

#

测试作业

1实现快速排序、选择排序、插入排序、希尔排序和冒泡排序

- 快速排序

```
1 //递归实现
2 //头文件中应包含partition和void quickSort的声明
3 //define N xxxx;
4 //将xxxx换成需要排序的数组长度
5 #include"myLibrary"
6 int partition(int* arr, int left, int right)
7 {
8     int i = left;
9     int k = left;
10    for (i = left; i < right; i++)
11    {
12        if (arr[right] > arr[i])
13        {
14            swap(arr[i], arr[k]);
15            k++;
16        }
17    }
18    swap(arr[k], arr[right]);
19    return k;
20 }
21 void quickSort(int* arr,int left, int right)
22 {
23     int pivot;
24     if(left<0||right>=N)
25     {
26         cout<<"这活我没法接，请输入正确的边界值"<<endl;
27         return;
28     }
29     if (left<right) //递归出口
30     {
31         pivot = partition(arr, left, right);
32         quickSort(arr, left,pivot - 1);
33         quickSort(arr, pivot - 1, right);
34     }
35 }
```

```
1 //递归实现 单个函数
2 //头文件中应包含void quickSort2的声明
3 //define N xxxx;
4 //将xxxx换成需要排序的数组长度
5 #include"myLibrary"
6 void quickSort2(int* arr, int left, int right)
7 {
8     if (left >= right) return;//递归出口
9     int i, j, temp;
```

```

10     i = left;
11     j = right;
12     int pivot = arr[i];
13     while (i < j)    //左右哨兵握手时候退出循环
14     {
15         while (i < j && arr[j] >= pivot)
16         {
17             j--;
18         }
19         while (i < j && arr[i] <= pivot)
20         {
21             i++;
22         }
23         if (i < j) //swap函数
24         {
25             temp = arr[i];
26             arr[i] = arr[j];
27             arr[j] = temp;
28         }
29     }
30     //重置哨兵值
31     arr[left] = arr[i];
32     arr[i] = pivot;
33     quickSort2( arr, left, i - 1);
34     quickSort2( arr,i-1, right);
35 }
36
37

```

o

- 选择排序

```

1  //头文件中应包含selectSort的声明
2  //剩余同上
3  #include"myLibrary"
4  void selectSort(int *Arr)
5  {
6      int Maxpots;
7      for(int i=0;i<N-1;i++)
8      {
9          Minpots=i; //每次循环重置Maxpots的值
10         for(int j=1;j<i;j++)
11         {
12             if(Arr[Minpots]>Arr[j])
13             {
14                 Minpots=j;
15             }
16             swap(Arr[i-1],Arr[Minpots]);
17         }
18     }
19 }

```

- 插入排序

```

1  //头文件中应包含inSert的声明
2  //剩余同上
3  #include"myLibrary"

```

```

4 void insert(int* arr)
5 {
6     int insertValue = 0;
7     int j;
8     for (int i = 0; i < N; i++)
9     {
10        insertValue = arr[i];
11        for (j = i - 1; j >= 0; j--)
12        {
13            if (arr[j] > insertValue)
14            {
15                arr[j + 1] = arr[j];
16            }
17            else
18            {
19                break;
20            }
21        }
22        arr[j + 1] = insertValue;
23        //在找到插入位置之后再放入insertValue的值
24    }
25 }

```

- 希尔排序

```

1 //头文件中应包含shellSort的声明
2 //剩余同上
3 #include"myLibrary"
4 void shellSort(int* arr)
5 {
6     int i, j, insertValue, gap;
7     for (gap = N >> 1; gap > 0; gap >>= 1)
8     {
9         for (int i = gap; i < N; i++)
10        {
11            insertValue = arr[i];
12            for (j = i - gap; j >= 0; j-=gap)
13            {
14                if (arr[j] > insertValue)
15                {
16                    arr[j + gap] = arr[j];
17                }
18                else
19                {
20                    break;
21                }
22            }
23            arr[j + gap] = insertValue;
24            //在找到插入位置之后再放入insertValue的值
25        }
26    }
27 }

```

- 冒泡排序

```

1 //头文件中应包含bubbleSort的声明
2 //剩余同上

```

```

3  #include"myLibrary"
4  void bubblesort(int *Arr)
5  {
6      for(int i=0;i<=N;i++)
7      {
8          for(int j=0;j<N-i;j++)
9          {
10             if(Arr[i]>Arr[i+1])
11             {
12                 swap(Arr[i],Arr[i+1]);
13             }
14         }
15     }
16 } //天都黑了啥时候冒好啊啊啊啊啊啊啊啊啊

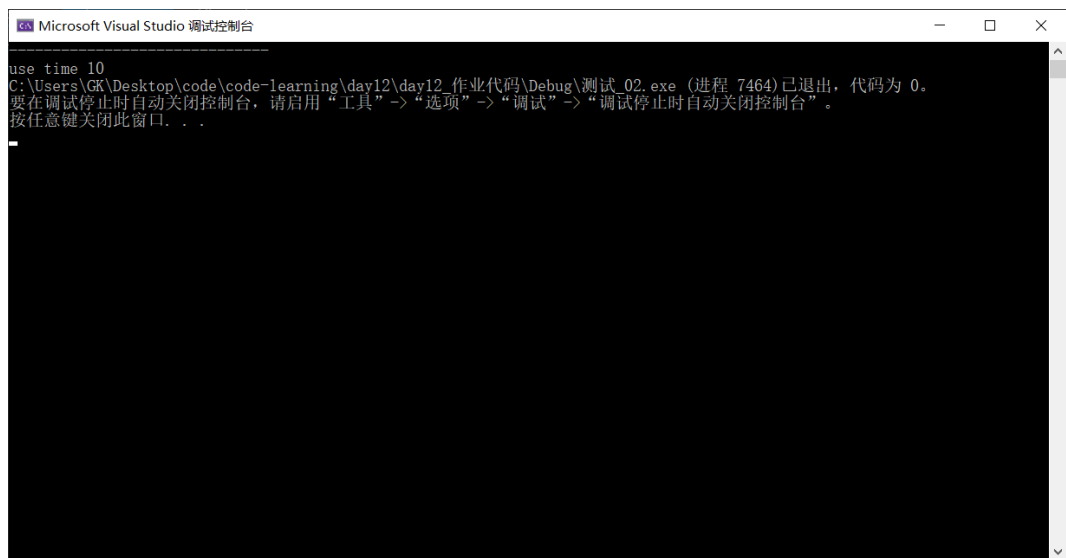
```

○

###

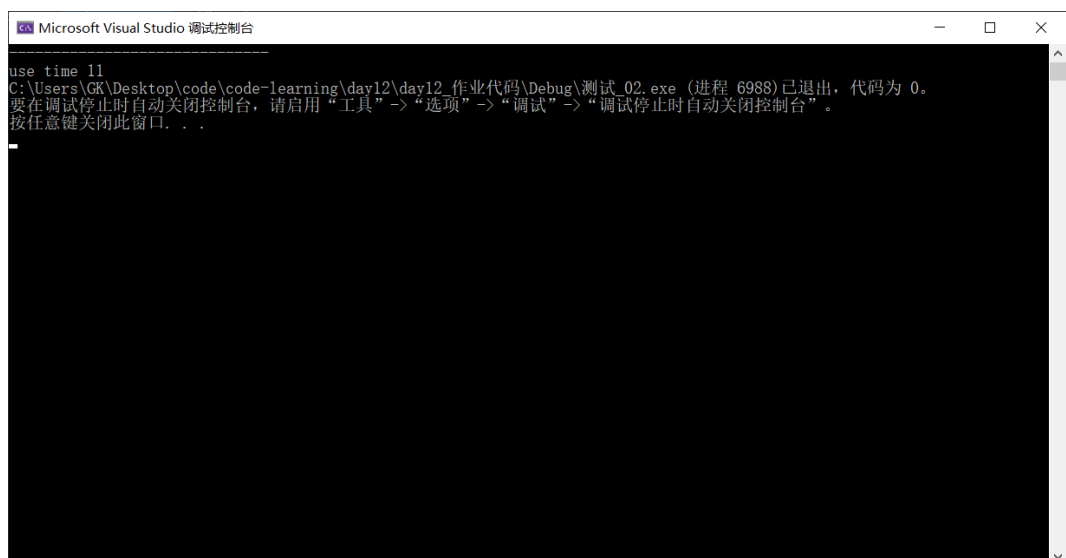
2排序2000万个数字（数字范围0~100000），比较希尔排序、快速排序和qsort的时间差异

- 快速排序：
- 希尔排序



• ○

- qsort



• ○

- 主函数

```
1  #include "myLibrary.h"
2  int compare(const void* p1, const void* p2);
3  int main()
4  {
5
6      int* Arr = (int*)malloc(N * sizeof(int));
7      srand(time_t(NULL));
8      time_t start, end;
9      for (int i = 0; i < N; i++)
10     {
11         Arr[i] = rand();
12         //cout << Arr[i] << " ";
13     }
14     //cout << endl;
15     cout << "-----" << endl;
16     start = time(NULL);
17     //qsort(Arr, N, sizeof(int), compare);
18     //shellSort(Arr);
19     //selectSort(Arr); //我睡着了 真的
20     quickSort(Arr, 0, N - 1);
21     //quickSort2(Arr, 0, N - 1); //待定 调试信息：引发了异常：读取访问权
    限冲突。
22     end = time(NULL);
23     printf("use time %d", (int)end - (int)start);
24     return 0;
25 }
26 int compare(const void* left, const void* right)
27 {
28     int* p1 = (int*)left;
29     int* p2 = (int*)right;
30     return *p1 - *p2;
31 }
32
```

- 头文件

```
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <iostream>
3  #include <cstdio>
4  #include <cmath>
5  #include <cstdlib>
6  #include <string>
7  #include <time.h>
8  constexpr auto N = 20000000;
9  using namespace std;
10 void shellSort(int* arr);
11 int partition(int* arr, int left, int right);
12 void quickSort(int* arr, int left, int right);
13 void quickSort2(int* arr, int left, int right);
14 void selectSort(int* Arr);
```

- sort.cpp

```

1  #include "myLibrary.h"
2  void shellSort(int* arr)
3  {
4      int i, j, insertValue, gap;
5      for (gap = N >> 1; gap > 0; gap >>= 1)
6      {
7          for (int i = gap; i < N; i++)
8          {
9              insertValue = arr[i];
10             for (j = i - gap; j >= 0; j -= gap)
11             {
12                 if (arr[j] > insertValue)
13                 {
14                     arr[j + gap] = arr[j];
15                 }
16                 else
17                 {
18                     break;
19                 }
20             }
21             arr[j + gap] = insertValue;
22             //在找到插入位置之后再放入insertValue的值
23         }
24     }
25 }
26 void selectSort(int* Arr)
27 {
28     int Minpots;
29     for (int i = 0; i < N - 1; i++)
30     {
31         Minpots = i; //每次循环重置Maxpots的值
32         for (int j = 1; j < i; j++)
33         {
34             if (Arr[Minpots] > Arr[j])
35             {
36                 Minpots = j;
37             }
38             swap(Arr[i - 1], Arr[Minpots]);
39         }
40     }
41 }
42 int partition(int* arr, int left, int right)
43 {
44     int i = left;
45     int k = left;
46     for (i = left; i < right; i++)
47     {
48         if (arr[right] > arr[i])
49         {
50             swap(arr[i], arr[k]);
51             k++;
52         }
53     }
54     swap(arr[k], arr[right]);
55     return k;
56 }
57 void quickSort(int* arr, int left, int right)

```

```

58 {
59     int pivot;
60     if (left < 0 || right >= N)
61     {
62         cout << "这活我没法接，请输入正确的边界值" << endl;
63         return;
64     }
65     if (left < right) //递归出口
66     {
67         pivot = partition(arr, left, right);
68         quickSort(arr, left, pivot - 1);
69         quickSort(arr, pivot + 1, right);
70     }
71 }
72 //void quickSort2(int* arr, int left, int right)
73 //{
74 //    if (left >= right) return; //递归出口
75 //    int i, j, temp;
76 //    i = left;
77 //    j = right;
78 //    int pivot;
79 //    pivot = arr[left];
80 //    while (i < j) //左右哨兵握手时候退出循环
81 //    {
82 //        while (i < j && arr[j] >= pivot)
83 //        {
84 //            j--;
85 //        }
86 //        while (i < j && arr[i] <= pivot)
87 //        {
88 //            i++;
89 //        }
90 //        if (i < j) //swap函数
91 //        {
92 //            temp = arr[i];
93 //            arr[i] = arr[j];
94 //            arr[j] = temp;
95 //        }
96 //    }
97 //    //重置哨兵值
98 //    arr[left] = arr[i];
99 //    arr[i] = pivot;
100 //    quickSort2(arr, left, i - 1);
101 //    quickSort2(arr, i + 1, right);
102 //}
103 //
104

```