PhoneBook

A program to manage a phone book that is a container with a finite capacity and can contain at most n entries.
An entry is a ‘voice’ triple <name, surname, phone number> or a ‘businessVoice’ <name, surname, phone number, email, company>.

The class phoneBook represents the address book of entries. Use an array to store the various entries. It contains all the fundamental methods and also use exceptions where appropriate.

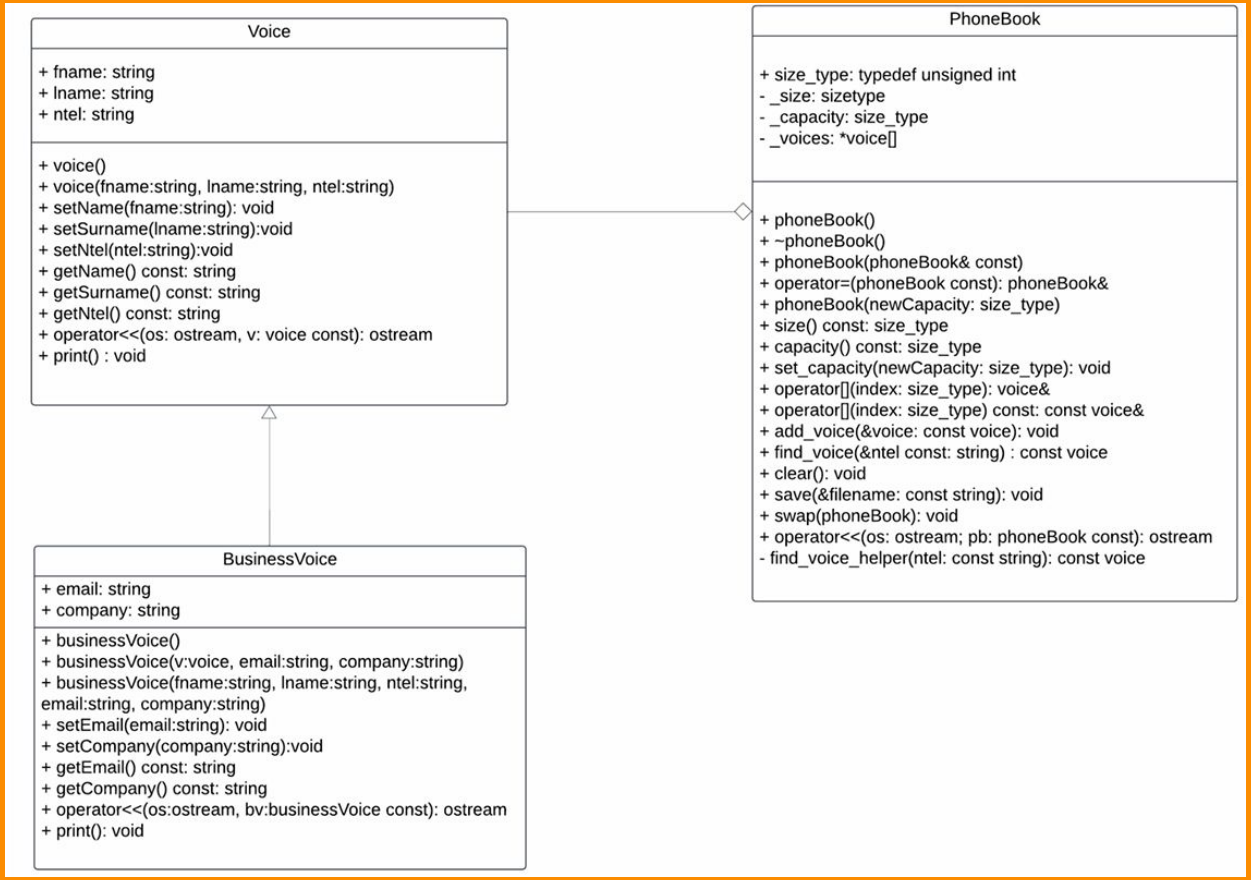
Operation

The dimension of the array must be chosen by the user during the construction phase and through a set_capacity method. If when calling set_capacity the address book already contains entries and the indicated capacity allows it (new capacity greater than the size), keep the entries, otherwise they will all be lost.

It is possible to know the number of entries that the directory can contain (capacity) and the number of entries actually entered (size). You can modify the attribute of the entries, using setters methods.
Use operator[] to read and/or write the i-th entry in the address book. Handle the case in which the address book is full. It is possible to search an entry given the telephone number.

Implement, via operator<<, the printing of the contents of an address book.
Save method that saves the contents of the address book to a text file.

UML


Voice
+ fname: string
+ lname: string
+ ntel: string
+ voice()
+ voice(fname:string, lname:string, ntel:string)
+ setName(fname:string): void
+ setSurname(lname:string):void
+ setNtel(ntel:string):void
+ getName() const: string
+ getSurname() const: string
+ getNtel() const: string
+ operator<<(os: ostream, v: voice const): ostream
+ print() : void
PhoneBook
+ size_type: typedef unsigned int
- _size: sizetype
- _capacity: size_type
- _voices: *voice[]
+ phoneBook()
+ ~phoneBook()
+ phoneBook(phoneBook& const)
+ operator=(phoneBook const): phoneBook&
+ phoneBook(newCapacity: size_type)
+ size() const: size_type
+ capacity() const: size_type
+ set_capacity(newCapacity: size_type): void
+ operator[](index: size_type): voice&
+ operator[](index: size_type) const: const voice&
+ add_voice(&voice: const voice): void
+ find_voice(&ntel const: string) : const voice
+ clear(): void
+ save(&filename: const string): void
+ swap(phoneBook): void
+ operator<<(os: ostream; pb: phoneBook const): ostream
- find_voice_helper(ntel: const string): const voice
BusinessVoice
+ email: string
+ company: string
+ businessVoice()
+ businessVoice(v:voice, email:string, company:string)
+ businessVoice(fname:string, lname:string, ntel:string, email:string, company:string)
+ setEmail(email:string): void
+ setCompany(company:string):void
+ getEmail() const: string
+ getCompany() const: string
+ operator<<(os:ostream, bv:businessVoice const): ostream
+ print(): void

Create an empty phoneBook.
Create a phoneBook capacity 10.
Use the assignment operator on the empty phoneBook, and show it.
Add 4 entries when the capacity of phoneBook is 10, and show them.

```
*** TEST PHONEBOOK ***

Capacity: 0
Size: 0

Capacity: 10
Size: 0

Capacity: 10
Size: 0

Capacity: 10
Size: 4
Printing voice number 1
Name: name1, Surname: surname1, Phone: 11111

Printing voice number 2
Name: name2, Surname: surname2, Phone: 22222
Company: uania@22.com, Email: comapani2g. BusinessVoice: YES

Printing voice number 3
Name: name3, Surname: surname3, Phone: 33333
Company: uania@33.com, Email: comapani3g. BusinessVoice: YES

Printing voice number 4
Name: name4, Surname: surname4, Phone: 44444
```

```cpp
void test_phoneBook(){
    std::cout<<std::endl<<"*** TEST PHONEBOOK ***"<<std::endl<<std::endl;

    phoneBook pb1;  //<empty phoneBook.
    std::cout<<pb1<<std::endl;

    phoneBook pb2(10);  //<capacity is 10.
    std::cout<<pb2<<std::endl;

    pb1=pb2; //Assignment operator.
    std::cout<<pb1<<std::endl;

    phoneBook pb3 = pb2; //Copy constructor.
    try{
        //Add 4 entries (NOTE: Capacity of pb3 is 10)
        pb3.add_voice("name1", "surname1", "11111");
        pb3.add_voice("name2", "surname2", "22222", "uania@22.com", "comapani2g");
        pb3.add_voice("name3", "surname3", "33333", "uania@33.com", "comapani3g");
        pb3.add_voice("name4", "surname4", "44444");
    }
    catch(full_phoneBook_exception &e){
        std::cout<<"!!! ERROR: your phoneBook is full !!!"<<std::endl;
    }
    std::cout<<pb3<<std::endl;
```

Add 6 entries when the capacity of the phoneBook is 5, and show them.
It causes exceptions.

```
!!! ERROR: your phoneBook is full !!!
Capacity: 5
Size: 5
Printing voice number 1
Name: name1, Surname: surname1, Phone: 11111

Printing voice number 2
Name: name2, Surname: surname2, Phone: 22222
Company: uania@22.com, Email: comapani2g. BusinessVoice: YES

Printing voice number 3
Name: name3, Surname: surname3, Phone: 33333
Company: uania@33.com, Email: comapani3g. BusinessVoice: YES

Printing voice number 4
Name: name4, Surname: surname4, Phone: 44444

Printing voice number 5
Name: name5, Surname: surname5, Phone: 55555
Company: uania@55.com, Email: compani5g. BusinessVoice: YES
```

```
phoneBook pb4(5);
try{
    //Add 6 entries (NOTE: Capacity of pb3 is 5)
    pb4.add_voice("name1", "surname1", "11111");
    pb4.add_voice("name2", "surname2", "22222", "uania@22.com", "comapani2g");
    pb4.add_voice("name3", "surname3", "33333", "uania@33.com", "comapani3g");
    pb4.add_voice("name4", "surname4", "44444");
    pb4.add_voice("name5", "surname5", "55555", "uania@55.com", "compani5g");
    pb4.add_voice("name6", "surname6", "66666");
}
catch(full_phoneBook_exception &e){
    std::cout<<"!!! ERROR: your phoneBook is full !!!"<<std::endl;
}
std::cout<<pb4<<std::endl;
```

```
!!! ERROR: entry not found !!!
Capacity: 5
Size: 0

Capacity: 7
Size: 2
Printing voice number 1
Name: name1, Surname: surname1, Phone: 11111

Printing voice number 2
Name: name2, Surname: surname2, Phone: 22222
Company: uania@22.com, Email: comapani2g. BusinessVoice: YES

Capacity: 1
Size: 0
```

```
try{
    voice v = pb3.find_voice("12345");
}
catch(voice_not_found_exception){
    std::cout<<"!!! ERROR: entry not found !!!"<<std::endl;
}

phoneBook pb6(3);
pb6.set_capacity(5);
std::cout<<pb6<<std::endl;
try{
    //Add 6 entries (NOTE: Capacity of pb3 is 5)
    pb6.add_voice("name1", "surname1", "11111");
    pb6.add_voice("name2", "surname2", "22222", "uania@22.com", "comapani2g");
}
catch(full_phoneBook_exception &e){
    std::cout<<"!!! ERROR: your phoneBook is full !!!"<<std::endl;
}
pb6.set_capacity(7);
std::cout<<pb6<<std::endl;

pb6.set_capacity(1);
std::cout<<pb6<<std::endl;
```

Result of searching for an entry that does not exist.

Create a phonebook of capacity 5,
then insert entries and try to increase the size to 7: this is possible because the new capacity is greater than the number of entries in the phonebook (size). If you try to change the capacity to 1 (which is less than the size) the phonebook will be set to 1 the capacity, but you will lose all the entries.

```
*** INTERACTIVE TEST ***

Insert the capacity of phoneBook: 2
Insert an entry
 Name (* for terminate): Francesca
 Surname: Sciacca
 Telephone Number: 32499933333
Insert an entry
 Name (* for terminate): Ludovica
 Surname: Olga
 Telephone Number: 235836499
Insert an entry
 Name (* for terminate): *
Do you want to save the phoneBook (y/n)?y
Nome del file: interactive_test
Printing voice number 1
Name: Francesca, Surname: Sciacca, Phone: 32499933333
Printing voice number 2
Name: Ludovica, Surname: Olga, Phone: 235836499
PhoneBook pb saved.
```

main.cpp uses an address book interactively:

Ask the user how many entries the address book should have and set the address book appropriately.

Ask the user which entries to insert by requesting the data of an entry. If the user inserts an asterisk ('*'), it means that he does not want to insert further entries.

Ask the user if he wants to save the address book and, if so, the name of the file to use. Save the address book and display its contents on the screen.

interactive_test

File Modifica Visualizza

```
Capacity: 2
Size: 2
Voice: Francesca Sciacca 32499933333
Voice: Ludovica Olga 235836499
```