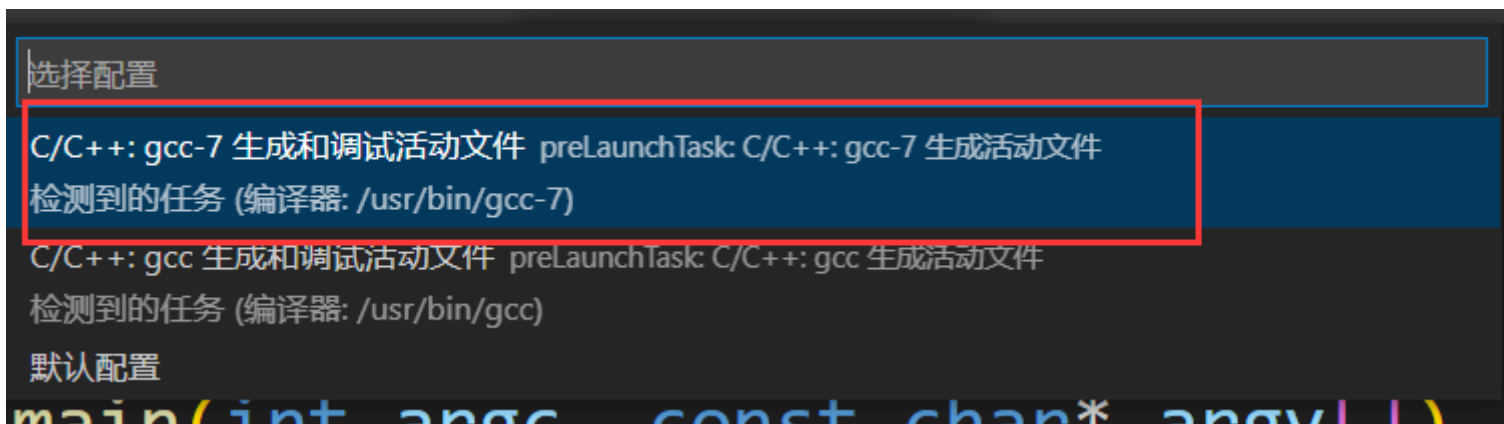


```
1 #include "seqlist.h"
2
3 int main(int argc, const char* argv[])
4 {
5     seqlist_t* h;
6
7     h = SeqListCreate();
8     if (h == NULL) {
9         printf("SeqListCreate error\n");
```

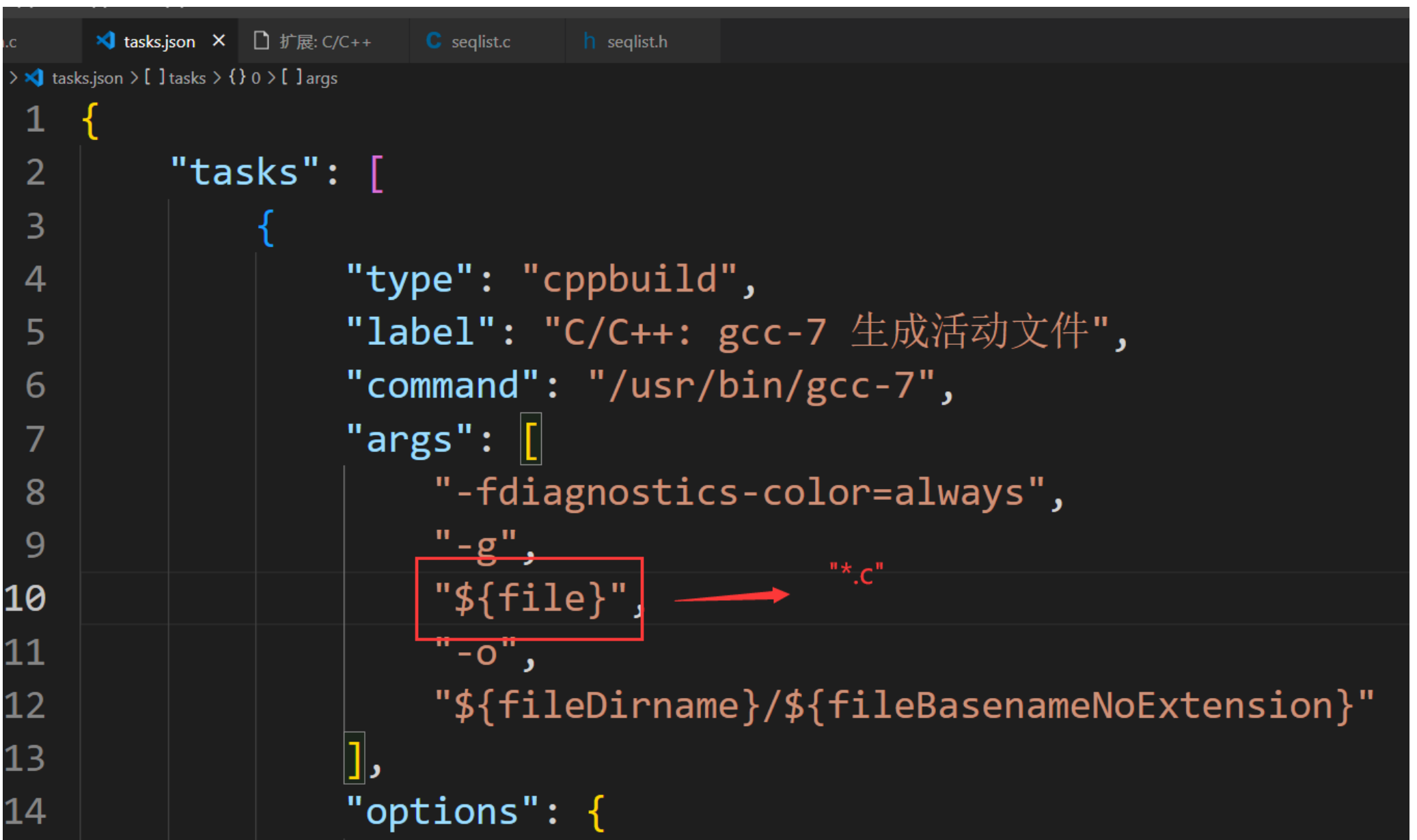
添加的断点

添加的断点

2. 按下调试的快捷键（F5或Fn+F5）



3. 修改配置文件

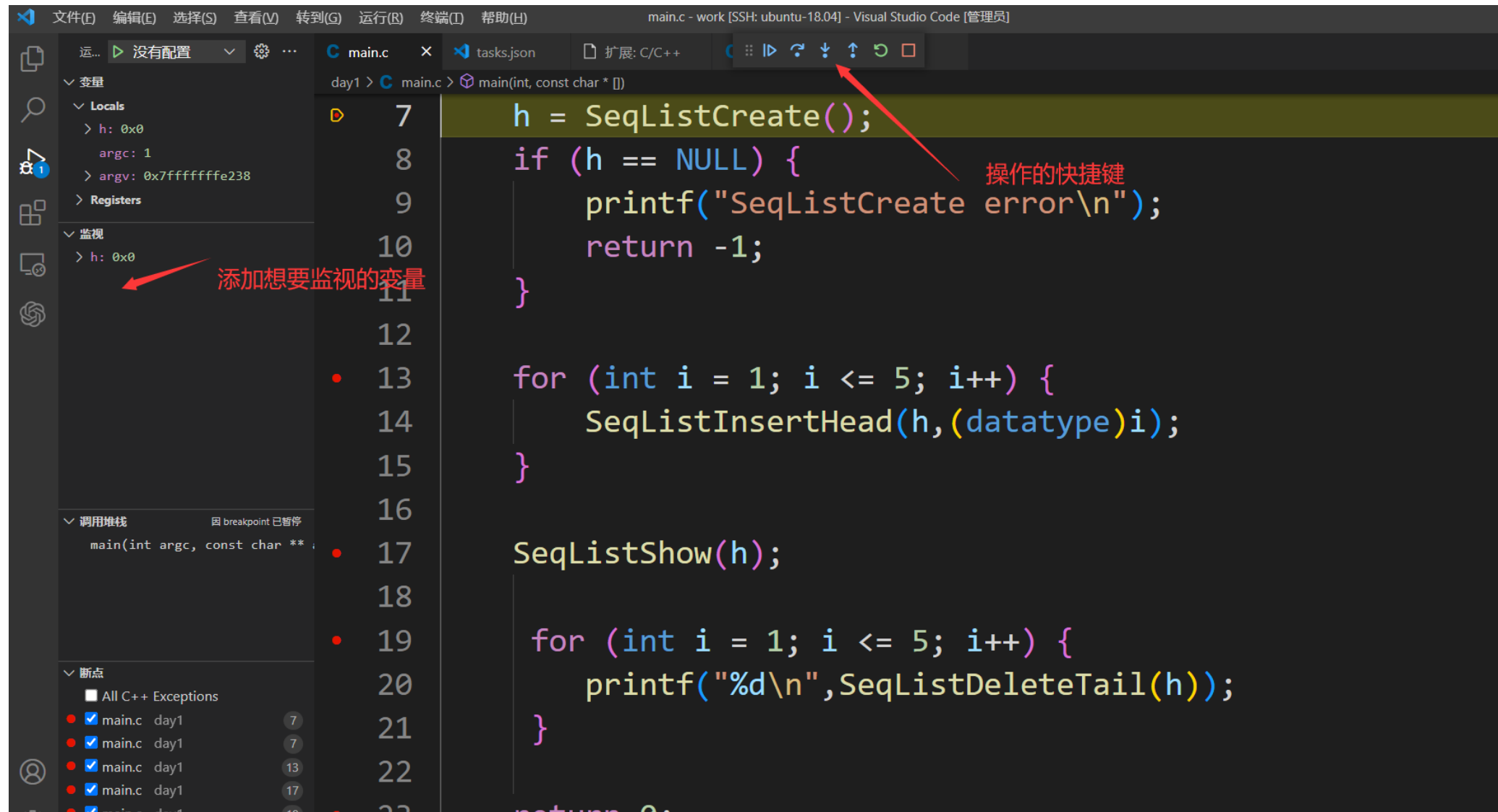


```
1 {
2     "tasks": [
3         {
4             "type": "cppbuild",
5             "label": "C/C++: gcc-7 生成活动文件",
6             "command": "/usr/bin/gcc-7",
7             "args": [
8                 "-fdiagnostics-color=always",
9                 "-g",
10                "${file}",
11                "-o",
12                "${fileDirname}/${fileBasenameNoExtension}"
13            ],
14            "options": {
```

*.c

4. 回到想要调试的main.c文件中重新按下 (F5或者Fn+F5)

5. 出现如下界面



继续

调到下一个

断点位置停下

单步跳过

下一步

退出函数

重启

停止