

Mechatronic Engineering

Object Oriented Programing and Software Engineering
Laboratory instruction 12
C++ introduction

AGH Kraków, 2020

Materials created for educational purposes.
Dedicated for students attending Software Engineering course.
Author would appreciate any feedback regarding errors of any kind found in the instruction script.
Please report those to the following email address: danielt@agh.edu.pl

Contents

1	Linked lists	4
1.1	Singly linked lists	4

1 Linked lists

Linked list is a dynamic structure of objects (data). This allows one to freely change its size while the program is running. The only limitation is computer memory. List is made up of connected elements. There are two types of lists:

- Singly linked lists,
- Doubly linked lists.

1.1 Singly linked lists

Each element of this list is connected only to the next element of this list. This type of list can be navigated only in one direction from the head (first element) to the tail (last element). The operation diagram of a singly linked list is shown in the figure 1.

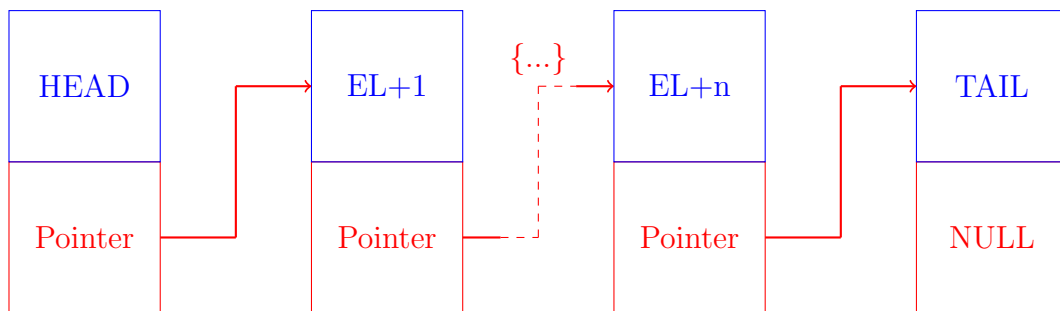


Figure 1: Singly linked list schematic

```
1 #include <iostream>
2 #include <string.h>
3
4 using namespace std;
5
6 //Element declaration
7 struct lst_el{
8     lst_el * next;
9     int key;
10    string name;
11
12 };
13
```

```

14
15 //definition of singly listed list class
16 class TsingleList{
17     lst_el * head, * tail;
18     int cnt;
19
20 public:
21     TsingleList(){
22         head = tail = NULL;
23         cnt = 0;
24     }
25
26     ~TsingleList(){
27         lst_el * el;
28
29         while(head){
30             el = head->next;
31             delete head;
32             head = el;
33         }
34     }
35
36 //Method returning list size
37     unsigned size(){
38         return cnt;
39     }
40
41 //Method adding an element at the front of the list
42     lst_el * push_head(lst_el * el){
43         el->next = head;
44         head = el;
45         if(!tail) tail = head;
46         cnt++;
47         return head;
48     }
49
50 //Method adding an element at the end of the list
51     lst_el * push_tail(lst_el * el){
52         if(tail) tail->next = el;
53         el->next = NULL;
54         tail = el;
55         if(!head) head = tail;
56         cnt++;

```

```

57     return tail;
58 }
59
60 //Method adding an element (el1) after an element (el2)
61 lst_el * insert(lst_el * el1, lst_el * el2){
62     el1->next = el2->next;
63     el2->next = el1;
64     if(!(el1->next)) tail = el1;
65     cnt++;
66     return el1;
67 }
68
69 //Method deleting the first element of the list
70 lst_el * rmHead(){
71     lst_el * el;
72
73     if(head){
74         el = head;
75         head = head->next;
76         if(!head) tail = NULL;
77         cnt--;
78         return el;
79     }
80     else return NULL;
81 }
82
83 //Method deleting the last element of the list
84 lst_el * rmTail(){
85     lst_el * el;
86
87     if(tail){
88         el = tail;
89         if(el == head) head = tail = NULL;
90         else{
91             tail = head;
92             while(tail->next != el) tail = tail->next;
93             tail->next = NULL;
94         }
95         cnt--;
96         return el;
97     }
98     else return NULL;
99 }

```

```

100
101 //Method deleting the el element of the list
102 lst_el * erase(lst_el * el){
103     lst_el * el1;
104
105     if(el == head) return rmHead();
106     else{
107         el1 = head;
108         while(el1->next != el) el1 = el1->next;
109         el1->next = el->next;
110         if(!(el1->next)) tail = el1;
111         cnt--;
112         return el;
113     }
114 }
115
116 //Method returns nth element of the list
117 lst_el * index(int n){
118     lst_el * el;
119
120     if((!n) || (n > cnt)) return NULL;
121     else if(n == cnt) return tail;
122     else{
123         el = head;
124         while(--n) el = el->next;
125         return el;
126     }
127 }
128
129 //Methods used to display data stored in the list
130 void showKeys(){
131     lst_el * el;
132
133     if(!head) cout << "List is empty." << endl;
134     else{
135         el = head;
136         while(el){
137             cout << el->key << " ";
138             el = el->next;
139         }
140         cout << endl;
141     }
142 }

```

```

143
144     void showNames(){
145         lst_el * el;
146
147         if(!head) cout << "List is empty." << endl;
148         else{
149             el = head;
150             while(el){
151                 cout << el->name << " ";
152                 el = el->next;
153             }
154             cout << endl;
155         }
156     }
157
158     void showElements(){
159         lst_el * el;
160
161         if(!head) cout << "List is empty." << endl;
162         else{
163             el = head;
164             while(el){
165                 cout << "Name: " << el->name << ", key: " << el->key <<";
166                 ";
167                 el = el->next;
168             }
169             cout << endl;
170         }
171     };
172
173
174
175     int main(){
176         TsingleList sl;
177         lst_el * p;
178         int i;
179
180         cout << "List should be empty : "; sl.showKeys();
181
182         //This will add 5 elements at the front of the list
183         for(i = 1; i <= 5; i++){
184             p = new lst_el;

```



```

185     p->key = i;
186     cout << "Enter name of the element: ";
187     cin >> p->name;
188     sl.push_head(p);
189 }
190
191 cout << "Now there should be "<< sl.size() <<" elements in the
    list: "; sl.showElements(); cout << endl;
192 cout << "Program also displays single fields of the elements\n";
193 cout << "Keys: "; sl.showKeys(); cout << endl;
194 cout << "Names: "; sl.showNames(); cout << endl;
195
196 //This will add 5 elements at the back of the list
197 for(i = 1; i <= 5; i++){
198     p = new lst_el;
199     p->key = i;
200     p->name = to_string(i);
201     sl.push_tail(p);
202 }
203
204 cout << "Keys of the list: "; sl.showKeys();
205 cout << "Names of the list: "; sl.showNames();
206
207 //Removing first element
208 sl.rmHead();
209
210 cout << "Keys of the list after operations: "; sl.showKeys();
211
212 //Removing last element
213 sl.rmTail();
214
215 cout << "Keys of the list after operations: "; sl.showKeys();
216
217 //Removing n-th element
218 delete sl.erase(sl.index(3));
219
220 cout << "Keys of the list after operations: "; sl.showKeys();
221
222 //Another way of removing an element
223 delete sl.erase(sl.index(sl.size() - 1));
224
225 cout << "Keys of the list after operations: "; sl.showKeys();
226

```

```

227 //Adding new element after 4th element
228     p = new lst_el;
229     p->key = 9;
230     p->name = to_string(9);
231     sl.insert(p,sl.index(4));
232
233     cout << "Keys of the list after operations: "; sl.showKeys();
234
235 //learing the list
236     while(sl.size()) sl.rmHead();
237
238     cout << "Empty list: "; sl.showElements();
239
240     cout << endl << endl;
241
242
243     return 0;
244 }

```

Task

Based on the informations provided in this manual, please improve the simple RPG character creation program.

Program requirements:

1. Add a history of the last 10 fights of the selected hero displayed as a list.