

My Project

Generated by Doxygen 1.9.5

1 Data Structure Index	1
1.1 Data Structures	1
2 File Index	3
2.1 File List	3
3 Data Structure Documentation	5
3.1 branch Struct Reference	5
3.2 customer Struct Reference	5
3.3 dbf_op Struct Reference	5
3.4 full_sales Struct Reference	6
3.5 head Struct Reference	6
3.6 lease_sales Struct Reference	6
3.7 loan_sales Struct Reference	7
3.8 machine_inst Struct Reference	7
3.9 misc Struct Reference	7
3.10 Node Struct Reference	8
3.10.1 Detailed Description	8
3.10.2 Field Documentation	8
3.10.2.1 letter	8
3.10.2.2 next	8
3.10.2.3 words	8
3.11 NODE Struct Reference	9
3.12 queue Struct Reference	9
3.13 reg_src Struct Reference	9
3.14 sql_op Struct Reference	9
3.15 WordNode Struct Reference	10
3.15.1 Detailed Description	10
3.15.2 Field Documentation	10
3.15.2.1 next	10
3.15.2.2 word	10
4 File Documentation	11
4.1 array.c File Reference	11
4.1.1 Detailed Description	11
4.2 Binary_inversion.c File Reference	12
4.2.1 Detailed Description	12
4.3 bit_segment.c File Reference	12
4.3.1 Detailed Description	13
4.4 copy_n.c File Reference	13
4.4.1 Detailed Description	13
4.5 count_str.c File Reference	14
4.5.1 Detailed Description	14

4.6	cpu_type.h	14
4.7	def.c File Reference	14
4.7.1	Detailed Description	15
4.8	def_b.c File Reference	15
4.8.1	Detailed Description	15
4.9	del_substr.c File Reference	16
4.9.1	Detailed Description	16
4.9.2	Function Documentation	16
4.9.2.1	del_substr()	16
4.10	Digit_group.c File Reference	17
4.10.1	Detailed Description	17
4.11	eight_queen_question.c File Reference	18
4.11.1	Detailed Description	18
4.12	family_age.c File Reference	19
4.12.1	Detailed Description	19
4.13	fgrep.c File Reference	19
4.13.1	Detailed Description	20
4.13.2	Function Documentation	20
4.13.2.1	cmp_str()	20
4.13.2.2	main()	20
4.13.2.3	print_str()	21
4.14	first.c File Reference	21
4.14.1	Detailed Description	22
4.14.2	Function Documentation	22
4.14.2.1	main()	22
4.14.2.2	read_column_numbers()	22
4.14.2.3	rearrange()	23
4.15	function_last.c File Reference	23
4.15.1	Detailed Description	24
4.15.2	Function Documentation	24
4.15.2.1	sort()	24
4.16	gcd.c File Reference	24
4.16.1	Detailed Description	25
4.17	gets_string.c File Reference	25
4.17.1	Detailed Description	26
4.18	identity.c File Reference	26
4.18.1	Detailed Description	26
4.19	key_encode.c File Reference	27
4.19.1	Detailed Description	27
4.19.2	Function Documentation	27
4.19.2.1	prepare_key()	27
4.20	point_seven.c File Reference	28

4.20.1 Detailed Description	29
4.20.2 Function Documentation	29
4.20.2.1 clearIndexList()	29
4.20.2.2 insertWordToList()	29
4.20.2.3 printIndexList()	30
4.21 print.c File Reference	30
4.21.1 Detailed Description	30
4.22 rand.c File Reference	31
4.22.1 Detailed Description	31
4.23 simple_data_structure.c File Reference	32
4.23.1 Detailed Description	32
4.24 singly_linked_node.h	32
4.25 sll_remove.c File Reference	33
4.25.1 Detailed Description	33
4.25.2 Function Documentation	33
4.25.2.1 printList()	33
4.25.2.2 sll_remove()	34
4.26 space_only.c File Reference	34
4.26.1 Detailed Description	35
4.26.2 Function Documentation	35
4.26.2.1 deblank()	35
4.27 store.c File Reference	35
4.27.1 Detailed Description	36
4.27.2 Function Documentation	36
4.27.2.1 store_bit_field()	36
4.28 sum.c File Reference	37
4.28.1 Detailed Description	37
4.29 test_singly_linked_list.c File Reference	37
4.29.1 Detailed Description	38
4.29.2 Function Documentation	38
4.29.2.1 printList()	38
4.30 test_static.c File Reference	38
4.30.1 Detailed Description	39
4.31 thirteen_five.c File Reference	39
4.31.1 Detailed Description	40
Index	41

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

branch	5
customer	5
dbl_op	5
full_sales	6
head	6
lease_sales	6
loan_sales	7
machine_inst	7
misc	7
Node	
一级链表的节点结构体，存储一个字母和一个指向该字母对应的二级链表的指针	8
NODE	9
queue	9
reg_src	9
sql_op	9
WordNode	
二级链表的节点结构体，存储一个单词和指向下一个节点的指针	10

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

array.c	访问伪数组 因为我看不太懂算式，不过后面还是看懂了，但是我先找了其他的。我从CSDN上复制的代码，原本还有另外一个，但是我发现他写的不对，就用了这个。 .	11
Binary_inversion.c	反转二进制数	12
bit_segment.c	编写位段声明，第十章第三题	12
copy_n.c	写一个strncpy函数，作用也相同	13
count_str.c	读取标准输入中的字符，然后计算各类字符所占百分比,不能用if语句	14
cpu_type.h		??
def.c	简单的宏	14
def_b.c	简单宏	15
del_substr.c	实现一个简单的函数，删除字符串的一部分	16
Digit_group.c	操纵位数组，本题的难度在于字符转数字 如果有精力，可以写一个菜单，然后通过菜单来和位数组互动，操纵位数组 我懒得写，所以实现函数后，看看结果就完事了。 . .	17
eight_queen_question.c	八皇后问题	18
family_age.c	读取一个文件，计算每个家族的平均年龄，假设每行十个家族	19
fgrep.c	实现unix里的 fgrap命令，第一个参数是字符串参数	19
first.c	这个程序从标准输入中读取输入行并在标准输出中打印这些输入行 每个输入行的后面一行是该行内容的一部分	21
function_last.c	写个快排，比较简单,13章第4题。	23
gcd.c	求最大公约数	24

gets_string.c	从标准输入读取字符串，并且不需要规定字符串大小,题目要求好像是要拷贝，我这没有，其实也简单，加一个新的动态分配的数组复制过去就行了。	25
identity.c	接受一个10X10的矩阵，并判断是否为单位矩阵	26
key_encode.c	简单的加密程序	27
point_seven.c	把一个单词插入到索引表中，本题是第十二章第七题，也是chatgpt生成 因为我大概知道该怎么写了，但是懒得写测试用例，所以选择了Ai帮忙	28
print.c	实现简单的printf	30
rand.c	这个程序因为编译器优先级不同，会产生不同的结果 根据不同的结果，这个非法表达式，在C和指针中的程序5.3下的表5.2中有。	31
simple_data_structure.c	其实就是个简化版链表	32
singly_linked_node.h	??
sll_remove.c	从链表中移除一个节点	33
space_only.c	如果有多个空格，简化为一个	34
store.c	给定值，储存到一个整数中的指定位数 C和指针第五章第五题	35
sum.c	C和指针第十五章第十题，检验效验和(checksum)	37
test_singly_linked_list.c	本题完全由chatgpt生成	37
test_static.c	测试局部静态变量效果	38
thirteen_five.c	C和指针 第13章第五题 我从网上抄的代码，用chatGPT写的注释和测试用例，这题并不难，问题在于题目翻译的太绕，我理解起来不太容易，直接看代码好多了。	39

Chapter 3

Data Structure Documentation

3.1 branch Struct Reference

Data Fields

- unsigned int **offset**: 8
- unsigned int **opcode**: 8

The documentation for this struct was generated from the following file:

- [bit_segment.c](#)

3.2 customer Struct Reference

Data Fields

- char **customer_name** [20]
- char **customer_address** [40]
- char **model** [20]

The documentation for this struct was generated from the following file:

- [car_sales.c](#)

3.3 dbl_op Struct Reference

Data Fields

- unsigned int **dst_reg**: 3
- unsigned int **dst_mode**: 3
- unsigned int **src_reg**: 3
- unsigned int **src_mode**: 3
- unsigned int **opcode**: 4

The documentation for this struct was generated from the following file:

- [bit_segment.c](#)

3.4 full_sales Struct Reference

Data Fields

- struct [customer](#) **customers**
- float **manufacturer**
- float **actual**
- float **sales_tax**
- float **licensing**

The documentation for this struct was generated from the following file:

- [car_sales.c](#)

3.5 head Struct Reference

Data Fields

- struct [queue](#) * **queue**

The documentation for this struct was generated from the following file:

- [simple_data_structure.c](#)

3.6 lease_sales Struct Reference

Data Fields

- struct [customer](#) **customers**
- float **manufacturer**
- float **actual**
- float **down_payment**
- float **security_deposit**
- float **monthly_payment**
- int **lease_term**

The documentation for this struct was generated from the following file:

- [car_sales.c](#)

3.7 loan_sales Struct Reference

Data Fields

- struct [customer](#) **customers**
- struct [full_sales](#) **loan**
- float **doun_payment**
- int **loan_duration**
- float **interest_rate**
- float **monthly_payment**
- char **name_of_bank** [20]

The documentation for this struct was generated from the following file:

- [car_sales.c](#)

3.8 machine_inst Struct Reference

Data Fields

- ```
union {
 struct misc misc
 struct branch branch
 struct sql_op sql_op
 struct reg_src reg_src
 struct dbl_op dbl_op
};
```

The documentation for this struct was generated from the following file:

- [bit\\_segment.c](#)

## 3.9 misc Struct Reference

### Data Fields

- unsigned int **opcode**: 16

The documentation for this struct was generated from the following file:

- [bit\\_segment.c](#)

## 3.10 Node Struct Reference

一级链表的节点结构体，存储一个字母和一个指向该字母对应的二级链表的指针

### Data Fields

- char `letter`
- struct `WordNode` \* `words`
- struct `Node` \* `next`

### 3.10.1 Detailed Description

一级链表的节点结构体，存储一个字母和一个指向该字母对应的二级链表的指针

### 3.10.2 Field Documentation

#### 3.10.2.1 letter

```
char letter
```

节点存储的字母

#### 3.10.2.2 next

```
struct Node* next
```

指向下一个一级链表节点的指针

#### 3.10.2.3 words

```
struct WordNode* words
```

指向该字母对应的二级链表的指针

The documentation for this struct was generated from the following file:

- [point\\_seven.c](#)

## 3.11 NODE Struct Reference

### Data Fields

- int **data**
- struct [NODE](#) \* **link**

The documentation for this struct was generated from the following file:

- [singly\\_linked\\_node.h](#)

## 3.12 queue Struct Reference

### Data Fields

- size\_t **num**

The documentation for this struct was generated from the following file:

- [simple\\_data\\_structure.c](#)

## 3.13 reg\_src Struct Reference

### Data Fields

- unsigned int **dst\_reg**: 3
- unsigned int **dst\_mode**: 3
- unsigned int **src\_reg**: 3
- unsigned int **opcode**: 7

The documentation for this struct was generated from the following file:

- [bit\\_segment.c](#)

## 3.14 sql\_op Struct Reference

### Data Fields

- unsigned int **dst\_reg**: 3
- unsigned int **dst\_mode**: 3
- unsigned int **opcode**: 10

The documentation for this struct was generated from the following file:

- [bit\\_segment.c](#)

## 3.15 WordNode Struct Reference

二级链表的节点结构体，存储一个单词和指向下一个节点的指针

### Data Fields

- char \* [word](#)
- struct [WordNode](#) \* [next](#)

### 3.15.1 Detailed Description

二级链表的节点结构体，存储一个单词和指向下一个节点的指针

### 3.15.2 Field Documentation

#### 3.15.2.1 next

```
struct WordNode* next
```

指向下一个二级链表节点的指针

#### 3.15.2.2 word

```
char* word
```

节点存储的单词

The documentation for this struct was generated from the following file:

- [point\\_seven.c](#)



## Chapter 4

# File Documentation

### 4.1 array.c File Reference

访问伪数组 因为我看不太懂算式，不过后面还是看懂了，但是我先找了其他的。我从CSDN上复制的代码，原本还有另外一个，但是我发现他写的不对，就用了这个。

```
#include <stdio.h>
#include <stdarg.h>
```

#### Macros

- `#define MAX_DIM 10`

#### Functions

- `int array_offset (int arrayinfo[],...)`
- `int compute_offset (int dim, int *arrayinfo, int *arg, unsigned int *weight)`
- `int main (void)`
- `int array_offset (int *arrayinfo,...)`

#### 4.1.1 Detailed Description

访问伪数组 因为我看不太懂算式，不过后面还是看懂了，但是我先找了其他的。我从CSDN上复制的代码，原本还有另外一个，但是我发现他写的不对，就用了这个。

#### Author

your name ( [you@domain.com](mailto:you@domain.com))

#### Version

0.1

#### Date

2023-02-26

#### Copyright

Copyright (c) 2023

## 4.2 Binary\_inversion.c File Reference

反转二进制数

```
#include <stdio.h>
#include <stdlib.h>
```

### Functions

- unsigned int **reverse\_bits** (unsigned int value)
- int **main** ()

### 4.2.1 Detailed Description

反转二进制数

#### Author

your name ( [you@domain.com](mailto:you@domain.com) )

#### Version

0.1

#### Date

2023-02-11

#### Copyright

Copyright (c) 2023

## 4.3 bit\_segment.c File Reference

编写位段声明，第十章第三题

```
#include <stdio.h>
```

### Data Structures

- struct [misc](#)
- struct [branch](#)
- struct [sql\\_op](#)
- struct [reg\\_src](#)
- struct [dbl\\_op](#)
- struct [machine\\_inst](#)

## Functions

- `int main ()`

### 4.3.1 Detailed Description

编写位段声明，第十章第三题

#### Author

your name ( [you@domain.com](mailto:you@domain.com) )

#### Version

0.1

#### Date

2023-03-11

#### Copyright

Copyright (c) 2023

## 4.4 copy\_n.c File Reference

写一个strncpy函数，作用也相同

```
#include <stdio.h>
```

## Functions

- `void copy_n (char *dst, char *src, int n)`  
如果不用`strlen`如何判断`src`的字符串长度？如果用`!0`判断，可能会有垃圾值，但是默认`src`输入的是一个字符串。那么剩下的问题，是程序员该预防的，那就是进行正确的输入。
- `int main (void)`

### 4.4.1 Detailed Description

写一个strncpy函数，作用也相同

#### Author

your name ( [you@domain.com](mailto:you@domain.com) )

#### Version

0.1

#### Date

2023-02-07

#### Copyright

Copyright (c) 2023

## 4.5 count\_str.c File Reference

读取标准输入中的字符，然后计算各类字符所占百分比,不能用if语句

```
#include <stdio.h>
#include <ctype.h>
```

### Functions

- int **main** ()

### 4.5.1 Detailed Description

读取标准输入中的字符，然后计算各类字符所占百分比,不能用if语句

#### Author

your name ( [you@domain.com](mailto:you@domain.com))

#### Version

0.1

#### Date

2023-03-15

#### Copyright

Copyright (c) 2023

## 4.6 cpu\_type.h

```
1
2 #define CPU_VAX
3 #define CPU_68000
4 #define CPU_68020
5 #define CPU_80386
6 #define CPU_6809
7 #define CP_6502
8 #define CPU_3B2
9 #define CPU_UNKNOWN
```

## 4.7 def.c File Reference

简单的宏

```
#include <stdio.h>
```

## Functions

- void **print\_ledger** (int num)
- void **print\_ledger\_long** (int num)
- void **print\_ledger\_detailed** (int num) void print\_ledger\_default(int num)

### 4.7.1 Detailed Description

简单的宏

#### Author

your name ( [you@domain.com](mailto:you@domain.com))

#### Version

0.1

#### Date

2023-03-18

#### Copyright

Copyright (c) 2023

## 4.8 def\_b.c File Reference

简单宏

```
#include <stdio.h>
#include "cpu_type.h"
```

## Functions

- int **cpu\_type** ()

### 4.8.1 Detailed Description

简单宏

#### Author

your name ( [you@domain.com](mailto:you@domain.com))

#### Version

0.1

#### Date

2023-03-18

#### Copyright

Copyright (c) 2023

## 4.9 del\_substr.c File Reference

实现一个简单的函数，删除字符串的一部分

```
#include <stdio.h>
```

### Functions

- int `del_substr` (char \*str, char const \*substr)  
我觉得我这个函数写的又精简，又棒，可以说是现在位置我想出来的最棒的写法了
- int `main` ()

### 4.9.1 Detailed Description

实现一个简单的函数，删除字符串的一部分

#### Author

your name ( `you@domain.com` )

#### Version

0.1

#### Date

2023-02-13

#### Copyright

Copyright (c) 2023

### 4.9.2 Function Documentation

#### 4.9.2.1 del\_substr()

```
int del_substr (
 char * str,
 char const * substr)
```

我觉得我这个函数写的又精简，又棒，可以说是现在位置我想出来的最棒的写法了

## Parameters

|               |         |
|---------------|---------|
| <i>str</i>    | 原字符串    |
| <i>substr</i> | 要删除的字符串 |

## Returns

int

## 4.10 Digit\_group.c File Reference

操纵位数组，本题的难度在于字符转数字 如果有精力，可以写一个菜单，然后通过菜单来和位数组互动，操纵位数组 我懒得写，所以实现函数后，看看结果就完事了。

```
#include <stdio.h>
```

## Functions

- void **set\_bit** (char bit\_array[], unsigned int number)
- void **clear\_bit** (char bit\_array[], unsigned int number)
- void **assing\_bit** (char bit\_array[], unsigned int number, int value)
- int **test\_bit** (char bit\_array[], unsigned int number)
- int **main** ()

### 4.10.1 Detailed Description

操纵位数组，本题的难度在于字符转数字 如果有精力，可以写一个菜单，然后通过菜单来和位数组互动，操纵位数组 我懒得写，所以实现函数后，看看结果就完事了。

## Author

your name ( [you@domain.com](mailto:you@domain.com) )

## Version

0.1

## Date

2023-02-11

## Copyright

Copyright (c) 2023

## 4.11 eight\_queen\_question.c File Reference

八皇后问题

```
#include <stdio.h>
```

### Macros

- `#define ROW 8`
- `#define COL 8`

### Functions

- void **print\_board** ()
- int **conflicts** (int row, int column)
- void **place\_queen** (int row)
- int **main** ()

### Variables

- int **board** [ROW][COL]

#### 4.11.1 Detailed Description

八皇后问题

#### Author

your name ( [you@domain.com](mailto:you@domain.com) )

#### Version

0.1

#### Date

2023-02-26

#### Copyright

Copyright (c) 2023



## 4.12 family\_age.c File Reference

读取一个文件，计算每个家族的平均年龄，假设每行十个家族

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
```

### Functions

- `int main ()`

#### 4.12.1 Detailed Description

读取一个文件，计算每个家族的平均年龄，假设每行十个家族

##### Author

your name ( [you@domain.com](mailto:you@domain.com) )

##### Version

0.1

##### Date

2023-03-20

##### Copyright

Copyright (c) 2023

## 4.13 fgrep.c File Reference

实现unix里的 fgrep命令，第一个参数是字符串参数

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <ctype.h>
```

### Functions

- `int cmp_str (char *src, char *dst)`
- `void print_str (char *str, char *filename)`  
打印文件中匹配的字符串的行号
- `int main (int argc, char *argv[])`  
主函数，接受命令行参数并调用其他函数

### 4.13.1 Detailed Description

实现unix里的 `tgrap`命令，第一个参数是字符串参数

#### Author

your name ( `you@domain.com` )

#### Version

0.1

#### Date

2023-04-19

#### Copyright

Copyright (c) 2023

### 4.13.2 Function Documentation

#### 4.13.2.1 `cmp_str()`

```
int cmp_str (
 char * src,
 char * dst)
```

#### Parameters

|            |         |
|------------|---------|
| <i>src</i> | 源字符串    |
| <i>dst</i> | 要对比的字符串 |

#### Returns

int 如果匹配则返回1，否则返回0

#### 4.13.2.2 `main()`

```
int main (
 int argc,
 char * argv[])
```

主函数，接受命令行参数并调用其他函数

**Parameters**

|             |         |
|-------------|---------|
| <i>argc</i> | 命令行参数数量 |
| <i>argv</i> | 命令行参数列表 |

**Returns**

int 返回程序执行状态

**4.13.2.3 print\_str()**

```
void print_str (
 char * str,
 char * filename)
```

打印文件中匹配的字符串的行号

**Parameters**

|                 |         |
|-----------------|---------|
| <i>str</i>      | 要查找的字符串 |
| <i>filename</i> | 文件名     |

**4.14 first.c File Reference**

这个程序从标准输入中读取输入行并在标准输出中打印这些输入行 每个输入行的后面一行是该行内容的一部分

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

**Macros**

- `#define MAX_COLS 20`
- `#define MAX_INPUT 1000`

**Functions**

- int [read\\_column\\_numbers](#) (int columns[], int max)  
读取列标号，如果超出规定范围则不予理会
- void [rearrange](#) (char \*output, char const \*input, int n\_columns, int const columns[])
- int [main](#) (void)

### 4.14.1 Detailed Description

这个程序从标准输入中读取输入行并在标准输出中打印这些输入行 每个输入行的后面一行是该行内容的一部分

#### Author

your name ( [you@domain.com](mailto:you@domain.com) )

输入第一行是一串列标号，串的最后一个是负数负责结尾 这些列标号成对出现，说明需要打印的输入行的列的范围 例如，0 3 10 12 -1表示第0列到第三列，第十列到第12列的内容将被打印 上面的内容是第一行需要输入的 翻译的太差了，我看了程序重新说一下这个程序的这几个参数的意思。0 3 是打印0-3列，然后第二组10 12 是打印10-12列，-1是表示列标号获取结束。所以在rearrange()函数里，进行了一个字符串拼接，用于输出

#### Version

0.1

#### Date

2022-11-09

#### Copyright

Copyright (c) 2022

### 4.14.2 Function Documentation

#### 4.14.2.1 main()

```
int main (
 void)
```

读取该串列标号

读取、处理和打印剩余的输入行

#### 4.14.2.2 read\_column\_numbers()

```
int read_column_numbers (
 int columns[],
 int max)
```

读取列标号，如果超出规定范围则不予理会

## Parameters

|                |  |
|----------------|--|
| <i>columns</i> |  |
| <i>max</i>     |  |

## Returns

int

取得列标号，如果所读取的数小于0则停止

确认已经读取的标号为偶数个，因为他们是以对的形式出现的

丢弃该行中包含最后一个数字的那部分内容

## 4.14.2.3 rearrange()

```
void rearrange (
 char * output,
 char const * input,
 int n_columns,
 int const columns[])
```

处理每对列标号

如果输入行结束或输出行已满，就结束任务

如果输出行数据空间不够，只复制可以容纳的数据

复制相关数据

## 4.15 function\_last.c File Reference

写个快排，比较简单,13章第4题。

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
```

## Functions

- void **sort** (void \*str, int count, size\_t len, int(\*cmp)(void \*, void \*, size\_t))  
排序函数
- int **compare** (void \*dst, void \*src, size\_t leng)
- int **main** ()

### 4.15.1 Detailed Description

写个快排，比较简单,13章第4题。

#### Author

your name ( [you@domain.com](mailto:you@domain.com) )

#### Version

0.1

#### Date

2023-03-15

#### Copyright

Copyright (c) 2023

### 4.15.2 Function Documentation

#### 4.15.2.1 `sort()`

```
void sort (
 void * str,
 int count,
 size_t len,
 int (*)(void *, void *, size_t) cmp)
```

排序函数

#### Parameters

|              |            |
|--------------|------------|
| <i>str</i>   | 排序数组的第一个指针 |
| <i>count</i> | 数组中值的个数    |
| <i>len</i>   | 数组中元素长度    |
| <i>cmp</i>   | 回调函数用于比较   |

## 4.16 `gcd.c` File Reference

求最大公约数

```
#include <stdio.h>
```

## Functions

- int **gcd** (int m, int n)
- int **main** ()

### 4.16.1 Detailed Description

求最大公约数

#### Author

your name ( [you@domain.com](mailto:you@domain.com))

#### Version

0.1

#### Date

2023-02-13

#### Copyright

Copyright (c) 2023

## 4.17 gets\_string.c File Reference

从标准输入读取字符串，并且不需要规定字符串大小,题目要求好像是要拷贝，我这没有，其实也简单，加一个新的动态分配的数组复制过去就行了。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

## Functions

- char \* **save\_str** ()
- int **main** ()

### 4.17.1 Detailed Description

从标准输入读取字符串，并且不需要规定字符串大小,题目要求好像是要拷贝，我这没有，其实也简单，加一个新的动态分配的数组复制过去就行了。

#### Author

your name ( [you@domain.com](mailto:you@domain.com) )

#### Version

0.1

#### Date

2023-03-12

#### Copyright

Copyright (c) 2023

## 4.18 identity.c File Reference

接受一个10X10的矩阵，并判断是否为单位矩阵

```
#include <stdio.h>
#include <stdbool.h>
```

### Functions

- `bool identity_matrix (int matrix[ ][10])`
- `int main (void)`

### 4.18.1 Detailed Description

接受一个10X10的矩阵，并判断是否为单位矩阵

接受一个任意矩阵，并判断是否为单位矩阵

#### Author

your name ( [you@domain.com](mailto:you@domain.com) )

#### Version

0.1

#### Date

2023-02-25

#### Copyright

Copyright (c) 2023



## 4.19 key\_encode.c File Reference

简单的加密程序

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
```

### Functions

- int `prepare_key` (char \*key)  
生成秘钥表
- void `encrypt` (char \*data, char const \*key)
- void `decrypt` (char \*data, char const \*key)
- int `main` ()

### 4.19.1 Detailed Description

简单的加密程序

#### Author

your name ( `you@domain.com` )

#### Version

0.1

#### Date

2023-03-08

#### Copyright

Copyright (c) 2023

### 4.19.2 Function Documentation

#### 4.19.2.1 prepare\_key()

```
int prepare_key (
 char * key)
```

生成秘钥表

#### Parameters

|            |    |
|------------|----|
| <i>key</i> | 单词 |
|------------|----|

#### Returns

返回是否成功

## 4.20 point\_seven.c File Reference

把一个单词插入到索引表中，本题是第十二章第七题，也是chatgpt生成 因为我大概知道该怎么写了，但是懒得写测试用例，所以选择了Ai帮忙

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <stdbool.h>
```

### Data Structures

- struct [WordNode](#)  
二级链表的节点结构体，存储一个单词和指向下一个节点的指针
- struct [Node](#)  
一级链表的节点结构体，存储一个字母和一个指向该字母对应的二级链表的指针

### Typedefs

- typedef struct [Node](#) **Node**  
一级链表的节点结构体，存储一个字母和一个指向该字母对应的二级链表的指针

### Functions

- void [printIndexList](#) ([Node](#) \*index)
- void [clearIndexList](#) ([Node](#) \*\*index)  
清空索引表
- bool [insertWordToList](#) ([Node](#) \*\*list, char \*word)  
用于插入单词到索引表中
- int **main** ()

### 4.20.1 Detailed Description

把一个单词插入到索引表中，本题是第十二章第七题，也是chatgpt生成 因为我大概知道该怎么写了，但是懒得写测试用例，所以选择了Ai帮忙

#### Author

your name ( [you@domain.com](mailto:you@domain.com) )

我发现Ai写的有点问题，可能是复制的原因，我明天重新写过，还是不能偷懒

#### Version

0.1

#### Date

2023-03-13

#### Copyright

Copyright (c) 2023

### 4.20.2 Function Documentation

#### 4.20.2.1 clearIndexList()

```
void clearIndexList (
 Node ** index)
```

清空索引表

#### Parameters

|              |           |
|--------------|-----------|
| <i>index</i> | 指向一级链表的指针 |
|--------------|-----------|

#### 4.20.2.2 insertWordToList()

```
bool insertWordToList (
 Node ** list,
 char * word)
```

用于插入单词到索引表中

**Parameters**

|             |                  |
|-------------|------------------|
| <i>list</i> | 一个指向指向一级链表的指针的指针 |
| <i>word</i> | 要插入的单词字符串        |

**Returns**

`true` 插入成功  
`false` 插入失败

**4.20.2.3 printIndexList()**

```
void printIndexList (
 Node * index)
```

打印索引表中所有单词的信息

**Parameters**

|              |                            |
|--------------|----------------------------|
| <i>index</i> | 索引表的头指针 打印索引表中的所有单词及其对应的页码 |
| <i>index</i> | 索引表                        |

**4.21 print.c File Reference**

实现简单的printf

```
#include <stdio.h>
#include <stdarg.h>
```

**Functions**

- int **print** (char \*format,...)
- void **print\_float** (float num)
- void **print\_integer** (int num)
- int **main** ()

**4.21.1 Detailed Description**

实现简单的printf

**Author**

your name ( [you@domain.com](mailto:you@domain.com) )

**Version**

0.1

**Date**

2023-02-14

**Copyright**

Copyright (c) 2023

## 4.22 rand.c File Reference

这个程序因为编译器优先级不同，会产生不同的结果 根据不同的结果，这个非法表达式，在C和指针中的程序5.3下的表5.2中有。

```
#include <stdio.h>
```

### Functions

- `int main ()`

#### 4.22.1 Detailed Description

这个程序因为编译器优先级不同，会产生不同的结果 根据不同的结果，这个非法表达式，在C和指针中的程序5.3下的表5.2中有。

**Author**

your name ( [you@domain.com](mailto:you@domain.com) )

**Version**

0.1

**Date**

2023-02-08

**Copyright**

Copyright (c) 2023

## 4.23 simple\_data\_structure.c File Reference

其实就是个简化版链表

```
#include <stdio.h>
#include <stdlib.h>
```

### Data Structures

- struct [queue](#)
- struct [head](#)

### Functions

- int [main](#) ()

#### 4.23.1 Detailed Description

其实就是个简化版链表

#### Author

your name ( [you@domain.com](mailto:you@domain.com) )

#### Version

0.1

#### Date

2023-03-12

#### Copyright

Copyright (c) 2023

## 4.24 singly\_linked\_node.h

```
1 #ifndef SINGLY_LINKED_NODE_H
2 #define SINGLY_LINKED_NODE_H
3
4 struct NODE {
5 int data;
6 struct NODE* link;
7 };
8
9
10 #endif // SINGLY_LINKED_NODE_H
```

## 4.25 `sll_remove.c` File Reference

从链表中移除一个节点

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include "singly_linked_node.h"
```

### Functions

- `bool sll_remove` (struct `NODE` \*\*rootp, struct `NODE` \*node)  
删除一个节点
- `void printList` (struct `NODE` \*first)  
输出单链表的所有节点的值
- `int main` ()

### 4.25.1 Detailed Description

从链表中移除一个节点

#### Author

your name ( `you@domain.com` )

#### Version

0.1

#### Date

2023-03-13

#### Copyright

Copyright (c) 2023

### 4.25.2 Function Documentation

#### 4.25.2.1 `printList()`

```
void printList (
 struct NODE * first)
```

输出单链表的所有节点的值

**Parameters**

|              |            |
|--------------|------------|
| <i>first</i> | 链表的第一个节点指针 |
|--------------|------------|

**Parameters**

|              |            |
|--------------|------------|
| <i>first</i> | 链表的第一个节点指针 |
|--------------|------------|

**4.25.2.2 sll\_remove()**

```
bool sll_remove (
 struct NODE ** rootp,
 struct NODE * node)
```

删除一个节点

**Parameters**

|              |           |
|--------------|-----------|
| <i>rootp</i> | 根指针       |
| <i>node</i>  | 需要删除的节点指针 |

**Returns**

int

**4.26 space\_only.c File Reference**

如果有多个空格，简化为一个

```
#include <stdio.h>
```

**Functions**

- void **deblank** (char string[ ])

这个函数，因为没有返回值，所以有两种做法，一种是用新的数组，不改变原数组，输出一个修改后的。另一个做法是，直接在原数组上操作。但是数组的插入和删除很不方便，删了中间的空格，要偏移后面的数据。所以我用第一种做法。

- int **main** (void)



### 4.26.1 Detailed Description

如果有多个空格，简化为一个

#### Author

your name ( [you@domain.com](mailto:you@domain.com) )

#### Version

0.1

#### Date

2023-02-07

#### Copyright

Copyright (c) 2023

### 4.26.2 Function Documentation

#### 4.26.2.1 deblank()

```
void deblank (
 char string[])
```

这个函数，因为没有返回值，所以有两种做法，一种是用新的数组，不改变原数组，输出一个修改后的。另一个做法是，直接在原数组上操作。但是数组的插入和删除很不方便，删了中间的空格，要偏移后面的数据。所以我用第一种做法。

#### Parameters

|               |  |
|---------------|--|
| <i>string</i> |  |
|---------------|--|

## 4.27 store.c File Reference

给定值，储存到一个整数中的指定位数 C和指针第五章第五题

```
#include <stdio.h>
#include <stdlib.h>
```

## Functions

- int `store_bit_field` (int original\_value, int value\_to\_store, unsigned starting\_bit, unsigned ending\_bit)
- int `main` ()

### 4.27.1 Detailed Description

给定值，储存到一个整数中的指定位数 C和指针第五章第五题

#### Author

your name ( `you@domain.com` )

#### Version

0.1

#### Date

2023-02-11

#### Copyright

Copyright (c) 2023

### 4.27.2 Function Documentation

#### 4.27.2.1 store\_bit\_field()

```
int store_bit_field (
 int original_value,
 int value_to_store,
 unsigned starting_bit,
 unsigned ending_bit)
```

#### Parameters

|                       |        |
|-----------------------|--------|
| <i>original_value</i> | 原始值    |
| <i>value_to_store</i> | 需要储存的值 |
| <i>starting_bit</i>   | 起始位    |
| <i>ending_bit</i>     | 结束位    |

**Returns**

int 返回值

## 4.28 sum.c File Reference

C和指针第十五章第十题，检验效验和(cheeksum)

**Functions**

- int **main** (int argc, char \*\*argv)

### 4.28.1 Detailed Description

C和指针第十五章第十题，检验效验和(cheeksum)

**Author**

your name ( [you@domain.com](mailto:you@domain.com))

**Version**

0.1

**Date**

2023-04-19

**Copyright**

Copyright (c) 2023

## 4.29 test\_singly\_linked\_list.c File Reference

本题完全由chatgpt生成

```
#include <stdio.h>
#include <stdlib.h>
#include "singly_linked_node.h"
```

**Functions**

- void **printList** (struct **NODE** \*first)  
输出单链表的所有节点的值
- int **main** ()

### 4.29.1 Detailed Description

本题完全由chatgpt生成

#### Author

your name ( [you@domain.com](mailto:you@domain.com) )

#### Version

0.1

#### Date

2023-03-13

#### Copyright

Copyright (c) 2023

### 4.29.2 Function Documentation

#### 4.29.2.1 printList()

```
void printList (
 struct NODE * first)
```

输出单链表的所有节点的值

#### Parameters

|              |            |
|--------------|------------|
| <i>first</i> | 链表的第一个节点指针 |
|--------------|------------|

## 4.30 test\_static.c File Reference

测试局部静态变量效果

```
#include <stdio.h>
```

### Functions

- `int main ()`

### 4.30.1 Detailed Description

测试局部静态变量效果

#### Author

your name ( [you@domain.com](mailto:you@domain.com) )

#### Version

0.1

#### Date

2023-04-19

#### Copyright

Copyright (c) 2023

## 4.31 thirteen\_five.c File Reference

C和指针 第13章第五题 我从网上抄的代码，用chatGPT写的注释和测试用例，这题并不难，问题在于题目翻译的太绕，我理解起来不太容易，直接看代码好多了。

```
#include <stdio.h>
#include <stdlib.h>
```

### Enumerations

- enum { **NONE** , **FLAG** , **ARG** }

### Functions

- int **argtype** (register int ch, register int \*control)
- char \*\* **do\_args** (int argc, char \*\*argv, char \*control, void(\*do\_arg)(int ch, char \*value), void(\*illegal\_arg)(int ch))
- void **print\_flag** (int ch, char \*value)
- void **print\_arg** (int ch, char \*value)
- void **illegal\_arg** (int ch)
- int **main** (int argc, char \*\*argv)

### 4.31.1 Detailed Description

C和指针 第13章第五题 我从网上抄的代码，用chatGPT写的注释和测试用例，这题并不难，问题在于题目翻译的太绕，我理解起来不太容易，直接看代码好多了。

**Author**

your name ( [you@domain.com](mailto:you@domain.com) )

**Version**

0.1

**Date**

2023-03-17

**Copyright**

Copyright (c) 2023

# Index

array.c, [11](#)

Binary\_inversion.c, [12](#)

bit\_segment.c, [12](#)

branch, [5](#)

clearIndexList  
    point\_seven.c, [29](#)

cmp\_str  
    fgrep.c, [20](#)

copy\_n.c, [13](#)

count\_str.c, [14](#)

customer, [5](#)

dbl\_op, [5](#)

deblank  
    space\_only.c, [35](#)

def.c, [14](#)

def\_b.c, [15](#)

del\_substr  
    del\_substr.c, [16](#)

del\_substr.c, [16](#)  
    del\_substr, [16](#)

Digit\_group.c, [17](#)

eight\_queen\_question.c, [18](#)

family\_age.c, [19](#)

fgrep.c, [19](#)  
    cmp\_str, [20](#)  
    main, [20](#)  
    print\_str, [21](#)

first.c, [21](#)  
    main, [22](#)  
    read\_column\_numbers, [22](#)  
    rearrange, [23](#)

full\_sales, [6](#)

function\_last.c, [23](#)  
    sort, [24](#)

gcd.c, [24](#)

gets\_string.c, [25](#)

head, [6](#)

identity.c, [26](#)

insertWordToList  
    point\_seven.c, [29](#)

key\_encode.c, [27](#)  
    prepare\_key, [27](#)

lease\_sales, [6](#)

letter  
    Node, [8](#)

loan\_sales, [7](#)

machine\_inst, [7](#)

main  
    fgrep.c, [20](#)  
    first.c, [22](#)

misc, [7](#)

next  
    Node, [8](#)  
    WordNode, [10](#)

NODE, [9](#)

Node, [8](#)  
    letter, [8](#)  
    next, [8](#)  
    words, [8](#)

point\_seven.c, [28](#)  
    clearIndexList, [29](#)  
    insertWordToList, [29](#)  
    printIndexList, [30](#)

prepare\_key  
    key\_encode.c, [27](#)

print.c, [30](#)

print\_str  
    fgrep.c, [21](#)

printIndexList  
    point\_seven.c, [30](#)

printList  
    sll\_remove.c, [33](#)  
    test\_singly\_linked\_list.c, [38](#)

queue, [9](#)

rand.c, [31](#)

read\_column\_numbers  
    first.c, [22](#)

rearrange  
    first.c, [23](#)

reg\_src, [9](#)

simple\_data\_structure.c, [32](#)

sll\_remove  
    sll\_remove.c, [34](#)

sll\_remove.c, [33](#)  
    printList, [33](#)  
    sll\_remove, [34](#)

sort

- function\_last.c, [24](#)
- space\_only.c, [34](#)
  - deblank, [35](#)
- sql\_op, [9](#)
- store.c, [35](#)
  - store\_bit\_field, [36](#)
- store\_bit\_field
  - store.c, [36](#)
- sum.c, [37](#)
  
- test\_singly\_linked\_list.c, [37](#)
  - printList, [38](#)
- test\_static.c, [38](#)
- thirteen\_five.c, [39](#)
  
- word
  - WordNode, [10](#)
- WordNode, [10](#)
  - next, [10](#)
  - word, [10](#)
- words
  - Node, [8](#)