

Mechatronic Engineering

Object Oriented Programing and Software Engineering
Laboratory instruction 15
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	Doubly 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 Doubly linked lists

Each element of the doubly linked list is connected to the next and previous element. This type of list can be passed in both directions - from the first element (head) to the last (tail) or from the last to the first. More importantly, the two-way list allows one to go back inside it. The operation diagram is presented in figure 1.

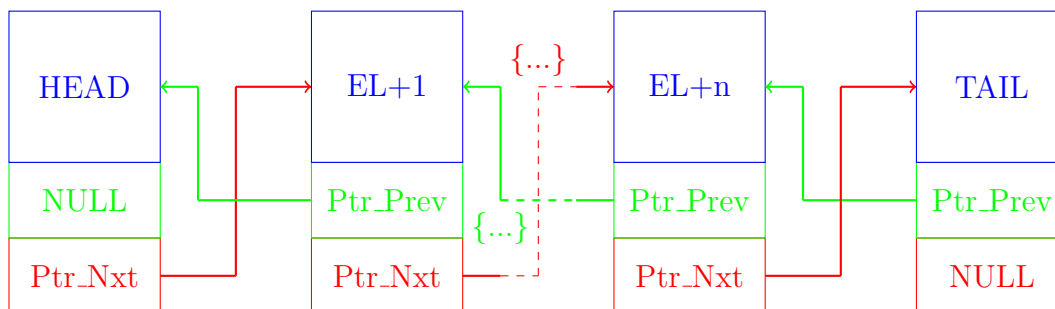


Figure 1: Doubly linked list schematic

Example [1]:

```
1 #include<iostream>
2 #include<cstdio>
3 #include<cstdlib>
4 /*
5  * Node Declaration
6  */
7
8 using namespace std;
9
10 struct node
11 {
```

```

12     int info;
13     struct node *next;
14     struct node *prev;
15 }*start;
16
17 /*
18  Class Declaration
19  */
20
21 class double_llist
22 {
23     public:
24         void create_list(int value);
25         void add_begin(int value);
26         void add_after(int value, int position);
27         void delete_element(int value);
28         void search_element(int value);
29         void display_dlist();
30         void count();
31         void reverse();
32         double_llist()
33         {
34             start = NULL;
35         }
36 };
37
38 /*
39  * Main: Conatins Menu
40  */
41 int main()
42 {
43     int choice, element, position;
44     double_llist dl;
45     while (1)
46     {
47         cout<<endl<<"-----"<<endl;
48         cout<<endl<<"Operations on Doubly linked list"<<endl;
49         cout<<endl<<"-----"<<endl;
50         cout<<"1.Create Node"<<endl;
51         cout<<"2.Add at begining"<<endl;
52         cout<<"3.Add after position"<<endl;
53         cout<<"4.Delete"<<endl;
54         cout<<"5.Display"<<endl;

```

```

55     cout<<"6.Count"<<endl;
56     cout<<"7.Reverse"<<endl;
57     cout<<"8.Quit"<<endl;
58     cout<<"Enter your choice : ";
59     cin>>choice;
60     switch ( choice )
61     {
62     case 1:
63         cout<<"Enter the element: ";
64         cin>>element;
65         dl.create_list(element);
66         cout<<endl;
67         break;
68     case 2:
69         cout<<"Enter the element: ";
70         cin>>element;
71         dl.add_begin(element);
72         cout<<endl;
73         break;
74     case 3:
75         cout<<"Enter the element: ";
76         cin>>element;
77         cout<<"Insert Element after postion: ";
78         cin>>position;
79         dl.add_after(element, position);
80         cout<<endl;
81         break;
82     case 4:
83         if (start == NULL)
84         {
85             cout<<"List empty,nothing to delete"<<endl;
86             break;
87         }
88         cout<<"Enter the element for deletion: ";
89         cin>>element;
90         dl.delete_element(element);
91         cout<<endl;
92         break;
93     case 5:
94         dl.display_dlist();
95         cout<<endl;
96         break;
97     case 6:

```

```

98         dl.count();
99         break;
100     case 7:
101         if (start == NULL)
102             {
103                 cout<<"List empty,nothing to reverse"<<endl;
104                 break;
105             }
106         dl.reverse();
107         cout<<endl;
108         break;
109     case 8:
110         exit(1);
111     default:
112         cout<<"Wrong choice"<<endl;
113     }
114 }
115 return 0;
116 }
117
118 /*
119  * Create Double Link List
120  */
121 void double_llist::create_list(int value)
122 {
123     struct node *s, *temp;
124     temp = new(struct node);
125     temp->info = value;
126     temp->next = NULL;
127     if (start == NULL)
128     {
129         temp->prev = NULL;
130         start = temp;
131     }
132     else
133     {
134         s = start;
135         while (s->next != NULL)
136             s = s->next;
137         s->next = temp;
138         temp->prev = s;
139     }
140 }

```

```

141
142  /*
143  * Insertion at the beginning
144  */
145  void double_llist::add_begin(int value)
146  {
147      if (start == NULL)
148      {
149          cout<<"First Create the list."<<endl;
150          return;
151      }
152      struct node *temp;
153      temp = new(struct node);
154      temp->prev = NULL;
155      temp->info = value;
156      temp->next = start;
157      start->prev = temp;
158      start = temp;
159      cout<<"Element Inserted"<<endl;
160  }
161
162  /*
163  * Insertion of element at a particular position
164  */
165  void double_llist::add_after(int value, int pos)
166  {
167      if (start == NULL)
168      {
169          cout<<"First Create the list."<<endl;
170          return;
171      }
172      struct node *tmp, *q;
173      int i;
174      q = start;
175      for (i = 0; i < pos - 1; i++)
176      {
177          q = q->next;
178          if (q == NULL)
179          {
180              cout<<"There are less than ";
181              cout<<pos<<" elements."<<endl;
182              return;
183          }

```



```

184     }
185     tmp = new(struct node);
186     tmp->info = value;
187     if (q->next == NULL)
188     {
189         q->next = tmp;
190         tmp->next = NULL;
191         tmp->prev = q;
192     }
193     else
194     {
195         tmp->next = q->next;
196         tmp->next->prev = tmp;
197         q->next = tmp;
198         tmp->prev = q;
199     }
200     cout<<"Element Inserted"<<endl;
201 }
202
203 /*
204  * Deletion of element from the list
205  */
206 void double_llist::delete_element(int value)
207 {
208     struct node *tmp, *q;
209     /*first element deletion*/
210     if (start->info == value)
211     {
212         tmp = start;
213         start = start->next;
214         start->prev = NULL;
215         cout<<"Element Deleted"<<endl;
216         free(tmp);
217         return;
218     }
219     q = start;
220     while (q->next->next != NULL)
221     {
222         /*Element deleted in between*/
223         if (q->next->info == value)
224         {
225             tmp = q->next;
226             q->next = tmp->next;

```

```

227         tmp->next->prev = q;
228         cout<<"Element Deleted"<<endl;
229         free(tmp);
230         return;
231     }
232     q = q->next;
233 }
234 /*last element deleted*/
235 if (q->next->info == value)
236 {
237     tmp = q->next;
238     free(tmp);
239     q->next = NULL;
240     cout<<"Element Deleted"<<endl;
241     return;
242 }
243 cout<<"Element " <<value<<" not found"<<endl;
244 }
245
246 /*
247  * Display elements of Doubly Link List
248  */
249 void double_llist::display_dlist()
250 {
251     struct node *q;
252     if (start == NULL)
253     {
254         cout<<"List empty,nothing to display"<<endl;
255         return;
256     }
257     q = start;
258     cout<<"The Doubly Link List is :"<<endl;
259     while (q != NULL)
260     {
261         cout<<q->info<<" <-> ";
262         q = q->next;
263     }
264     cout<<"NULL"<<endl;
265 }
266
267 /*
268  * Number of elements in Doubly Link List
269  */

```

```

270 void double_llist::count()
271 {
272     struct node *q = start;
273     int cnt = 0;
274     while (q != NULL)
275     {
276         q = q->next;
277         cnt++;
278     }
279     cout<<"Number of elements are: "<<cnt<<endl;
280 }
281
282 /*
283  * Reverse Doubly Link List
284  */
285 void double_llist::reverse()
286 {
287     struct node *p1, *p2;
288     p1 = start;
289     p2 = p1->next;
290     p1->next = NULL;
291     p1->prev = p2;
292     while (p2 != NULL)
293     {
294         p2->prev = p2->next;
295         p2->next = p1;
296         p1 = p2;
297         p2 = p2->prev;
298     }
299     start = p1;
300     cout<<"List Reversed"<<endl;
301 }

```

Bibliography

[1] <https://www.sanfoundry.com/cpp-program-implement-doubly-linked-list/>.

Task

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

Program requirements:

1. Implement items into your program. It should be stored inside an additional text file. Every item should have its name and bonus abilities (e.g. Sword of ice, rises strength of the user by 2, adds 5 ice damage in combat; Armor of spikes, reduces by 5 damage taken by the character; Potion of health, replenishes 20 hp) depending on your game mechanics.
2. Add a functionality of item dropping after a monster is slain. Give specific items different drop rates.
3. Create character inventory with the doubly linked list. Make items stackable if there are couple of the same type.
4. Add a functionality for equipping items from character inventory.