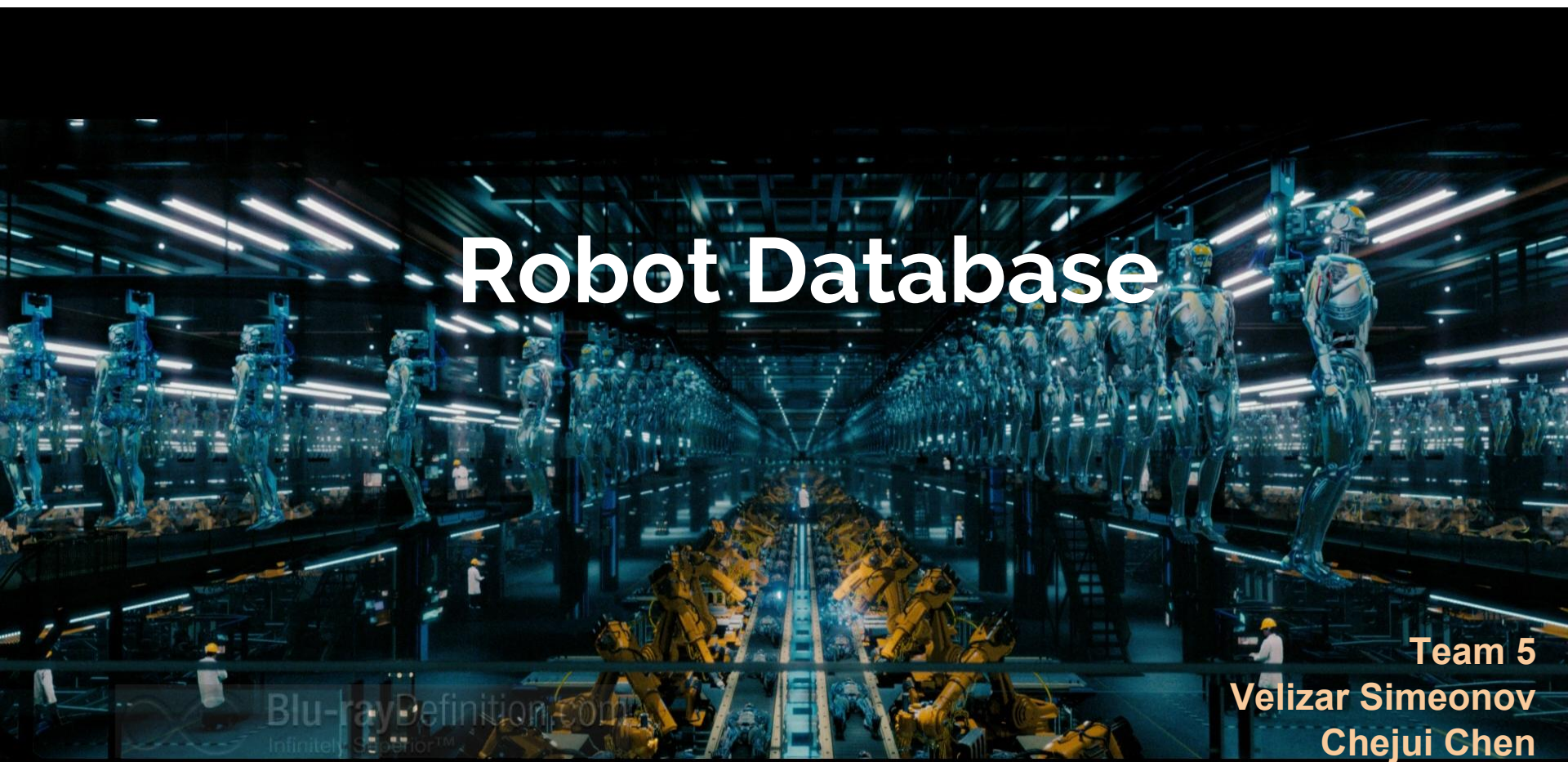
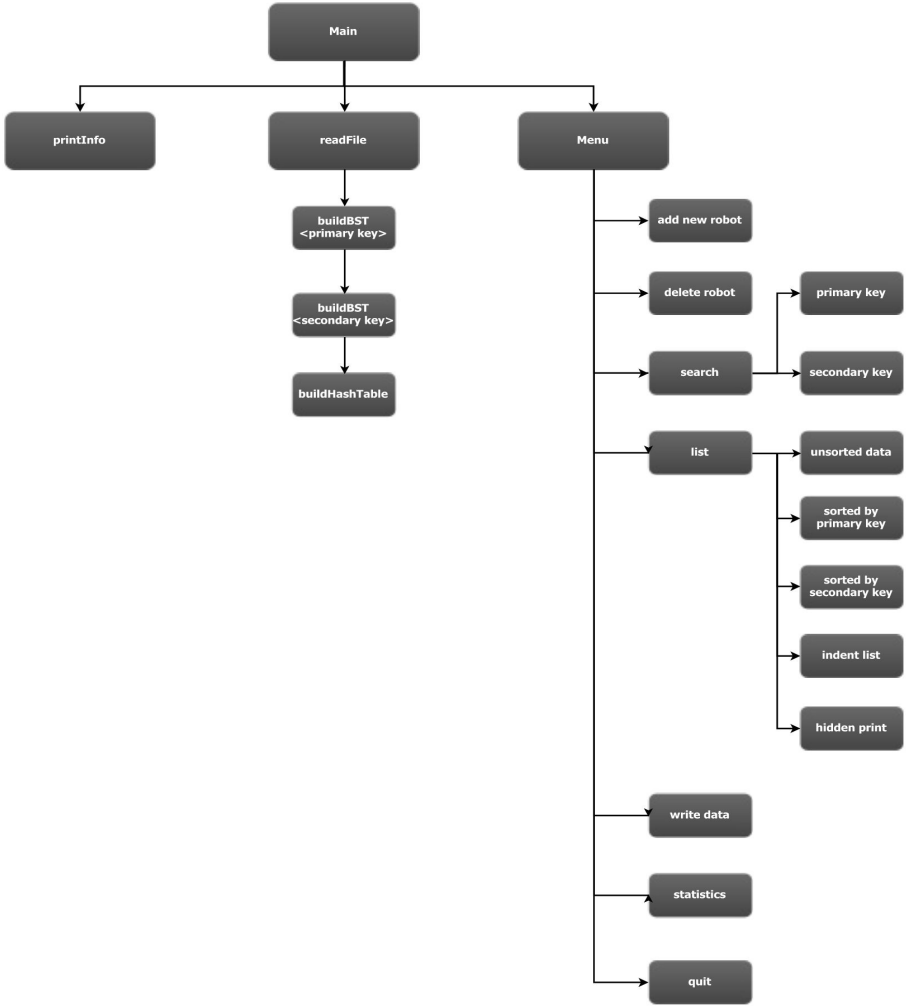


Robot Database

Team 5
Velizar Simeonov
Chejui Chen
Yuchen Jiang
Aijun Qin

Program purpose:

- Store robot data into a text file
- Sort data by its serial number, reduce collision
- Functions for user to manage the database (insert, remove, search, print, etc)


Main
printInfo
readFile
Menu
buildBST <primary key>
buildBST <secondary key>
buildHashTable
add new robot
delete robot
search
list
write data
statistics
quit
primary key
secondary key
unsorted data
sorted by primary key
sorted by secondary key
indent list
hidden print

Project Components

- Robot Class
- BST
- Queue
- Hashtable
- primeHelper

- basic file io
- Diagrams (DSD, SC, UML)

Robot class

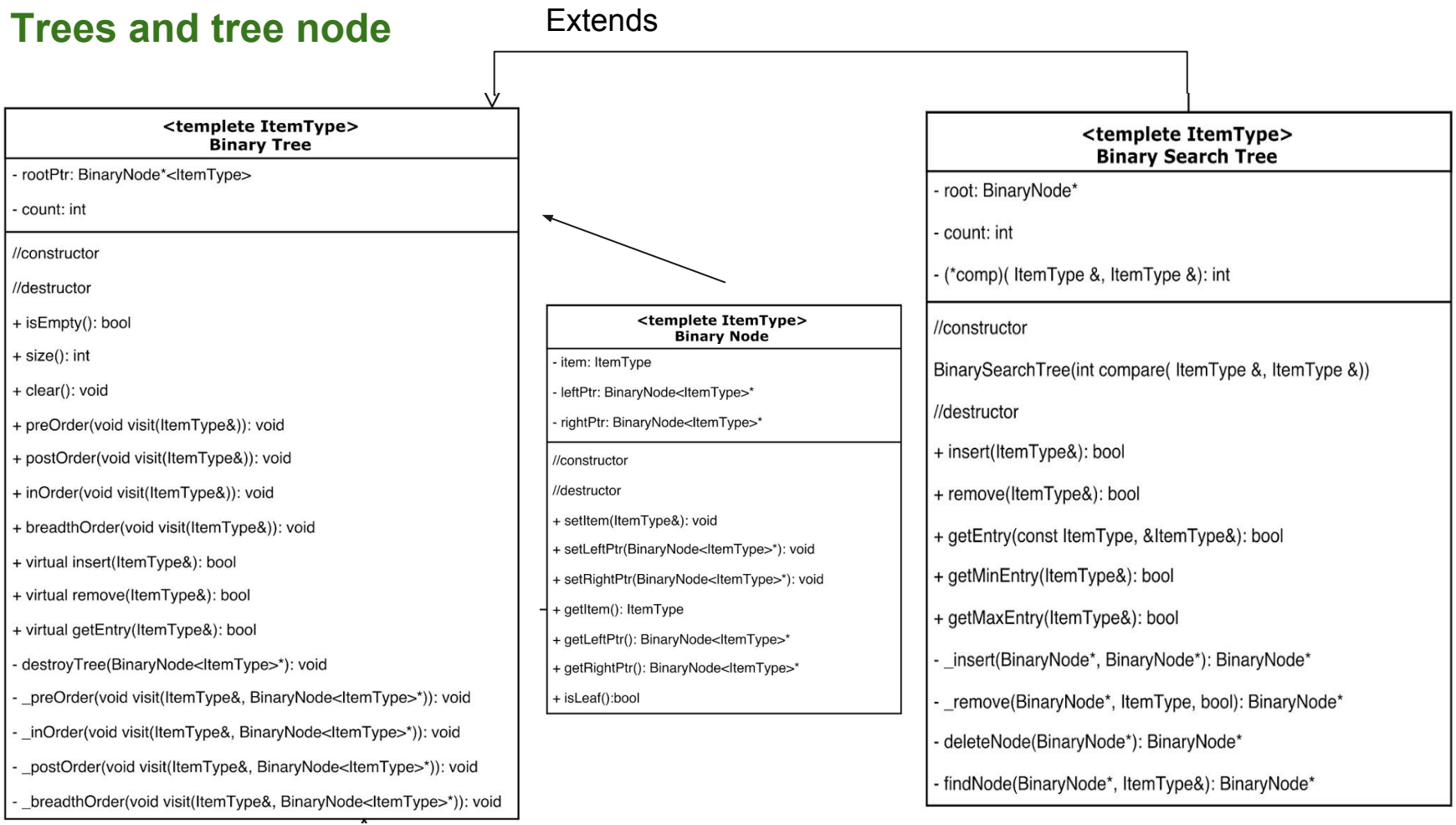
Robot
- serialnumber: string - model: string - alias: string - comment: string - productiondate: string

//constructor
+ Robot(string, string, string, string, string)
+ Robot(const Robot &obj)
//destructor
+ getSerialNumber(): String
+ getModel(): String
+ getAlias(): String
+ getComment(): String
+ getProductionDate(): String
+ setAlias(string): void
+ setComment(string): void
+ setProductionDate(string): void
+ operator==(const Robot &r) const: bool
+ operator<(const Robot &r) const: bool
+ operator>(const Robot &r) const: bool
// friend
+ operator<<(ostream &, const Robot &): ostream &

Queue(Multiple uses)

- Temporary storage from file input
- Display multiple robots with the same model key
- Remove multiple robots under the same model key

Trees and tree node



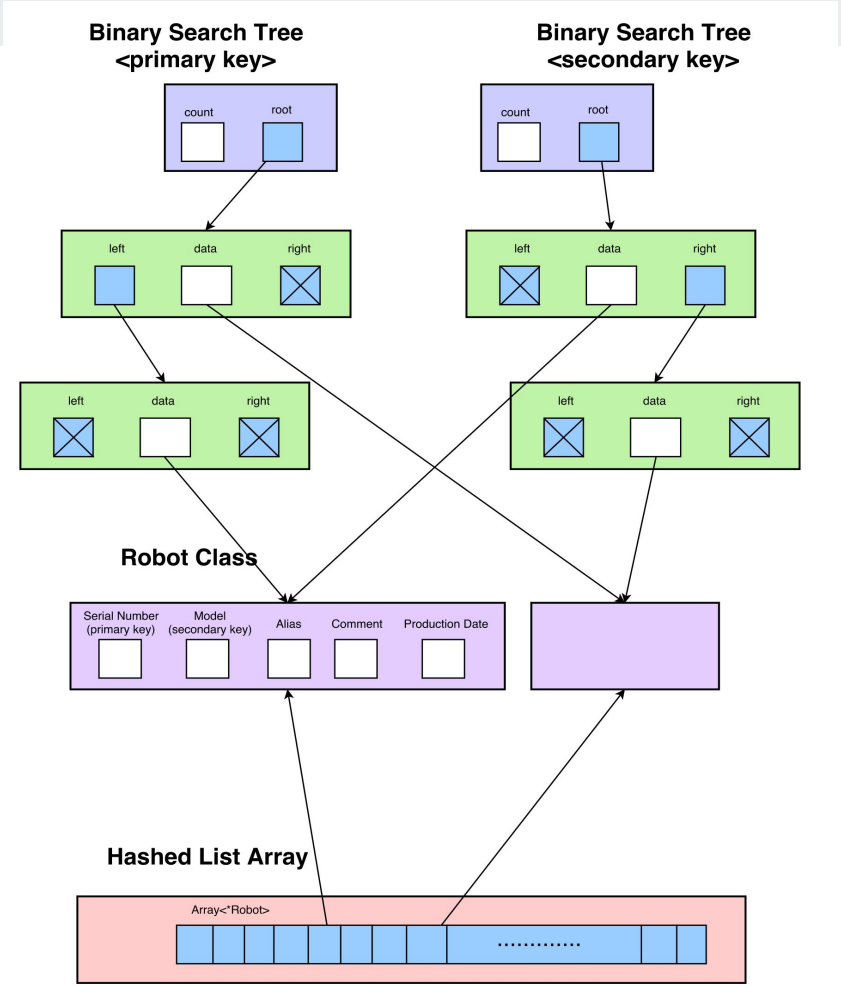
DATA STRUCTURES

hashed list array (contiguous memory)

- hash of primary key (must be string)
- hash with 2 functions ("good" and "bad")

2 binary search trees

- 1 bst for primary key
- 1 bst for secondary key

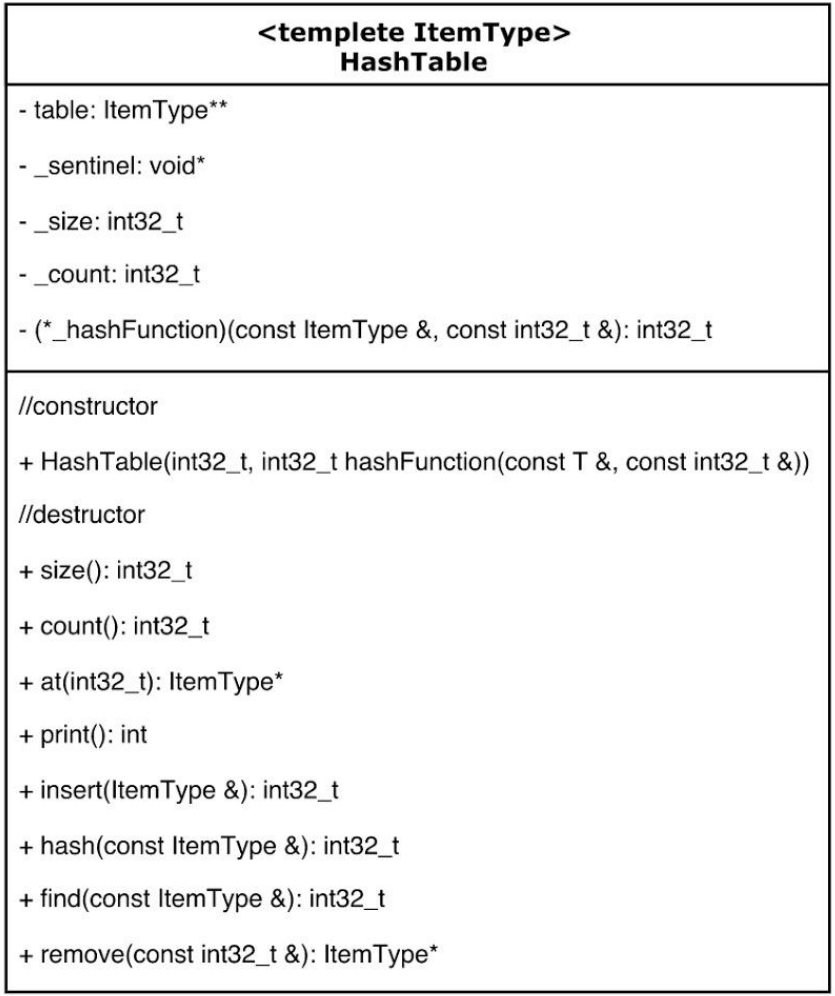


HashTable

A Good version and Bad version

Additional: primeHelper .h

To find its closest prime number to given input integer



Hash algorithm

```
algorithm robotHash (Robot robot, int tableSize)
    Int sum = 0
    Char *cPtr = robot.snr
    while(*cPtr)
        sum += (*cPtr)^3
        cPtr++
    End while
    Return sum % tableSize
End algorithm
```

Collisions Algorithm

will be resolved in "linear probe" like method
collision resolving method alternates between positive and negative index offsets from home address

```
if home is full

        loop (offset = 1)

                try home + offset

                        exit loop on success

                if offset < 0

                        offset * -1 + 1

                Else    (offset + 1) * -1
```

Thank you