数据

char 转 int

```
char a = '0';
int ia = a - '0';

int iia = atoi(a); //stdlib.h
```

string 转int

```
int stoi(string str);
```

判断是否是int

int 转 string

```
int a = 10;
string str_a = to_string(a);
```

字符串

搜索字符串

`find`接口，没查找到，返回`string::npos`

```
str.find(goal) != string::npos;
```

字符串扩容

```
// 将原来的字符串扩容到n的大小
string.resize(n)
```

子字符串

取从pos开始，长度为length的子字符串

```
str.substr(pos,length);
```

队列 deque

STL提供了双向队列，能够从头部和尾部取出元素

- 访问队头元素

```
item = deque.front()
```

- 弹出队头元素

```
deque.pop_front()
```

- 添加元素

```
deque.push_back(item)
```

字典 unordered_map

遍历

- 迭代器

```
unordered_map<string, string> umap;
for (auto iter = umap.begin(); iter != umap.end(); ++iter) {
    cout << "<" << iter->first << ", " << iter->second << ">" << endl;
}
```

- 判断某个key是否存在

```
unordered_map<char, vector<char>> edges;
// 判断 某个key是否存在，不存在的话，创建一个vector数组作为value
if (!edges.count(key)) {
    edges[key] = vector<char>();
}

edges.find(key) != edges.end()//查
```

数组 vector

判断是否为空

```
vector<int>& arr;
if(arr.empty()){

}
```

反转

```
vector<int>& arr;
reverse(arr.begin(), arr.begin() + n);
```

排序

- 按照字符串长度和字母序排序

```
vector<string>& words;
sort(words.begin(), words.end(), [](const string &a, const string &b) {
  // 如果两个字符串长度不相等，返回 更长的字符串；
  // 如果两个字符串长度相等，返回 字母序靠前的字符串
  return a.size() != b.size() ? a.size() < b.size() : a > b;
});
```

- 按照某个算法排序，例如按照两者组合后的结果顺序排序

```
 vector<string> strs;
 sort(strs.begin(),strs.end(),compare);
 static bool compare(const string &a,const string &b)
{
    return a+b<b+a;
}
```

求和`accumulate`

```
#include<numeric>

vector<int> account;
accumulate(account.begin(), account.end(), 0)
```

复制

从一个数组复制到另外一个数组

```
vector<int> A
vector<int> B = vector<int>(A.begin(),A.begin() + offset);
vector<int> C = vector<int>(A.begin(),A.end());
```

最大最小值

```
#include <vector>
#include <algorithm>

vector<double> s;
double min = *min_element(s.begin(), s.end());//返回s中的最小值
double max= *max_element(s.begin(), s.end());//返回最大值
```

数学

- `__builtin_popcount(x)`

精确计算二进制中1的个数