**Лабораторна робота № 8. Списки**

***Мета:*** одержати навички та закріпити знання при виконанні операцій на списках.

**1 Вимоги**

**1.1 Розробник**

* Макаренко Владислав Олександрович
* Студент 1-го курсу
* Групи КІТ-120а

**1.2 Загальне завдання**

Передбачити такі функції:

* додавання елементів у список та підсписок (при додаванні елемента у головний список дадається і відповідний підсписок);
* видалення елементів зі списку та підсписків (при видаленні елемента з головного списка видаляється і пов’язаний з ним підсписок);
* видачк вмісту списків та підсписків у консоль;
* видалення списків.

**1.2 Загальне завдання**

**Список** Назви країн

**Підсписок** Міста країни

**2 Описи програм**

**Код програми**

#include <iostream>  
#include <string>  
  
using namespace std;  
  
struct CityElement{  
  
 CityElement \*next;  
 string name;  
};  
  
struct CityList {  
  
 CityElement \*head;  
};  
  
struct CountryElement{  
  
 CityList \*cities\_list;  
 CountryElement \*next;  
 string name;  
};  
  
struct CountriesList{  
  
 CountryElement \*head;  
};  
  
void InitCountryList(struct CountriesList &*country\_list*){  
  
 *country\_list*.head = new struct CountryElement;  
 *country\_list*.head->next = NULL;  
}  
  
void AddCountry(struct CountriesList &*country\_list*, string *name*){  
  
 struct CountryElement \*new\_country = new struct CountryElement;  
 new\_country->name = name;  
 new\_country->next = NULL;  
 new\_country->cities\_list = NULL;  
 struct CountryElement \*current\_last = *country\_list*.head->next;  
 if (current\_last == NULL)  
 *country\_list*.head->next = new\_country;  
 else  
 {  
 while (current\_last->next != NULL)  
 {  
 current\_last = current\_last->next;  
 }  
 current\_last->next = new\_country;  
 }  
}  
  
void AddCity(struct CountriesList &*country\_list*, string *country\_name*, string *name*){  
  
 struct CountryElement \*current = *country\_list*.head->next;  
 while (current->name != country\_name)  
 {  
 current = current->next;  
 if (current == NULL)  
 {  
 cout << "Not country with name " << country\_name << endl;  
 return;  
 }  
 }  
 if (current->cities\_list == NULL)  
 {  
 current->cities\_list = new struct CityList;  
 current->cities\_list->head = new struct CityElement;  
 current->cities\_list->head->next = NULL;  
 }  
 struct CityElement \*new\_city = new struct CityElement;  
 new\_city->next = NULL;  
 new\_city->name = name;  
 struct CityElement \*current\_handed\_over\_object = current->cities\_list->head->next;  
 if (current\_handed\_over\_object == NULL)  
 {  
 current->cities\_list->head->next = new\_city;  
 }  
 else  
 {  
 while (current\_handed\_over\_object->next != NULL)  
 {  
 current\_handed\_over\_object = current\_handed\_over\_object->next;  
 }  
 current\_handed\_over\_object->next = new\_city;  
 }  
}  
  
void DeleteCountry(struct CountriesList &*country\_list*, string *country\_name*)  
{  
 struct CountryElement \*deleted = *country\_list*.head->next;  
 struct CountryElement \*prev = *country\_list*.head->next;  
 if (deleted == NULL)  
 {  
 cout << "Not country with name " << country\_name << endl;  
 return;  
 }  
 while (deleted->name != country\_name)  
 {  
 prev = deleted;  
 deleted = deleted->next;  
 if (deleted == NULL)  
 {  
 cout << "Not country with name " << country\_name << endl;  
 return;  
 }  
 }  
 if (deleted == *country\_list*.head->next)  
 *country\_list*.head->next = deleted->next;  
 else  
 {  
 prev->next = deleted->next;  
 }  
  
 struct CityElement \*current\_city = NULL;  
 if (deleted->cities\_list != NULL)  
 {  
 if (deleted->cities\_list->head != NULL)  
 {  
 current\_city = deleted->cities\_list->head->next;  
 }  
 }  
  
 if (current\_city != NULL)  
 {  
 struct CityElement \*deleted\_handed\_over\_object\_element = current\_city;  
 while (current\_city != NULL)  
 {  
 current\_city = current\_city->next;  
 delete (deleted\_handed\_over\_object\_element);  
 deleted\_handed\_over\_object\_element = current\_city;  
 }  
 if (deleted->cities\_list->head != NULL)  
 delete (deleted->cities\_list->head);  
 if (deleted->cities\_list != NULL)  
 delete (deleted->cities\_list);  
 }  
 delete (deleted);  
}  
  
void DeleteCityFromCountry(struct CountriesList &*country\_list*, string *country\_name*, string *city\_name*)  
{  
 struct CountryElement \*country\_from\_deleted = *country\_list*.head->next;  
 if (country\_from\_deleted != NULL)  
 {  
 while (country\_from\_deleted->name != country\_name)  
 {  
 country\_from\_deleted = country\_from\_deleted->next;  
 if (country\_from\_deleted == NULL)  
 {  
 cout << "No country with name: " << country\_name << endl;  
 return;  
 }  
 }  
 if (country\_from\_deleted->cities\_list != NULL)  
 {  
 struct CityElement \*deleted\_city = country\_from\_deleted->cities\_list->head->next;  
 struct CityElement \*prev = NULL;  
 while (deleted\_city->name != city\_name)  
 {  
 prev = deleted\_city;  
 deleted\_city = deleted\_city->next;  
 if (deleted\_city == NULL)  
 break;  
 }  
 if (deleted\_city != NULL)  
 {  
 if (deleted\_city == country\_from\_deleted->cities\_list->head->next)  
 {  
 country\_from\_deleted->cities\_list->head->next = deleted\_city->next;  
 }  
 else  
 {  
 prev->next = deleted\_city->next;  
 }  
 delete (deleted\_city);  
 }  
 else  
 {  
 cout << "Not city in country: " << country\_name << " with name: " << city\_name << endl;  
 return;  
 }  
 }  
 else  
 {  
 cout << "Not city in country: " << country\_name << " with name: " << city\_name << endl;  
 return;  
 }  
 }  
 else  
 {  
 cout << "Not country with name: " << country\_name << endl;  
 }  
}  
  
void DeleteCountriesList(struct CountriesList &*country\_list*)  
{  
 struct CountryElement \*current = *country\_list*.head->next;  
 if (current != NULL)  
 {  
 while (current != NULL)  
 {  
 string deleted\_name = current->name;  
 current = current->next;  
 DeleteCountry(country\_list, deleted\_name);  
 }  
 }  
 delete (*country\_list*.head);  
 delete (&*country\_list*);  
}  
  
void PrintCountriesList(struct CountriesList &*country\_list*)  
{  
 struct CountryElement \*current\_country = *country\_list*.head->next;  
 if (current\_country == NULL)  
 {  
 cout << "Not countries" << endl;  
 }  
 else  
 {  
 while (current\_country != NULL)  
 {  
 cout << "Country: " << current\_country->name << endl;  
 if (current\_country->cities\_list != NULL)  
 {  
 if (current\_country->cities\_list->head != NULL)  
 {  
 struct CityElement \*current\_city = current\_country->cities\_list->head->next;  
 if (current\_city != NULL)  
 {  
 cout << "Cities: " << endl;  
 while (current\_city != NULL)  
 {  
 cout << "- " << current\_city->name << endl;  
 current\_city = current\_city->next;  
 }  
 }  
 }  
 }  
 current\_country = current\_country->next;  
 }  
 }  
}  
  
int main()  
{  
 struct CountriesList \*country\_list = new struct CountriesList;  
 InitCountryList(\*country\_list);  
 AddCountry(\*country\_list, "Ukraine");  
 AddCountry(\*country\_list, "USA");  
 PrintCountriesList(\*country\_list);  
 cout << endl << "Add cities: " << endl;  
 AddCity(\*country\_list, "Ukraine", "Kharkiv");  
 AddCity(\*country\_list, "Ukraine", "Kiev");  
 AddCity(\*country\_list, "Ukraine", "Lviv");  
 AddCity(\*country\_list, "USA", "Washington");  
 AddCity(\*country\_list, "USA", "California");  
 PrintCountriesList(\*country\_list);  
 cout << endl << "Delete the country:" << endl;  
 DeleteCountry(\*country\_list, "USA");  
 PrintCountriesList(\*country\_list);  
 cout << endl << "Delete the city:" << endl;  
 DeleteCityFromCountry(\*country\_list, "Ukraine", "Lviv");  
 PrintCountriesList(\*country\_list);  
 AddCountry(\*country\_list, "England");  
 cout << endl << "Add country: " << endl;  
 PrintCountriesList(\*country\_list);  
 cout << endl << "Delete the list:" << endl;;  
 DeleteCountriesList(\*country\_list);  
 PrintCountriesList(\*country\_list);  
 return 0;  
}

**Результати виконання програми**

За алгоритмом коду демонструємо роботу програми (див. рис. 1).

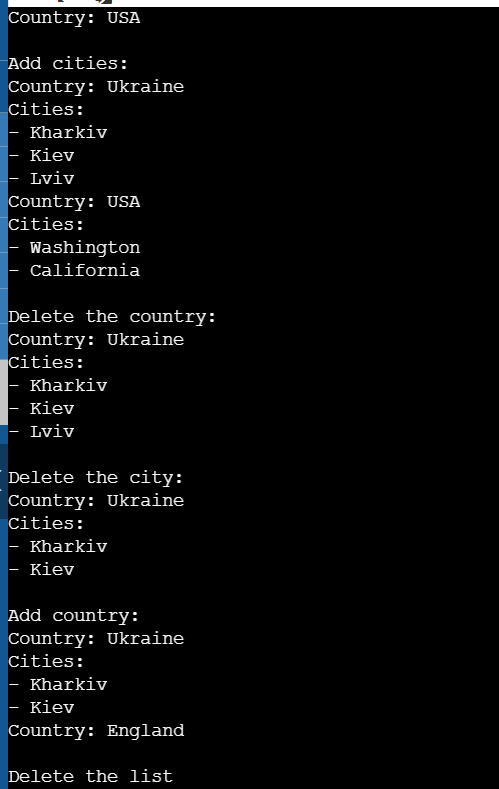


Рисунок 1 – Результати роботи першого кроку

**Висновок:** на цій лабораторній роботі ми одержали навички та закріпили знання при виконанні операцій на списках.