



Phonebook

By:
Alexandru Tentes
Quintin Luong
Jody Doyle
Matthew Rangel-Alvares

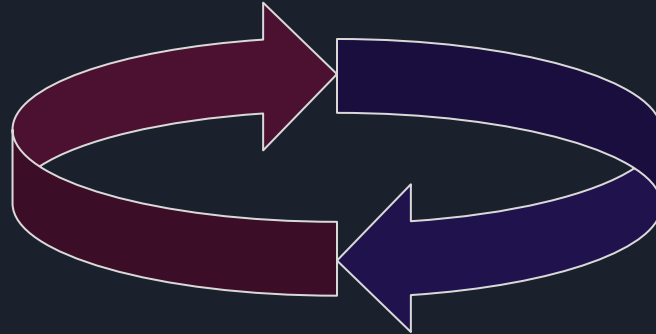


main()

In main, the variables for name and phone number are declared.

The user is prompted for input.

The desired action is carried out, then the program loops back to user input.




Adding Data

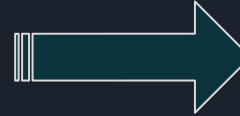
When the user selects 1, the user is prompted to input a name and number for the new contact.

A person struct is created for the details entered, which are assigned.

The function `add_dup` is called to add the data to the `unordered_map`.



```
Type person data:  
Name: Ada Lovelace  
Number: 0800 00 1066
```



p person

Name: Ada Lovelace

Number: 0800 00 1066

Struct stored in memory



add_dup

In add_dup, a temporary string is created.

For each loop, the next digit of the name is added to the temporary string, starting from the start.

Then, the hash map is searched for the string.

- If it does appear, the node is attached to the end of the list appearing in that entry.
- Otherwise, a new entry is created and the list is placed in there.

```
void add_dup(string name, p& person)
{
    int i = 0;
    string tmp = "";
    while (name[i] != '\0')
    {
        tmp += name[i];
        map[tmp].add_back(person);
        i++;
    }
}
```



add_dup example

Given contact name: daVID, phone number: 1000, a total of 5 + 4 linked lists are modified/made:

- | | |
|-----------------|----------------|
| 1. map["d"] | |
| 2. map["da"] | 6. map["1"] |
| 3. map["dav"] | 7. map["10"] |
| 4. map["davi"] | 8. map["100"] |
| 5. map["david"] | 9. map["1000"] |

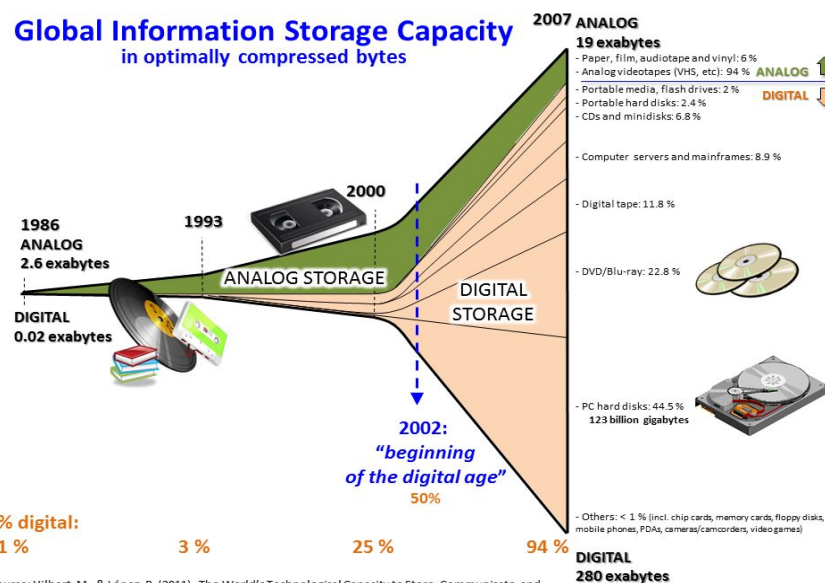
Each list contains a node with name = daVID, number = 1000.

This is less efficient but easy to implement.

when you download more ram on
your pc



Global Information Storage Capacity in optimally compressed bytes



Source: Hilbert, M., & López, P. (2011). The World's Technological Capacity to Store, Communicate, and Compute Information. *Science*. 332(6025). 60–65. <http://www.martinhilbert.net/WorldInfoCapacity.html>



The problem of being
faster than light is that
you can only live in
darkness.

Downloads

Please select the amount of RAM to download:

1GB



Overview

- * 1GB CT12864AA800 Memory
- * 240-pin DIMM
- * DDR2 PC2-6400, CL=6

Was: ~~\$99.99~~ Now: **FREE**

☒ [Download Now](#)

2GB



Overview

- * 2 GB (2 x 1 GB)
- * 240-pin DIMM
- * DDR2 800 MHz (PC2-6400)

Was: ~~\$149.99~~ Now: **FREE**

☒ [Download Now](#)

4GB



Overview

- * 4 GB (2 x 2 GB)
- * 240-pin DIMM
- * DDR2 800 MHz (PC2-6400)

Was: ~~\$199.99~~ Now: **FREE**

☒ [Download Now](#)

<https://downloadmoreram.com/>

Delete Data Functionality, deleteByChar

When deleting data, the function deleteByChar is called.

It prompts the user for input, then shows the user the corresponding entries from the hash map.

The user then enters the index of the entry to be deleted.

The data in the linked list is stored temporarily, then the entries are deleted from the map using these temporary values using delete_all_by_entry.



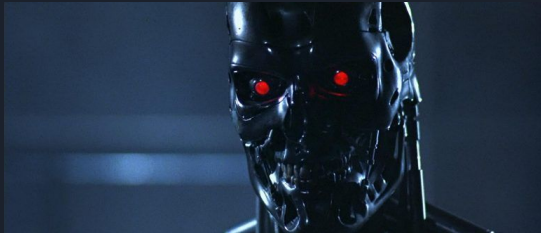
```
void deleteByChar() {  
    string blank;  
    string name = inputName(blank);  
    while (map[name].size > 1) {  
        cout << "Please try again\n";  
        name = inputName(blank);  
    }  
  
    int index;  
    if (map[name].size != 0)  
    {  
        cout << "\nDelete which contact?:\n";  
        cin >> index;  
        while (index < 0 || index >= map[name].size) {  
            cout << "Please try again\n";  
            cin >> index;  
        }  
    }  
  
    if (map[name].size != 0)  
    {  
        string tmp_name = map[name].search_pos(index)->value.name;  
        string tmp_phone = map[name].search_pos(index)->value.phone;  
        delete_all_by_entry(tmp_name, tmp_phone);  
        delete_all_by_entry(tmp_phone, tmp_phone);  
    }  
  
    if (map[name].size == 0)  
        map.erase(name);  
}
```

delete_all_by_entry

When delete by entry is called, an empty string is created which is then populated with the input string, character by character to be deleted in the loop.

For substring, the corresponding entry in the hash map has it's list deleted, with the entry being deleted after.

It calls the search function to find the pointer to the desired node, then deletes that node.



Given that "Zuck" had already been input with number "666", the following nodes would be deleted:

1. `map["z"]`
2. `map["zu"]`
3. `map["zuc"]`
4. `map["zuck"]`
5. `map["6"]`
6. `map["66"]`
7. `map["666"]`

Search


Search is carried out for option 3.

inputName simulates real time search by printing out all the possible names and numbers from the user input.


The user input is updated in real time by detecting normal keys, which add to the search, backspace inputs which removes last character and enter which finalises the search.

To do this, it takes user input and calls printNames and addLetter.

```
Name: David, Number: 0700 500
600
Name: Dana, Number: 0240 999 123
Input name or number:
Da|
```



```
Name: David, Number: 0700 500
600
Input name or number:
Davi|
```



After narrowing the search,
Dana is no longer found



printNames and addLetter

addLetter simply pops off the last letter if it detects a backspace or pushes back the last letter input.

printNames uses the overloaded << operator to print out the linked list elements with a simple function call.

- The overloaded operator defines the specific behaviour when printing the data inside the list.



Import and Export from file

The program can save names and phone numbers to a text file using `saveContact`.

These can then be imported back into the system later by `populate_map_from_file`.

Specific contact details from the file can be removed by providing the details to `removeFromFile`.



fin