1 Qt Creator 推荐设置

考虑到 Qt 的默认设置比较恼人，为了更加高效简便的完成课题，这里给出一套推荐设置。

Build & Run pane

- On General tab
  - Save all files before build ☑
  - top applications before building = All
  - Choose All to end the running program when re-building.
- On Compile Output tab
  - Open Compile Output when building ☑
- On Application Output tab
- Clear old output on a new run ☑

Debugger pane

- Debugger font size follows main editor ☑
- Switch to previous mode on debugger exit ☑

Disable style analyzer

- on Analyzer pane
  - Analyze open files ✗
- on C++ pane
  - Use clangd ✗

2 CS106

Collections

这里给出 CS106 这个库的容器接口，免得考场上想不起来某个方法是怎么用的。

```
class string {
  bool empty() const;                          // O(1)
  int size() const;                            // O(1)
  int find(char ch) const;                     // O(N)
  int find(char ch, int start) const;          // O(N)
  string substr(int start) const;              // O(N)
  string substr(int start, int length) const; // O(N)
  char &operator[](int index);                 // O(1)
  const char &operator[](int index) const;     // O(1)
};
class Vector {
  bool isEmpty() const;                    // O(1)
  int size() const;                        // O(1)
  void add(const Type &elem);              // operator+= used
similarly — O(1)
  void insert(int pos, const Type &elem); // O(N)
  void remove(int pos);                    // O(N)
  Type &operator[](int pos);               // O(1)
};
class Grid {
  int numRows() const;                     // O(1)
  int numCols() const;                     // O(1)
  bool inBounds(int row, int col) const; // O(1)
  Type get(int row, int col) const;        // or operator [][] also
works — O(1)
  void set(int row, int col, const Type &elem); // O(1)
};
class Stack {
  bool isEmpty() const;        // O(1)
  void push(const Type &elem); // O(1)
  Type pop();                  // O(1)
};
class Queue {
  bool isEmpty() const;           // O(1)
  void enqueue(const Type &elem); // O(1)
  Type dequeue();                 // O(1)
};
class Map {
  bool isEmpty() const;                          // O(1)
  int size() const;                              // O(1)
  void put(const Key &key, const Value &value); // O(logN)
  bool containsKey(const Key &key) const;        // O(logN)
  Value get(const Key &key) const;               // O(logN)
  Value &operator[](const Key &key);             // O(logN)
};
// Example for loop: for (Key key : mymap){…}
class HashMap {
  bool isEmpty() const;                          // O(1)
  int size() const;                              // O(1)
  void put(const Key &key, const Value &value); // O(1)
  bool containsKey(const Key &key) const;        // O(1)
  Value get(const Key &key) const;               // O(1)
  Value &operator[](const Key &key);             // O(1)
};
// Example for loop: for (Key key : mymap){…}
class Set {
  bool isEmpty() const;       // O(1)
  int size() const;           // O(1)
  void add(const Type &elem); // operator+= also adds elements —
O(logN)
  bool contains(const Type &elem) const; // O(logN)
  void remove(ValueType value);          // O(logN)
};
// Example for loop: for (Type elem : mymap){…}
class Lexicon {
  int size() const;                      // O(1)
  bool isEmpty() const;                  // O(1)
  void clear();                          // O(N)
  void add(string word);                 // O(W) where W is
word.length()
  bool contains(string word) const;      // O(W) where W is
word.length()
  bool containsPrefix(string pre) const; // O(W) where W is
pre.length()
};
// Example for loop: for (string str : english){…}
```

utility

```
int isalpha(int ch); // Check if the given character is an
alphabetic character
int isspace(int c); // Check if the given character is an white
space such as Space, '\t', '\n',etc.
string toUpperCase(string str);
string toLowerCase(string str);
```

这里不再枚举各个库中具体函数的使用，仅仅列出头文件。借助 ide 的提示应该可以很快的找到并使用需要的函数。

- **filelib.h**: 文件操作相关
- **random.h**: 随机相关
- **simpio.h**: IO 相关
- **strlib.h**: 字符串相关

3 What? Algorithm?

这里给出可能会用到的深搜、广搜、链表模板。请注意，这些都是伪代码。

dfs

```
void solve_helper(Value sth, ..., Result res) {
  if (is_done(sth, ...)) {
    if (is_good(sth, mmm))
      res = get_result(sth, ...);
    return;
  }

  for (auto i : iter(sth, ...)) {
    change_something(sth, ...);
    solve_helper(new_sth, ..., res);
    recover(sth, ...);
  }
}

Result solve(Value sth, ...) {
  Result res;
  solve_helper(sth, 0, ..., res);
  return res;
}
```

permutation

```
void get_permutations_helper(const Vector<Value> &vec, int depth,
Vector<Vector<Value>> &res) {
  if (depth == vec.size()) {
    res += vec;
    return
  }

  auto v = vec;
  get_permutations_helper(v, depth + 1, res);
  for (int i = depth; i < vec.size(); ++i) {
    for (int j = i + 1; j < vec.size(); ++j) {
      std::swap(v[i], v[j]);
      get_permutations(v, depth + 1, res);
      std::swap(v[i], v[j]);
```

```
    }
  }
}

Vector<Vector<Value>> get_permutations(const Vector<Value> &vec) {
  Vector<Vector<Value>> res;
  get_permutations_helper(vec, 0, res);
  return res;
}
```

subset

```
void get_subsets_helper(const Vector<Value> &vec, const
Vector<Value> &v, int depth, Vector<Vector<Value>> &res) {
  if (depth == vec.size()) {
    res += v;
    return;
  }

  get_subsets_helper(vec, v, depth + 1, res);
  get_subsets_helper(vec, v + vec[depth], depth + 1, res);
}

Vector<Vector<Value>> get_subsets(const Vector<Value> &vec) {
  Vector<Vector<Value>> res;
  get_subsets_helper(vec, {}, 0, res);
  return res;
}
```

bfs

```
Result solve(Value sth) {
  Queue<Stage> q;
  q.enqueue(get_stage(sth));

  Map<Stage, Record> vis;

  ResultType res;

  while (!q.isEmpty()) {
    auto p = q.dequeue();

    if (is_good(p)) {
      res = get_result(p);
      break;
    }

    for (auto i : iter(p)) {
      if (!vis.containsKey(p)) {
        vis[p] = get_record(p);
        q.enqueue(proc(p));
      }
    }
  }

  return res;
}
```

maze

```
Queue<Pos> q;
q.enqueue(start);

Map<Pos, Pos> vis;

Vector<Pos> res;

while (!q.isEmpty()) {
  auto p = q.dequeue();

  if (p == target) {
    for (auto x = p; x != start; x = vis[x])
      res.insert(0, x);
    res.insert(0, start);
    break;
  }

  for (int i = 0; i < 4; ++i) {
    Pos np{p.x + dx[i], p.y + dy[i]};
    if (in_bound(np) && free_to_go(np) && !vis.containsKey(np)) {
      vis[np] = p;
      q.enqueue(np);
    }
  }
}

cout << res << endl;
```

list

```
struct Node {
  Value data;
  Node *next;
}

Node *build(const Vector<Value> &vec) {
  if (vec.isEmpty()) return nullptr;
  return new Node{vec[0], build(vec.subList(1))};
}

void print(Node *list) {
  if (list == nullptr) {
    cout << endl;
  } else {
    cout << list->data;
    if (list->next != nullptr) cout << " -> ";
    print(list->next);
  }
}
```