

Forum:[Hack Forums](#)

Thread: [How would you manage thiss list with code?](#)

Renovatum - March 8, 2018, 9:41 am

I have a list of 20 million or so names (average 6 chars), one each line, saved in either a .csv or a txt file.

Some names appear more than twice, other's appear only once. I only want those who appear more than x amount of times to do resaved into another list of the names with the total number amount of times it appeared in the file next to each name, those who appeared less than x being deleted.

(I know this won't be interesting, but is what I've been trying to do):

- Tried using .bat and powershell because it's all I KIND OF know

Basically my code checked each line and tried to save it to a either a %variable% or \$variable, respectively

- if the variable exists, set %(name)(number)%+1 and save to the variable

%name%%namenumber%

nextline (or check all content with get-content)

- When it finishes each %name% is printed with %namenumber) next to it to show the amount

I realised this was retarded and tried using arrays but I couldn't get it to work.

I don't code but I thought the puzzle was fun. It was for the first hour or so, but I have work to do which relates to this and I don't know enough about the language to do it

Nachfolger - March 8, 2018, 9:52 am

What application are you wanting to implement this into? Desktop, Web, etc? You mentioned a batch file but that's retarded imo. Sorting 20 million items is going to be a lengthy/intensive task, regardless of the application of implementation.

Because of my limited knowledge, I'm thinking of using something like PHP and save each name as a JSON key with the number of counts as a object within the JSON. I'm not sure how effecient this will be, though.

Renovatum - March 8, 2018, 10:12 am

(Mar 8, 2018 10:52 AM)Nachfolger Wrote: [\[CITE\]](#) What application are you wanting to implement this into? Desktop, Web, etc? You mentioned a batch file but that's retarded imo. Sorting 20 million items is going to be a lengthy/intensive task, regardless of the application of implementation.

Because of my limited knowledge, I'm thinking of using something like PHP and save each name as a JSON key with the number of counts as a object within the JSON. I'm not sure how effecient this will be, though.

I'm hoping to use it on a desktop or Windows server.

I know it's retarded haha, but it's all I could learn at the time. I tried vbs but lost interest after high school.

Anyway, I don't know PHP :(

aeiro - March 8, 2018, 10:18 am

(Mar 8, 2018 11:12 AM)Renovatum Wrote: [\[CITE\]](#) I'm hoping to use it on a desktop or Windows server.

I know it's retarded haha, but it's all I could learn at the time. I tried vbs but lost interest after high school.

Anyway, I don't know PHP :(

This shouldn't be too hard in c++. Try using hash table sorting. If the name already == another name in the list just ++ the number.

(Mar 8, 2018 11:18 AM)aeiro Wrote: [\[CITE\]](#) This shouldn't be too hard in c++. Try using hash table sorting. If the name already == another name in the list just ++ the number.

Written in C++. This requires that each name in your list is on a new line. Will output a new file with a new list looking like.

Name:NumberOfTimes\n

<http://www.mediafire.com/file/a4g9zyn5sv99cq3/SortNames.rar>

EDIT: There is was an error with the HashAdd function. if you want to manually fix it the code here
Spoiler [\(Click to View\)](#)

```
Code void Hash_handler::HashAdd(Hash_handler* hh, std::string name)
{
```

```
    int dex = HashIndxr(name);
```

```
    if ((hh->_D[dex]->name[0] == NULL)) {
        hh->_D[dex]->name = name;
        hh->_D[dex]->Count++;
        return;
    }
```

```
    else {
        if (hh->_D[dex]->name == name) {
            hh->_D[dex]->Count++;
            return;
        }
```

```
        Data* ptr = hh->_D[dex];
        while (ptr->next != NULL)
        {
            if (ptr->name == name)
            {
                ptr->Count++;
                return;
            }
            ptr = ptr->next;
        }
```

```
        Data* tmp = new Data();
        tmp->name = name;
        tmp->Count = 1;
        tmp->next = NULL;
```

```
ptr->next = tmp;
}
return;
}
I'm re uploading the source now.
```

Gorzoid - March 8, 2018, 1:46 pm

OP you asked this question only a week ago <https://hackforums.net/showthread.php?tid=5805429> why make a new thread.

(Mar 8, 2018 11:18 AM)aeiro Wrote: [\[CITE\]](#) Written in C++. This requires that each name in your list is on a new line. Will output a new file with a new list looking like.

Not sure if you have just never written C++ or just been taught by a troll. You use the new keyword as if it is java and never call delete, you have no idea how member functions work (you don't need the pointer to the instance as first argument, just reference the member object like you would java `_D[idx]` or `this->_D[idx]`) and you lack any knowledge of the STL datatypes that do this all for you. Your code can be done in much fewer lines and much faster.

```
Code const std::size_t x = 2;
std::ifstream inFile("NameList.txt");
std::ofstream outFile("UniqueNameList.txt");
```

```
std::unordered_map<std::string,std::size_t> map;
std::string line;
```

```
for (getline(inFile, line, '\n'); !inFile.eof(); getline(inFile, line, '\n'))
    ++map[line];
```

```
for (const auto& p : map)
{
```

```
    if (p.second >= x)
        outFile << p.first << std::endl;
```

}removed the occurrence count since op never asked for it. for file with 10million lines (500k unique values), that code took 14 seconds. Your code... well it's taking a while, will edit if it ever terminates.

aeiro - March 8, 2018, 2:03 pm

(Mar 8, 2018 02:46 PM)Gorzoid Wrote: [\[CITE\]](#) OP you asked this question only a week ago <https://hackforums.net/showthread.php?tid=5805429> why make a new thread.

Not sure if you have just never written C++ or just been taught by a troll. You use the new keyword as if it is java and never call delete, you have no idea how member functions work (you don't need the pointer to the instance as first argument, just reference the member object like you would java `_D[idx]` or `this->_D[idx]`) and you lack any knowledge of the STL datatypes that do this all for you. Your code can be done in much fewer lines and much faster.

```
Code const std::size_t x = 2;
std::ifstream inFile("NameList.txt");
std::ofstream outFile("UniqueNameList.txt");
```

```
std::unordered_map<std::string,std::size_t> map;
std::string line;
```

```
for (getline(inFile, line, '\n'); !inFile.eof(); getline(inFile, line, '\n'))  
    ++map[line];
```

```
for (const auto& p : map)  
{
```

```
    if (p.second >= x)  
        outFile << p.first << std::endl;
```

}removed the occurrence count since op never asked for it. for file with 10million lines (500k unique values), that code took 14 seconds. Your code... well it's taking a while, will edit if it ever terminates. I've never been taught by anyone. i learn from writing the code.

I also assumed that once the function returned. The pointer would delete itself. I also was unable to test with a file with that many lines. If you have one please forward it to me so i can test.

Also I'm sure if you made the TABLE_SIZE larger it would would much faster.

also

(Mar 8, 2018 02:46 PM)Gorzoid Wrote: [\[CITE\]](#) removed the occurrence count since op never asked for it. for file with 10million lines (500k unique values), that code took 14 seconds. Your code... well it's taking a while, will edit if it ever terminates. (Mar 8, 2018 10:41 AM)Renovatum Wrote: [\[CITE\]](#) I only want those who appear more than x amount of times to do resaved into another list of the names with the total number amount of times it appeared in the file next to each name, those who appeared less than x being deleted.

Renovatum - March 8, 2018, 2:21 pm

Thanks to the two people who helped.

The guy learns from coding BTW- I think that's admirable, and you could teach him in a nicer way. I appreciate all that both of you have done for me

Woes - March 8, 2018, 2:25 pm

20M is a bunch, but let me know if you need this done in Python lol.

Renovatum - March 8, 2018, 3:47 pm

(Mar 8, 2018 03:25 PM)Woes Wrote: [\[CITE\]](#) 20M is a bunch, but let me know if you need this done in Python lol.

That's the estimate off what it might come to. More, even, when I reach celebrity status ;)

I still need code that can do it quick enough with a count on duplicates for those which have them. It would also help if the user who appear just once are deleted to save space, or if they can be organised from most to least amounts to save time.

Gorzoid - March 8, 2018, 3:48 pm

(Mar 8, 2018 03:03 PM)aeiro Wrote: [\[CITE\]](#) I've never been taught by anyone. i learn from writing the code.

I also assumed that once the function returned. The pointer would delete itself. I also was unable to test with a file with that many lines. If you have one please forward it to me so i can test.

Also I'm sure if you made the TABLE_SIZE larger it would would much faster.

Yeah I started with a harsh comment and then thought maybe I should just point out the mistakes, kinda did a lazy ass job of it.

So you use new but C++ is not like java in that it has garbage collection, what it does have is RAI, which is kinda like a compile time garbage collection, the compiler will detect when a variable goes out of scope and call its destructor. But sometimes you want a variable to last longer than that and not be deleted when your function ends. That's where pointers are used, and even then the stl

library provides very convenient wrappers like `std::unique_ptr` and `std::shared_ptr` to auto delete it when it is no longer referenced anywhere. `new` is basically C++'s `malloc(size_t)`, and `delete` is `free(void*)`, if you call `new` and never call `delete` you will have a memory leak and that data will never be deleted. So most of the time, there's actually no need for pointers. Instead of calling `new Hash_Handler();` you can just do

`Code Hash_Handler handler;` unlike java, this won't be a null pointer, it will be default constructed just like your pointer, but it will be on the stack instead of on the heap, so at the end of the function/scope the compiler will insert the code to destroy your variable.

The other problems was you having `Hash_Handler* hh;` in your function args, this is not needed in OOP, when you call a member function `obj.Func();` it will call `Func` with a hidden first argument, the object itself, it can be interacted with by this-`>_D[idx]`, or the much simpler way, just naming the member `_D[idx];`

The last is just lack of knowledge of the STL library, which I think is actually quite common, people often reinvent the wheel recreating datastructures and algorithms that stl has implemented, in my opinion, really well.

btw I am selftaught aswell, I was just lucky enough to avoid the OOP aspects until after I read up a bit more about how it works.

also fuck me for not fully reading post, assumed it was same as last weeks, OP if you need the count just change

```
outFile << p.first << std::endl;
```

to

```
outFile << p.first << ":" << p.second << std::endl;
```

the names that appear once were removed because thats what you had in your request

Quote:I only want those who appear more than x amount of times to do resaved into another list of the names

change `const std::size_t x` to `= 1` or just remove the whole branch.

```
Code std::ifstream inFile("NameList.txt");
```

```
std::ofstream outFile("UniqueNameList.txt");
```

```
std::unordered_map<std::string,std::size_t> map;
```

```
std::string line;
```

```
for (getline(inFile, line, '\n'); !inFile.eof(); getline(inFile, line, '\n'))
```

```
++map[line];
```

```
for (const auto& p : map)
```

```
outFile << p.first << ":" << p.second << std::endl;
```

Renovatum - March 8, 2018, 4:20 pm

(Mar 8, 2018 04:48 PM)Gorzoid Wrote: [\[CITE\]](#) Yeah I started with a harsh comment and then thought maybe I should just point out the mistakes, kinda did a lazy ass job of it.

So you use `new` but C++ is not like java in that it has garbage collection, what it does have is RAII, which is kinda like a compile time garbage collection, the compiler will detect when a variable goes out of scope and call its destructor. But sometimes you want a variable to last longer than that and not be deleted when your function ends. That's where pointers are used, and even then the stl library provides very convenient wrappers like `std::unique_ptr` and `std::shared_ptr` to auto delete it when it is no longer referenced anywhere. `new` is basically C++'s `malloc(size_t)`, and `delete` is

free(void*), if you call new and never call delete you will have a memory leak and that data will never be deleted. So most of the time, there's actually no need for pointers. Instead of calling new Hash_Handler(); you can just do

Code Hash_Handler handler; unlike java, this won't be a null pointer, it will be default constructed just like your pointer, but it will be on the stack instead of on the heap, so at the end of the function/scope the compiler will insert the code to destroy your variable.

The other problems was you having Hash_Handler* hh; in your function args, this is not needed in OOP, when you call a member function obj.Func(); it will call Func with a hidden first argument, the object itself, it can be interacted with by this->_D[idx], or the much simpler way, just naming the member _D[idx];

The last is just lack of knowledge of the STL library, which I think is actually quite common, people often reinvent the wheel recreating datastructures and algorithms that stl has implemented, in my opinion, really well.

btw I am selftaught aswell, I was just lucky enough to avoid the OOP aspects until after I read up a bit more about how it works.

also fuck me for not fully reading post, assumed it was same as last weeks, OP if you need the count just change

```
outFile << p.first << std::endl;
to
outFile << p.first << ":" << p.second << std::endl;
```

the names that appear once were removed because thats what you had in your request

change const std::size_t x to = 1 or just remove the whole branch.

```
Code std::ifstream inFile("NameList.txt");
std::ofstream outFile("UniqueNameList.txt");
```

```
std::unordered_map<std::string, std::size_t> map;
std::string line;
```

```
for (getline(inFile, line, '\n'); !inFile.eof(); getline(inFile, line, '\n'))
++map[line];
```

```
for (const auto& p : map)
outFile << p.first << ":" << p.second << std::endl;
```

Damn, coding still confuses the hell out of me with how it works like that. Thanks for doing this so well, and props to you for pointing out his mistakes in a more constructive way.

I hope this works as it's such a useful tool to have :)

aeiro - March 8, 2018, 4:20 pm

(Mar 8, 2018 04:48 PM)Gorzoid Wrote: [CITE](#) Yeah I started with a harsh comment and then thought maybe I should just point out the mistakes, kinda did a lazy ass job of it.

So you use new but C++ is not like java in that it has garbage collection, what it does have is RAII, which is kinda like a compile time garbage collection, the compiler will detect when a variable goes out of scope and call its destructor. But sometimes you want a variable to last longer than that and not be deleted when your function ends. That's where pointers are used, and even then the stl library provides very convenient wrappers like std::unique_ptr and std::shared_ptr to auto delete it when it is no longer referenced anywhere. new is basically C++'s malloc(size_t), and delete is

free(void*), if you call new and never call delete you will have a memory leak and that data will never be deleted. So most of the time, there's actually no need for pointers. Instead of calling new Hash_Handler(); you can just do

Code Hash_Handler handler; unlike java, this won't be a null pointer, it will be default constructed just like your pointer, but it will be on the stack instead of on the heap, so at the end of the function/scope the compiler will insert the code to destroy your variable.

The other problems was you having Hash_Handler* hh; in your function args, this is not needed in OOP, when you call a member function obj.Func(); it will call Func with a hidden first argument, the object itself, it can be interacted with by this->_D[idx], or the much simpler way, just naming the member _D[idx];

The last is just lack of knowledge of the STL library, which I think is actually quite common, people often reinvent the wheel recreating datastructures and algorithms that stl has implemented, in my opinion, really well.

btw I am selftaught aswell, I was just lucky enough to avoid the OOP aspects until after I read up a bit more about how it works.

also fuck me for not fully reading post, assumed it was same as last weeks, OP if you need the count just change

```
outFile << p.first << std::endl;
to
outFile << p.first << ":" << p.second << std::endl;
```

the names that appear once were removed because thats what you had in your request

change const std::size_t x to = 1 or just remove the whole branch.

```
Code std::ifstream inFile("NameList.txt");
std::ofstream outFile("UniqueNameList.txt");
```

```
std::unordered_map<std::string, std::size_t> map;
std::string line;
```

```
for (getline(inFile, line, '\n'); !inFile.eof(); getline(inFile, line, '\n'))
++map[line];
```

```
for (const auto& p : map)
```

```
outFile << p.first << ":" << p.second << std::endl;
Honestly that is extremely helpful. Thank you. This code does exactly what was asked. Looks like i need to read up on this library more.
```