

**程 序 设 计 专 题**

**大 程 序 报 告**



大程名称： 排序算法可视化

**姓名：**

**学号：**

**电话：**

**指导老师： 纪守领**

**2020~2021春夏学期 2021 年 5 月 7 日**

**报告撰写注意事项**

1. 图文并茂。文字通顺，语言流畅，无错别字。
2. 书写格式规范，排版良好，内容完整。
3. 存在拼凑、剽窃等现象一律认定为抄袭；0分
4. 蓝色文字为说明，在最后提交的终稿版本，请删除这些文字。

**目 录**

[1 大程序简介 4](#_Toc69749328)

[1.1 选题背景及意义 4](#_Toc69749329)

[1.2 目标要求 4](#_Toc69749330)

[1.3 术语说明 4](#_Toc69749331)

[2 需求分析 4](#_Toc69749332)

[2.1 业务需求 4](#_Toc69749333)

[2.2 功能需求 4](#_Toc69749334)

[2.3 数据需求 5](#_Toc69749335)

[2.4 性能需求 5](#_Toc69749336)

[3 程序开发设计 5](#_Toc69749337)

[3.1 总体架构设计 5](#_Toc69749338)

[3.2 功能模块设计 5](#_Toc69749339)

[3.3 数据结构设计 5](#_Toc69749340)

[3.4 源代码文件组织设计 6](#_Toc69749341)

[3.5 函数设计描述 6](#_Toc69749342)

[4 部署运行和使用说明 6](#_Toc69749343)

[4.1 编译安装 6](#_Toc69749344)

[4.2 运行测试 6](#_Toc69749345)

[4.3 使用操作 7](#_Toc69749346)

[5 参考文献资料 7](#_Toc69749347)

“排序算法可视化”大程序设计

# 大程序简介

*<说明:描述大程序选题或开发背景，要实现的程序设计目标与要求、主要术语>*

## 选题背景及意义

排序算法是C程序设计中学习的重要算法，在实际应用中也有重要的基础性作用。将排序算法可视化，提供直观的演示，可降低初学者学习排序算法的难度。同时，“排序算法可视化”大程序也能综合应用已学的GUI程序设计、多文件、递归、文件I/O等知识。

## 目标要求

将几种基于比较的排序算法过程可视化，展示每个步骤数据的动态变化。支持自动执行和单步执行，并提供文件I/O支持。

## 术语说明

* 排序算法：改变一组数据的顺序，使其变得有序的算法，如从小到大。
* 基于比较的排序算法：通过数据之间的比较来实现排序的算法，大部分通常的排序算法，如冒泡排序、选择排序、归并排序、快速排序等，其理论最坏时间复杂度下界为。
* 非基于比较的排序算法：如计数排序、基数排序等。

# 需求分析

*<说明:分析你要开发的大程序的具体功能需求，包括哪些业务逻辑要求、需要哪些功能、哪些数据结构、哪些性能方面的要求>*

## 业务需求

*从业务上，描述实现的大程序是解决什么领域的什么问题，有哪些需求。*

将几种基于比较的排序算法过程可视化，展示每个步骤数据的动态变化。

* 排序算法数量不少于3种，至少包括冒泡排序、选择排序、归并排序。
* 排序的元素个数不少于10个。
* 显示排序算法的每一个步骤，展现动态移动的变化过程。



* 演示过程能够反映所采用的排序算法的特征。

## 功能需求

*从功能上，描述实现的大程序应该有哪些功能。画一个功能模块架构图，然后逐一介绍每个功能。*

* + 1. **基本功能**
* 演示过程中，既可自动执行，也可以点击或键盘操控单步执行。
* 能够从文件中读入被排序的数据（支持int、float、double），文件为文本文件，格式为：<类型> <数据表>

例如：float 1.0 3.0 15.0 6 8

* 支持过程的存档和调取，即能够在演示过程中，1）将必要的数据保存在快照文件中（二进制文件），2）关闭程序，3）读入快照文件中保存的数据，继续演示。
  + 1. **额外功能**
* 支持随机生成数据，调整数据规模。
* 支持调整演示速度，显示当前排序进度。
* 支持单步后退，即单步撤销上次的修改。
* 显示算法代码，实时高亮当前执行位置。
* 对于递归算法，通过不同颜色区分数据区域。

## 数据需求

*从数据上，描述实现的大程序应该包含怎样的数据要求，包括数据结构，数据格式以及内容。*

* int、float、double格式的原始数据、正在排序的数据
* 排序中各元素的当前颜色
* 当前状态和播放的进度
* 各种排序算法预处理后生成的帧数据，用于播放

## 性能需求

*从性能上，分析要实现的大程序应该满足怎样的性能，包括各个关键算法的运行时间速度、数据量、吞吐量等方面要求。*

* 单步执行或定时器触发重绘时，直接O(N)绘制各元素，因为数据量不大
* 预处理时生成的帧经过压缩，只保留每帧变化的部分，内部数据重绘复杂度大部分为O(1)，归并排序整个序列改变颜色除外
* 保存的状态文件空间复杂度与排序算法同阶，因为采取了压缩措施，比直接存储每帧所有元素节省大量空间

# 程序开发设计

*<说明：这部分详细介绍大程序开发设计有关内容，包括大程序总体架构设计、功能模块设计、数据结构设计、文件组织设计>*

## 总体架构设计

*<说明：这里给出一个总体架构图>*

## 功能模块设计

* **数据模块**：存储排序原始数据、当前数据，并提供比较、操纵数据的接口；支持多种数据类型
* **绘图模块**：将当前数据绘制在排序区，同时绘制菜单和工具栏
* **控制模块**：控制预处理、单步执行、保存状态（快照）等
* **算法模块**：实现多种排序算法，控制其与用户界面的交互

## 数据结构设计

1. color\_t: 表示元素颜色（彩色归并除外）的枚举类型
2. data\_t: 表示数据类型
3. algo\_t: 表示算法类型
4. status\_t: 表示排序状态，包括未初始化、正在排序、排序完成
5. data: 存储多种类型的排序数据、原始数据、元素颜色、最大最小值（便于绘制）等
6. ctrl: 存储当前帧、总帧数、排序状态、演示速度等控制数据
7. BubbleData: 存储冒泡排序的帧数据
8. SelectData: 存储选择排序的帧数据
9. MergeData: 存储归并排序的帧数据及元素是否正在进行二路归并等状态

## 源代码文件组织设计

**1）文件函数结构**

（可通过图或文字描述程序分为几个文件（需写出文件全名，即\*.h，\*.c）,每个.c文件包含哪些函数定义, 每个.h包含哪些内容。）

.

├── Makefile.win 项目构建文件

├── data.txt 数据文件样例

├── doc

│   ├── 2021大程序设计-纪守领.pdf 大程要求

│   └── 程序设计专题大程报告.docx

├── libgraphics

│   ├── include

│   │   ├── boolean.h 布尔类型支持

│   │   ├── exception.h

│   │   ├── extgraph.h 扩展图形库

│   │   ├── gcalloc.h

│   │   ├── genlib.h

│   │   ├── graphics.h 标准图形库

│   │   ├── imgui.h 界面控件库

│   │   ├── linkedlist.h

│   │   ├── random.h

│   │   ├── simpio.h

│   │   └── strlib.h

│   └── src

│   ├── exceptio.c

│   ├── genlib.c

│   ├── graphics.c 作了部分更改

│   ├── imgui.c

│   ├── linkedlist.c

│   ├── random.c

│   ├── simpio.c

│   └── strlib.c

├── obj

│   ├── \*.o 目标文件（省略）

├── src

│   ├── bubble.c 实现冒泡排序

│   ├── bubble.h

│   ├── control.c 控制模块

│   ├── control.h

│   ├── data.c 数据模块

│   ├── data.h

│   ├── draw.c 排序区和工具栏绘制

│   ├── draw.h

│   ├── general.h 通用头文件

│   ├── log.c 日志模块（不重要）

│   ├── log.h

│   ├── main.c 主程序

│   ├── menu.c 菜单绘制

│   ├── menu.h

│   ├── merge.c 实现归并排序

│   ├── merge.h

│   ├── select.c 实现选择排序

│   └── select.h

├── state.sav 保存的状态（快照）

├── vsort.dev 项目文件

├── vsort.exe 可执行文件

├── vsort.exe.Manifest 新版视觉效果

├── vsort.ico 图标文件

└── vsort.log 日志，包含渲染用时等，仅用于调试

**2）多文件构成机制**

（说明分文件构成程序的实现机制，即如何具体采用文件包含、#define保护、外部变量或外部函数）

src

├── bubble.c general.h bubble.h data.h control.h

├── bubble.h

├── control.c general.h control.h data.h bubble.h select.h

│ merge.h draw.h

├── control.h

├── data.c general.h data.h merge.h control.h

├── data.h

├── draw.c general.h draw.h data.h merge.h control.h

│ log.h menu.h bubble.h select.h

├── draw.h

├── general.h stdio.h stdlib.h time.h string.h math.h

│ graphics.h extgraphics.h imgui.h windows.h

│ commctrl.h Boolean.h

├── log.c general.h log.h

├── log.h

├── main.c general.h data.h draw.h control.h log.h

├── menu.c general.h menu.h log.h control.h data.h

│ merge.h draw.h

├── menu.h

├── merge.c general.h merge.h data.h control.h

├── merge.h

├── select.c general.h select.h data.h control.h

└── select.h

头文件都使用了#define \_\_HEADER\_H保护，防止重复编译。在头文件中定义的变量都用了extern，便于在其他文件中使用，并在对应的源文件中用非extern定义。

## 函数设计描述

<说明：每个函数按如下形式描述：

函数原型：

功能描述；

参数描述：

返回值描述：

重要局部变量定义：

重要局部变量用途描述：

函数算法描述：

>

**main.c**

/\*\*

\* Char event callback for imgui

\* Redraw after GUI update

\*\*/

static void CharEventProcess**(**char ch**);**

/\*\*

\* Keyboard event callback for imgui

\*\*/

static void KeyboardEventProcess**(**int key**,** int event**);**

/\*\*

\* Mouse event callback for imgui

\*\*/

static void MouseEventProcess**(**int x**,** int y**,** int button**,** int event**);**

/\*\*

\* Timer event callback for playing animation

\* Next step on timer

\*\*/

static void TimerEventProcess**(**int timerID**);**

/\*\*

\* Init GUI for libgraphics

\*\*/

void Main**();**

**menu.h**

/\*\*

\* Process file menu, concerning load / save state

\* @selection start from 1

\*\*/

void menuFileProcess**(**int selection**);**

/\*\*

\* Control menu, concerning play navigation

\*\*/

void menuControlProcess**(**int selection**);**

/\*\*

\* Data menu, concerning randomize / load data

\*\*/

void menuDataProcess**(**int selection**);**

/\*\*

\* Algorithm menu, concerning switch sorting algorithms

\*\*/

void menuAlgoProcess**(**int selection**);**

/\*\*

\* Render the menu

\*\*/

void drawMenu**();**

**data.h**

/\*\*

\* calculate bar length of data[index] for chart

\* return 0.5 if all values are equal (min = max)

\* @index index of data

\* **@return** value range [0.1, 1]

\*\*/

double relativeRatio**(**int index**);**

/\*\*

\* read from MergeData instead of data

\*\*/

double relativeMergeRatio**(**int index**);**

/\*\*

\* convert data to string, for chart label

\* @index index of data

\* **@return** string output

\* !free string after using!

\*\*/

char **\***tostring**(**int index**);**

/\*\*

\* read from MergeData instead of data

\*\*/

char **\***toMergestring**(**int index**);**

/\*\*

\* allocate data & color arrays

\*\*/

void allocData**();**

/\*\*

\* free arrays

\*\*/

void freeData**();**

/\*\*

\* backup / restore data

\* @backup backup to pbk if TRUE

\* restore from pbk if FALSE

\*\*/

void transferData**(**bool backup**);**

/\*\*

\* init min & max to data

\*\*/

void initMinMax**();**

/\*\*

\* randomly generate data

\*\*/

void generateData**();**

/\*\*

\* compare data[x] and data[y]

\* @x y indices of data

\* **@return** <0 if data[x] < data[y]

\* =0 if data[x] = data[y]

\* >0 if data[x] > data[y]

\*\*/

int cmp**(**int x**,** int y**);**

/\*\*

\* swap data[x] and data[y]

\*\*/

void swap**(**int x**,** int y**);**

/\*\*

\* load data from file

\*\*/

void loadData**();**

**control.h**

/\*\*

\* init frames if not & reset if sorted

\*\*/

void init**();**

/\*\*

\* create frames with corresponding algorithm

\*\*/

void sort**();**

/\*\*

\* display next step with corresponding algorithm

\* decode from frame information created by 'sort'

\*\*/

void NextStep**();**

/\*\*

\* display previous step with corresponding algorithm

\* apparently harder than 'NextStep'

\*\*/

void PrevStep**();**

/\*\*

\* set all color to CNone / not fade

\*\*/

void clearFrame**();**

/\*\*

\* load state from binary file \*.sav

\*\*/

void loadState**();**

/\*\*

\* save state to binary file \*.sav

\*\*/

void saveState**();**

**draw.h**

/\*\*

\* draw chart, except merge sort

\*\*/

void drawChart**();**

/\*\*

\* draw chart, merge sort with simple color only

\*\*/

void drawMergeChart**();**

/\*\*

\* draw chart, merge sort colorful only

\*\*/

void drawMergeChartColorful**();**

/\*\*

\* redraw menu & chart

\*\*/

void redraw**();**

/\*\*

\* draw toolbar on the bottom

\*\*/

void drawToolbar**();**

/\*\*

\* draw code on the right

\*\*/

void drawCode**();**

void WINAPI CreateBars**(**

HWND hwndDlg**,** // handle of dialog box (parent window)

UINT iMin**,** // minimum value in trackbar range

UINT iMax**);** // maximum value in trackbar range

**bubble.h**

/\*\*

\* generate frames by bubble sort

\*\*/

void Bubblesort**();**

/\*\*

\* next frame by updating data and color

\*\*/

void BubbleNextStep**();**

/\*\*

\* previous frame; exactly undo what NextStep did

\*\*/

void BubblePrevStep**();**

**select.h**

/\*\*

\* generate frames by selection sort

\*\*/

void Selectsort**();**

/\*\*

\* next frame by updating data and color

\*\*/

void SelectNextStep**();**

/\*\*

\* previous frame; exactly undo what NextStep did

\*\*/

void SelectPrevStep**();**

**merge.h**

/\*\*

\* merge sort first run with recursion

\*\*/

void PreMergesort**(**int l**,** int r**);**

/\*\*

\* merge sort second run, creating frames

\*\*/

void InMergesort**(**int l**,** int r**);**

/\*\*

\* generate frames by selection sort

\* call PreMergesort & InMergesort

\*\*/

void Mergesort**();**

/\*\*

\* next frame by updating data and color (non-colorful)

\*\*/

void MergeNextStep**();**

/\*\*

\* next frame; except for colorful version

\*\*/

void MergeNextStepColorful**();**

/\*\*

\* previous frame; exactly undo what NextStep did (non-colorful)

\*\*/

void MergePrevStep**();**

/\*\*

\* previous frame; except for colorful version

\*\*/

void MergePrevStepColorful**();**

# 部署运行和使用说明

*<此部分介绍如何由提交的源代码包，进行存放、编译生成.exe文件的过程说明，以及运行.exe后的用户使用手册>。*

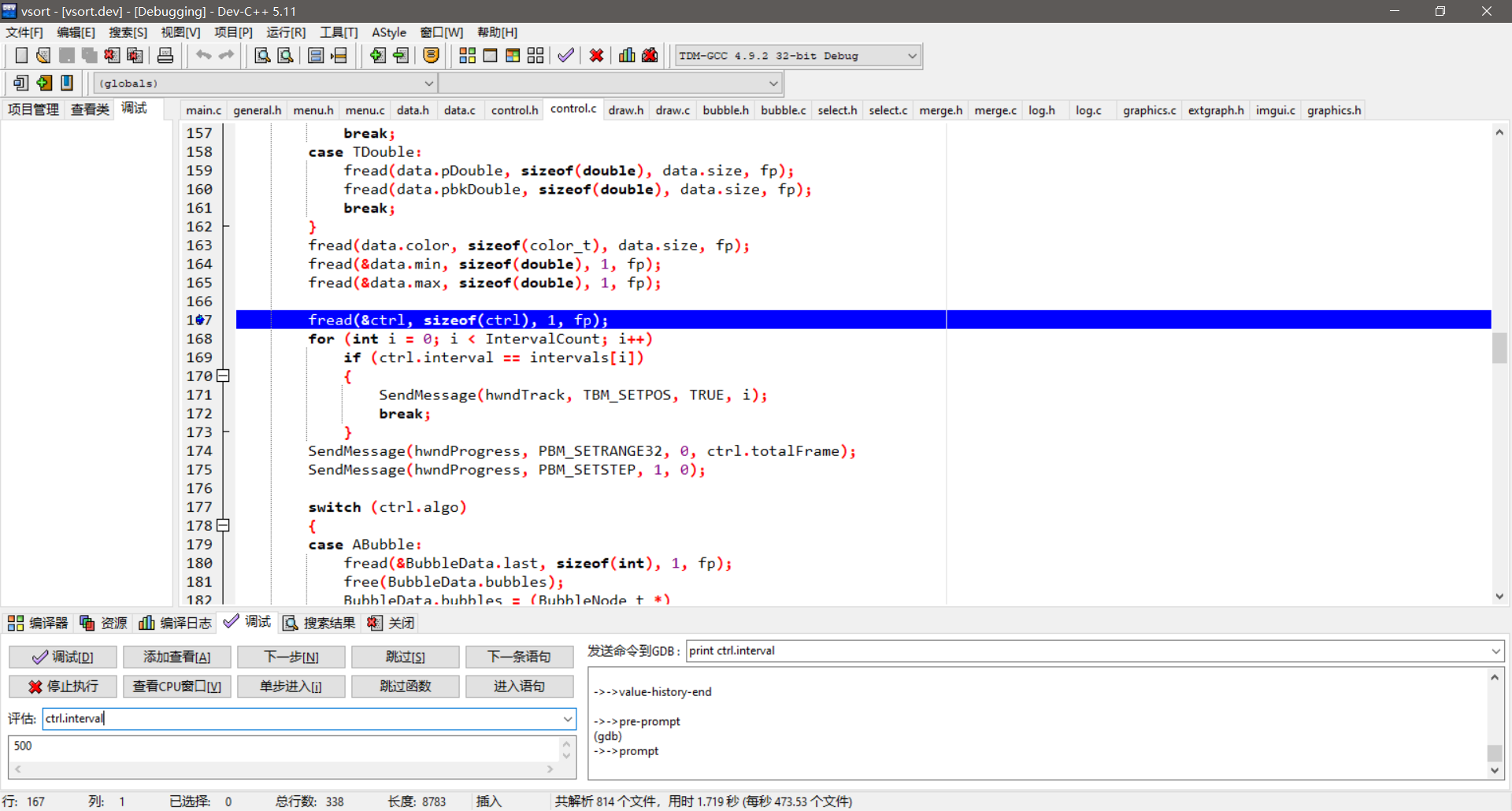
## 编译安装

*<说明：如何从源代码开始编译生成可执行文件，如何运行的说明>*

1. 若未安装Dev-C++，请安装；
2. 打开根目录下的vsort.dev项目文件，按F12全部重新编译；
3. 按F10或打开vsort.exe即可运行；
4. 若需要进行调试，打开“项目选项”，将“编译器”标签页下的编译器改为Debug，选择Yes，并在“代码生成”子标签页下的**语言标准改为C99标准**（与pintia一致）。

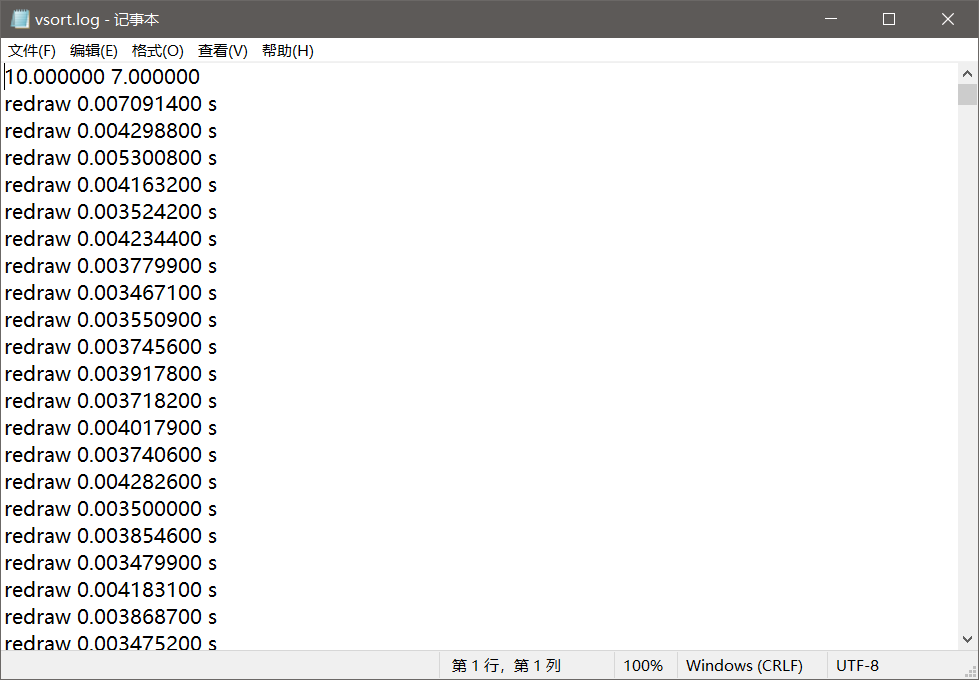
## 运行测试

*<选取测试阶段典型的案例，说明如何设计测试数据，发现和定位错误的，测试结果可以含有屏幕截图。>*



测试方法一：使用Dev-C++进行调试，图为解决演示速度滑动条未更新的bug

使用内置的调试工具（GDB）还能轻松找出程序崩溃的位置。

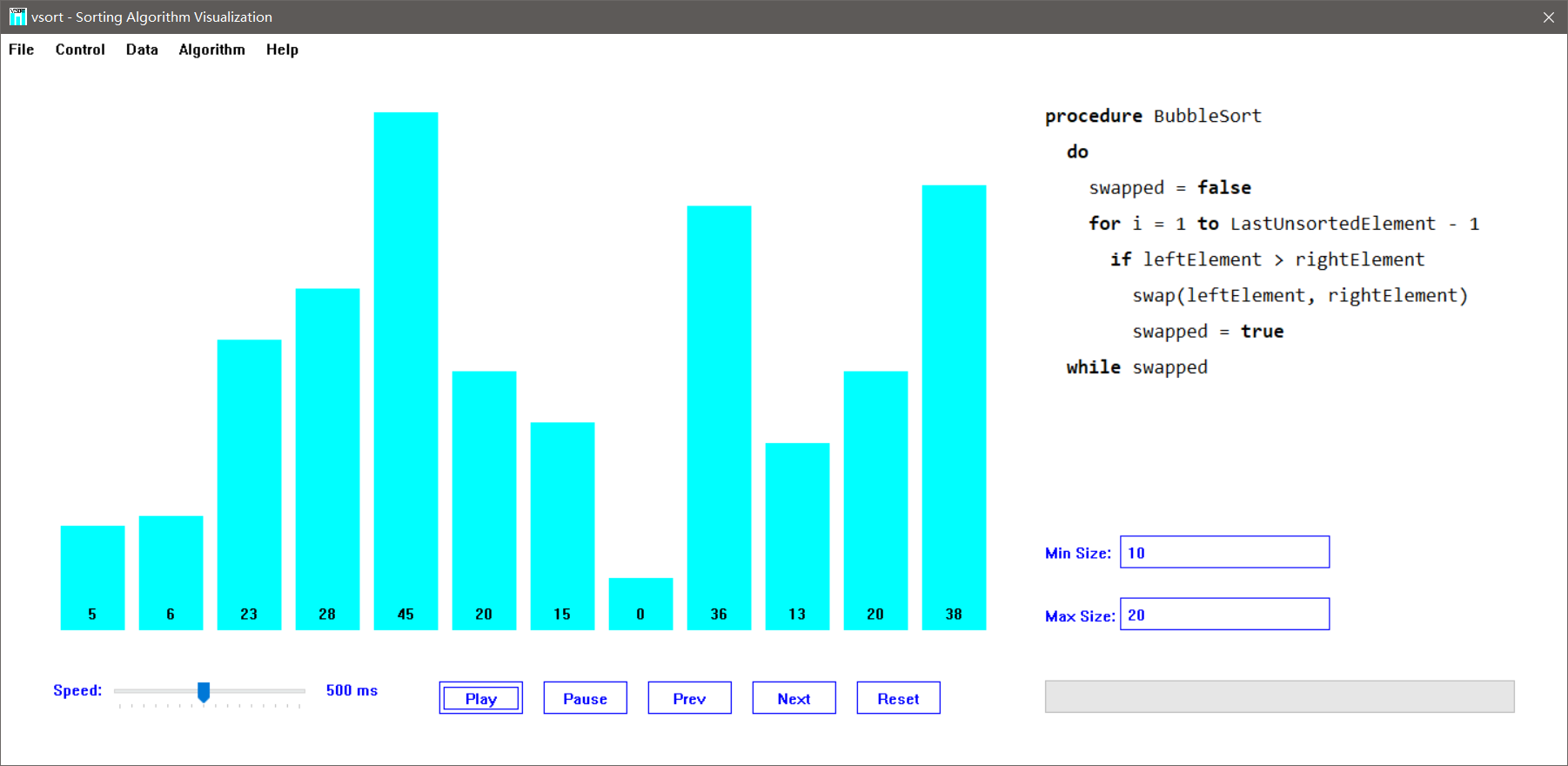


测试方法二：输出调试，由于控制台操作不美观，使用日志文件

## 使用操作

*<此部分介绍如何使用操作大程序，包括各个功能如何操作，结合一些截图>。*

**主界面**



⑦

⑥

⑤

④

③

②

①

① 菜单栏：**文件**菜单用于读取、保存快照，**控制**菜单用于播放、暂停、单步执行、重置动画播放，**数据**菜单用于随机生成数据或从文件载入数据，**算法**菜单用于切换算法（其中彩色归并比普通归并更美观，推荐），**帮助**菜单用于打开关于对话框。

② 排序区：用于显示排序动画，其中数据标签过窄时不显示

③ 代码区：用于显示排序算法对应的伪代码及执行情况，支持代码高亮

④ 用于设置随机数据的规模

⑤ 滑动条（直接用底层Windows API实现）：用于调整演示速度，右边显示的标签为两次重绘的时间间隔

⑥ 导航按钮：与控制菜单功能一致，便于操作

⑦ 进度条（直接用底层Windows API实现）：显示排序进度

**特色功能介绍**

1. 支持随机生成数据，支持超大规模的排序数据。即使数据规模达到1000也能较为流畅地运行，最高可达9999，仍能正常运行。
2. 采用底层Windows API实现了滑动条、进度条、自定义的程序图标，方便用户调整演示速度和观察排序进度。并通过避免重绘，解决了Windows控件与imgui一起工作时的闪烁问题。
3. 相比普通的排序算法可视化，增加了连续单步后退功能，能撤销之前的操作，便于用户观察排序算法。并采用压缩技巧大幅减小快照文件。
4. 显示了算法的伪代码，方便用户学习算法。
5. 实现了彩色归并，非常漂亮。自己实现了HSL转RGB，并绕过了libgraphics仅支持256种颜色的限制。

## 一小部分实现历程

我非常想有一个好的用户界面来调整演示（动画）速度，有两种选择：一种是自己用类似 imgui 的方式来实现，捕捉各种鼠标事件，但显然非常麻烦，而且会产生大量潜在 bug；另一种是直接用 Win32 Trackbar，非常方便，但和 imgui 集成有很大的困难。我最终决定还是用 Windows 控件省事。

Microsoft Docs 真的非常详细，但要求读者有一定的基础，而我并没有多少 Win32 编程的经验，于是磕磕绊绊。创建滑动条可以直接抄代码，后面就比较麻烦了。我先后遇到了**消息响应**、**标签提示**、**闪烁**、**调节背景色**、**吞焦点导致 imgui 的快捷键失效**等一系列问题。

* 当滑动条移动的时候怎么更新演示速度呢？查了半天，得改窗体的消息响应（这个写在 graphics.c 中，于是接下来我改了很多图形库的代码），处理*WM\_HSCROLL*事件。这个事件还不会返回当前滑动条的位置，还需要发送*TBM\_GETPOS*消息请求。然后，我不希望用户还没松开鼠标就改变演示速度，这个只要在*TB\_ENDTRACK*时再更新演示速度即可。
* 由于我采用了非线性速度调节，即创建了一个数组来映射定时器时间间隔，需要提示用户当前选择的速度对应的时间间隔。这个我非常想用内置的*Hint*，但怎么也实现不了，只能显示数组下标……最终我决定直接在滑动条旁边放一个*Label*来显示。
* 闪烁问题显然是 libgraphics 或 imgui 引起的，仔细检查发现是*DisplayClear*的锅，这玩意儿会强制所有区域重绘。于是解决方法很简单，把控件所在的矩形区域挖掉，不要强制重绘就好了，在窗体创建的时候绘制一次就好了。
* 滑动条的背景色不是白色，在白色背景的窗体上显得很突兀。要是一般的 UI 设计方案，只要改一下类似*BackgroundColor*就好了的事情，底层就很麻烦。我没想到需要在窗体处理*WM\_CTLCOLORSTATIC*事件，返回一个白色的*HBRUSH*。
* 最后好不容易滑动条终于融入了窗体，突然发现拖动滑动条之后，菜单的快捷键失效了！所幸我还知道 Win32 焦点那套东西，即在窗体上按 Tab 可以切换焦点，到达不同的控件。然而这个窗口上只有一个滑动条控件，菜单根本不是控件，事件完全由窗体来处理！那么解决思路也很简单，把焦点还给窗体即可，只要一个*SetFocus*。
* 可是还是有一个缺点，在鼠标拖动滑动条时它还是会获得焦点，同时外面出现了虚线框来表示它具有焦点，这就显得特别丑陋和突兀。查到需要发送*WM\_CHANGEUISTATE*消息到滑动条，参数是*UISF\_HIDEFOCUS*……

# 参考文献资料

*<列出参考的书籍、论文、网站的信息和地址等>。*

需求参考网站：

<https://visualgo.net/en/sorting>

<https://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html>

HSL to RGB：

<https://www.rapidtables.com/convert/color/hsl-to-rgb.html>

微软官方文档：

<https://docs.microsoft.com/en-us/windows/win32/controls/window-controls>