Санкт-Петербургский государственный
университет телекоммуникаций
им. Профессора М.А. Бонч-Бруевича
Факультет ИСиТ

Дисциплина: Технологии программирования

Отчёт по лабораторной работе №6
Разработка шаблона класса для работы с динамическими структурами данных

Выполнил:
студент группы ИСТ-666
Ерохин Б. А.

Проверил:
Медведев В. А.

Санкт-Петербург
2014

Цель

Получение практических навыков создания шаблонов классов и динамических структур данных, разработки и отладки объектно-ориентированных программ.

Задание

A) Спроектировать шаблон класса dynamo для реализации односвязного линейного списка.

B) При использовании шаблона класса в качестве данных для элемента односвязного списка использовать структурную переменную, содержащую следующую персональную информацию:

+) личный номер;

+) фамилия;

+) имя;

+) возраст;

+) страна проживания;

+) профессия;

+) хобби;

+) номер телефона.

C) Для представления символьных строк в элементе списка использовать объекты класса String, спроектированного в лабораторной работе №6.

D) Создать список из 10-15 элементов.

E) Разработать функции работы со списком:

поиск информации по номеру, фамилии;

добавления элемента в список;

вывод на экран данных элемента списка;

вывод на экран фамилий, если задан(ы) (с клавиатуры): возраст, профессия, хобби;

сортировка списка по номеру, фамилии (дополнительно);

F) Интерфейс пользователя оформить в виде меню.

Код программы

Содержимое файла **main.cpp**:

```
#include "String.h"
#include "Dynamo.h"
#include "Human.h"

void backMessage();
void editMessage(Element<human> *current_data, human *input_data);
void inputInteger(const char *name, int *current_value, int *dest_value);
void inputString(const char *name, String *current_value, String *dest_value);
bool isNumber(char *str, int length);

int main()
{
	human elem_array[] = {
		{1, "Путин", "Владимир", 62, "Россия", "Президент", "Охота",
"79991234567"},
		{2, "Медведев", "Дмитрий", 49, "Россия", "Председатель
правительства", "Неизвестно", "79991234567"},
		{3, "Обама", "Барак", 53, "США", "Президент", "Неизвестно",
"19991234567"},
		{6, "Джонстон", "Дэвид", 73, "Канада", "Президент", "Неизвестно",
"19991234567"},
		{10, "Гаук", "Йоахим", 74, "Германия", "Президент", "Неизвестно",
"499991234567"},
		{4, "Олланд", "Франсуа", 60, "Франция", "Президент", "Неизвестно",
"339991234567"},
		{7, "Наполитано", "Джорджо", 89, "Италия", "Президент",
"Неизвестно", "399991234567"},
		{8, "Соммаруга", "Симонетта", 54, "Швейцария", "Президент",
"Неизвестно", "419991234567"},
		{9, "Нийнистё", "Саули", 66, "Финляндия", "Президент", "Неизвестно",
"3589991234567"},
		{5, "Елизавета II", "Александра", 88, "Британия", "Королева",
"Неизвестно", "449991234567"}
	};

	Dynamo<human> list;

	for (unsigned int i = 0; i < sizeof(elem_array) / sizeof(*elem_array); i+
+) {
		list.add(elem_array[i]);
	}

	char input[32];
	int input_menu;

	do {
		cout << "Выберите один из пунктов путём ввода цифры:" << endl;
		cout << "1 - Найти запись по номеру" << endl;
		cout << "2 - Найти запись по фамилии" << endl;
		cout << "3 - Удалить запись по номеру" << endl;
		cout << "4 - Удалить запись по фамилии" << endl;
		cout << "5 - Изменить запись по номеру" << endl;
		cout << "6 - Изменить запись по фамилии" << endl;
		cout << "7 - Добавить элемент в список" << endl;
		cout << "8 - Сортировать список по номеру" << endl;
		cout << "9 - Сортировать список по фамилии" << endl;
		cout << "10 - Вывести весь список" << endl;
		cout << "0 - Выход" << endl;
```

```
		cout << "Выбор: ";

		cin.getline(input, 32);
		input_menu = atoi(input);

		switch (input_menu) {
			case 1:
				{
					int input_number;

					cout << "Введи номер записи: ";
					cin.getline(input, 32);
					input_number = atoi(input);

					Element<human> *input_elem =
list.get(input_number);
					if (input_elem == NULL) {
						cout << "Ошибка: элемент под номером '" <<
input_number << "' не найден" << endl;
					} else {
						input_elem->show();
					}

					backMessage();
				}
				break;
			case 2:
				{
					cout << "Введи фамилию записи: ";
					cin.getline(input, 32);
					OemToChar(input, input);

					Element<human> *input_elem = list.get(input);
					if (input_elem == NULL) {
						cout << "Ошибка: элемент с фамилией '" <<
input << "' не найден" << endl;
					} else {
						input_elem->show();
					}

					backMessage();
				}
				break;
			case 3:
				{
					int input_number;

					cout << "Введи номер записи: ";
					cin.getline(input, 32);
					input_number = atoi(input);

					Element<human> *input_elem =
list.get(input_number);
					if (input_elem == NULL) {
						cout << "Ошибка: элемент под номером '" <<
input_number << "' не найден" << endl;
					} else {
						list.remove(input_elem);
						cout << "Элемент удалён" << endl;
					}

					backMessage();
				}
```

```
                        break;
                  case 4:
                        {
                              cout << "Введи фамилию записи: ";
                              cin.getline(input, 32);
                              OemToChar(input, input);

                              Element<human> *input_elem = list.get(input);
                              if (input_elem == NULL) {
                                    cout << "Ошибка: элемент с фамилией '" <<
input << "' не найден" << endl;
                              } else {
                                    list.remove(input_elem);
                                    cout << "Элемент удалён" << endl;
                              }

                              backMessage();
                        }
                        break;
                  case 5:
                        {
                              int input_number;

                              cout << "Введи номер записи: ";
                              cin.getline(input, 32);
                              input_number = atoi(input);

                              Element<human> *input_elem =
list.get(input_number);
                              if (input_elem == NULL) {
                                    cout << "Ошибка: элемент под номером '" <<
input_number << "' не найден" << endl;
                              } else {
                                    human edit_elem;
                                    editMessage(input_elem, &edit_elem);

                                    list.edit(input_elem, edit_elem);

                                    cout << "Элемент отредактирован" << endl;
                              }

                              backMessage();
                        }
                        break;
                  case 6:
                        {
                              cout << "Введи фамилию записи: ";
                              cin.getline(input, 32);
                              OemToChar(input, input);

                              Element<human> *input_elem = list.get(input);
                              if (input_elem == NULL) {
                                    cout << "Ошибка: элемент с фамилией '" <<
input << "' не найден" << endl;
                              } else {
                                    human edit_elem;
                                    editMessage(input_elem, &edit_elem);

                                    list.edit(input_elem, edit_elem);

                                    cout << "Элемент отредактирован" << endl;
                              }

                              backMessage();
```

```
                        }
                        break;
                  case 7:
                        {
                              human input_elem;

                              editMessage(NULL, &input_elem);

                              list.add(input_elem);

                              cout << "Запись успешно добавлена" << endl;
                              backMessage();
                        }
                        break;
                  case 8:
                        list.sort();

                        cout << "Список отсортирован по номеру" << endl;
                        backMessage();
                        break;
                  case 9:
                        list.sortByName();

                        cout << "Список отсортирован по фамилии" << endl;
                        backMessage();
                        break;
                  case 10:
                        list.show();
                        backMessage();
                        break;
            }
      } while (input_menu != 0);
      return 0;
}

void backMessage()
{
      cout << endl << "Для возврата в меню нажмите ввод" << endl;
      char input[32];
      cin.getline(input, 32);
}

void editMessage(Element<human> *current_data, human *input_data)
{
      if (current_data == NULL) {
            inputInteger("номер", NULL, &input_data->number);
            inputString("фамилию", NULL, &input_data->last_name);
            inputString("имя", NULL, &input_data->first_name);
            inputInteger("возраст", NULL, &input_data->age);
            inputString("страну", NULL, &input_data->country);
            inputString("работу", NULL, &input_data->job);
            inputString("хобби", NULL, &input_data->hobby);
            inputString("телефон", NULL, &input_data->phone_number);
      } else {
            inputInteger("номер", &current_data->getData()->number, &input_data-
>number);
            inputString("фамилию", &current_data->getData()->last_name,
&input_data->last_name);
            inputString("имя", &current_data->getData()->first_name,
&input_data->first_name);
            inputInteger("возраст", &current_data->getData()->age, &input_data-
>age);
            inputString("страну", &current_data->getData()->country,
&input_data->country);
```

```
            inputString("работу", &current_data->getData()->job, &input_data-
>job);
            inputString("хобби", &current_data->getData()->hobby, &input_data-
>hobby);
            inputString("телефон", &current_data->getData()->phone_number,
&input_data->phone_number);
      }
}

void inputInteger(const char *name, int *current_value, int *dest_value)
{
      char input[32];

      do {
            if (current_value == NULL) {
                  cout << "Введите " << name << ": ";
            } else {
                  cout << "Введите " << name << " [" << *current_value << "]: ";
            }
            cin.getline(input, 32);

            if (strlen(input) == 0 && current_value != NULL) {
                  *dest_value = *current_value;
                  return;
            }
      } while (!isNumber(input, 32));

      *dest_value = atoi(input);
}

void inputString(const char *name, String *current_value, String *dest_value)
{
      char input[32];

      if (current_value == NULL) {
            do {
                  cout << "Введите " << name << ": ";
                  cin.getline(input, 32);
            OemToChar(input, input);

                  dest_value->set(input);
            } while (strlen(input) == 0);
      } else {
            cout << "Введите " << name << " [" << *current_value << "]: ";
            cin.getline(input, 32);
        OemToChar(input, input);

            if (strlen(input) == 0) {
                  *dest_value = *current_value;
            } else {
                  dest_value->set(input);
            }
      }
}

bool isNumber(char *str, int length)
{
      return isdigit(str[0]);
}
```

Содержимое файла **String.h**:

```
#include <iostream>
#include <stdio.h>
```

```
#define strlen __strlen
#include <windows.h>
#undef strlen

#ifndef STRING_H
#define STRING_H

using namespace std;

class String {
public:
    String();
    String(const char *input_string);
    String(const String &obj);
    ~String();

    int length();
    char at(int index);
    void set(const char *str);
    void setchar(int index, char symbol);
    void append(const char *str);
    void erase(int start_pos, int end_pos);
    void insert(const char *str, int index);
    char *c_str();
    friend int strlen(const char *str);

    String& operator+(String& str);
    bool operator==(String& str);
    bool operator!=(String& str);
    bool operator<(String& str);
    bool operator>(String& str);

    friend istream& operator>>(istream& in, String& str);
    friend ostream& operator<<(ostream& out, String& str);

private:
    char *s;
    int n;
};

ostream& operator<<(ostream& out, const char *obj);

#endif // STRING_H
```

Содержимое файла **String.cpp**:

```
#include "String.h"

String::String()
{
    s = new char [256];
    s[0] = '\0';
    n = 0;
}

String::String(const char *input_string)
{
    s = new char [256];
    s[0] = '\0';
    n = 0;
    append(input_string);
}
```

```
String::String(const String &obj)
{
    s = new char [256];
    s[0] = '\0';
    n = 0;
    append(obj.s);
}

String::~String()
{
    if (s) {
        delete s;
        s = NULL;
    }
}

int String::length()
{
    return n;
}

char String::at(int index)
{
    return s[index];
}

void String::setchar(int index, char symbol)
{
    s[index] = symbol;
}

void String::append(const char *str)
{
    // получаем длину str
    int str_len = strlen(str);

    // создаём новую строку с бОльшим размером
    char *buf = new char [n + str_len + 1];

    // копируем
    for (int i = 0; i < n; i++) {
        buf[i] = s[i];
    }

    for (int i = n; i < n + str_len; i++) {
        buf[i] = str[i - n];
    }

    n += str_len;
    buf[n] = '\0';

    // удаляем старый объект и присваиваем указателю адрес buf
    delete s;
    s = buf;
}

void String::set(const char *str)
{
    int str_len = strlen(str);
    char *buf = new char [str_len + 1];

    // копируем
    for (int i = 0; i < str_len; i++) {
        buf[i] = str[i];
```

```
    }

    n = str_len;
    buf[n] = '\0';

    // удаляем старый объект и присваиваем указателю адрес buf
    delete s;
    s = buf;
}

void String::erase(int start_pos, int end_pos)
{
    end_pos++;

    for (int i = 0; s[i] != '\0'; i++) {
        s[start_pos + i] = s[end_pos + i];
    }

    n -= end_pos - start_pos;
    s[n] = '\0';
}

void String::insert(const char *str, int index)
{
    // получаем длину str
    int str_len = strlen(str);

    // создаём новую строку с бОльшим размером
    char *buf = new char [n + str_len + 1];

    // копируем
    for (int i = 0; i < index; i++) {
        buf[i] = s[i];
    }

    for (int i = 0; i < str_len; i++) {
        buf[i + index] = str[i];
    }

    for (int i = index; i < n; i++) {
        buf[i + str_len] = s[i];
    }

    n += str_len;
    buf[n] = '\0';

    // удаляем старый объект и присваиваем указателю адрес buf
    delete s;
    s = buf;
}

char *String::c_str()
{
    return s;
}

int strlen(const char *str)
{
    int len = 0;
    while (str[len] != '\0') {
        len++;
    }
    return len;
}
```

```
String& String::operator+(String& str)
{
    String *buf = new String;
    buf->append(s);
    buf->append(str.s);
    return *buf;
}

bool String::operator==(String& str)
{
    if (n != str.n) {
        return false;
    }

    for (int i = 0; i < n; i++) {
        if (s[i] != str.s[i]) {
            return false;
        }
    }
    return true;
}

bool String::operator!=(String& str)
{
    return !(*this == str);
}

bool String::operator<(String& str)
{
    int min_length = n < str.n ? n : str.n;

    for (int i = 0; i < min_length; i++) {
        if (s[i] == str.s[i]) {
            continue;
        }
        return s[i] < str.s[i];
    }
    return n < min_length;
}

bool String::operator>(String& str)
{
    int min_length = n < str.n ? n : str.n;

    for (int i = 0; i < min_length; i++) {
        if (s[i] == str.s[i]) {
            continue;
        }
        return s[i] > str.s[i];
    }
    return n < min_length;
}

istream& operator>>(istream& in, String& str)
{
    in >> str.s;
    OemToChar(str.s, str.s);
    str.n = strlen(str.s);
    return in;
}

ostream& operator<<(ostream& out, String& str)
{
```

```
    out << str.s;
    return out;
}

ostream& operator<<(ostream& out, const char *obj)
{
    int len = strlen(obj);
    char *str = new char [len + 1];

    CharToOem(obj, str);

    for (int i = 0; i < len; i++) {
        out << str[i];
    }
    return out;
}
```

Содержимое файла **Dynamo.h**:

```
#include <iostream>
#include "String.h"

using namespace std;

template <class T>
class Element {
public:
      void show();

      T *getData();
      void setData(T *input_data);

      Element *getNext();
      void setNext(Element *input_next);

private:
      T *data;
      Element *next;
};

template <class T>
class Dynamo {
public:
      Dynamo();
      ~Dynamo();

      void add(T data);
      bool remove(Element<T> *elem);
      void edit(Element<T> *elem, T data);
      void show();

      Element<T> *get(const char *str);
      Element<T> *get(const int value);

      void sort();
      void sortByName();

private:
      Element<T> *head;
};

template <class T>
Dynamo<T>::Dynamo()
{
```

```
	head = NULL;
}

template <class T>
Dynamo<T>::~Dynamo()
{
	while (head != NULL) {
		Element<T> *temp = head->getNext();
		delete head;
		head = temp;
	}
}

template <class T>
void Dynamo<T>::add(T data)
{
	Element<T> *temp = new Element<T>;
	temp->setData(new T(data));
	temp->setNext(head);
	head = temp;
}

template <class T>
bool Dynamo<T>::remove(Element<T> *elem)
{
	Element<T> *temp = head;

	if (temp == elem) {
		head = elem->getNext();
		return true;
	}

	while (temp != NULL) {
		if (temp->getNext() == elem) {
			temp->setNext(elem->getNext());
			delete elem;
			return true;
		}
		temp = temp->getNext();
	}
	return false;
}

template <class T>
void Dynamo<T>::edit(Element<T> *elem, T data)
{
	Element<T> *temp = new Element<T>;
	temp->setData(new T(data));
	temp->setNext(elem->getNext());
	*elem = *temp;
}

template <class T>
void Dynamo<T>::show()
{
	Element<T> *temp = head;

	while (temp != NULL) {
		cout << *temp->getData() << endl;
		temp = temp->getNext();
	}
}

template <class T>
```

```
Element<T> *Dynamo<T>::get(const int value)
{
	Element<T> *temp = head;

	while (temp != NULL) {
		if (temp->data == value) {
			return temp;
		}
		temp = temp->next;
	}
	return NULL;
}

template <class T>
Element<T> *Dynamo<T>::get(const char *str)
{
	Element<T> *temp = head;

	while (temp != NULL) {
		if (temp->data == str) {
			return temp;
		}
		temp = temp->next;
	}
	return NULL;
}

template <class T>
void Dynamo<T>::sort()
{
	if (head) {
		Element<T> *res = head;
		head = head->next;
		res->next = NULL;

		while (head) {
			Element<T> *elem = head;
			head = head->next;

			if (*elem->data < *res->data) {
				elem->next = res;
				res = elem;
			} else {
				Element<T> *p = res;

				while (p->next) {
					if (*p->next->data > *elem->data) {
						break;
					}

					p = p->next;
				}

				elem->next = p->next;
				p->next = elem;
			}
		}
		head = res;
	}
}

template <class T>
T *Element<T>::getData()
{
```

```
    return data;
}

template <class T>
void Element<T>::setData(T *input_data)
{
    data = input_data;
}

template <class T>
Element<T> *Element<T>::getNext()
{
    return next;
}

template <class T>
void Element<T>::setNext(Element *input_next)
{
    next = input_next;
}
```

Содержимое файла **Human.h**:

```
#include <iostream>

using namespace std;

struct human {
      int number;
      String last_name;
      String first_name;
      int age;
      String country;
      String job;
      String hobby;
      String phone_number;
};

template <>
Element<human> *Dynamo<human>::get(const int value)
{
      Element<human> *temp = head;

      while (temp != NULL) {
            if (temp->getData()->number == value) {
                  return temp;
            }
            temp = temp->getNext();
      }
      return NULL;
}

template <>
Element<human> *Dynamo<human>::get(const char *str)
{
      Element<human> *temp = head;
      String buf_str(str);

      while (temp != NULL) {
            if (temp->getData()->last_name == buf_str) {
                  return temp;
            }
            temp = temp->getNext();
      }
      return NULL;
```

```
}

template <>
void Element<human>::show()
{
	if (this == NULL) {
		return;
	}
	cout << "Номер: " << data->number << endl
		<< "Фамилия: " << data->last_name << endl
		<< "Имя: " << data->first_name << endl
		<< "Возраст: " << data->age << endl
		<< "Страна: " << data->country << endl
		<< "Работа: " << data->job << endl
		<< "Хобби: " << data->hobby << endl
		<< "Телефон: " << data->phone_number << endl;
}

template <>
void Dynamo<human>::show()
{
	Element<human> *temp = head;

	while (temp != NULL) {
		cout << temp->getData()->number << " "
			<< temp->getData()->last_name << " "
			<< temp->getData()->first_name << " "
			<< temp->getData()->age << " "
			<< temp->getData()->country << " "
			<< temp->getData()->job << " "
			<< temp->getData()->hobby << " "
			<< temp->getData()->phone_number << endl;

		temp = temp->getNext();
	}
}

template <>
void Dynamo<human>::sort()
{
	if (head) {
		Element<human> *res = head;
		head = head->getNext();
		res->setNext(NULL);

		while (head) {
			Element<human> *elem = head;
			head = head->getNext();

			if (elem->getData()->number < res->getData()->number) {
				elem->setNext(res);
				res = elem;
			} else {
				Element<human> *p = res;

				while (p->getNext()) {
					if (p->getNext()->getData()->number > elem-
>getData()->number) {
						break;
					}

					p = p->getNext();
				}
```

```
                        elem->setNext(p->getNext());
                        p->setNext(elem);
                  }
            }
            head = res;
      }
}

template <>
void Dynamo<human>::sortByName()
{
      if (head) {
            Element<human> *res = head;
            head = head->getNext();
            res->setNext(NULL);

            while (head) {
                  Element<human> *elem = head;
                  head = head->getNext();

                  if (elem->getData()->last_name < res->getData()->last_name) {
                elem->setNext(res);
                        res = elem;
                  } else {
                        Element<human> *p = res;

                        while (p->getNext()) {
                              if (p->getNext()->getData()->last_name > elem-
>getData()->last_name) {
                                    break;
                              }

                              p = p->getNext();
                        }

                        elem->setNext(p->getNext());
                        p->setNext(elem);
                  }
            }
            head = res;
      }
}
```

Результат работы программы

```
Выберите один из пунктов путём ввода цифры:
1 - Найти запись по номеру
2 - Найти запись по фамилии
3 - Удалить запись по номеру
4 - Удалить запись по фамилии
5 - Изменить запись по номеру
6 - Изменить запись по фамилии
7 - Добавить элемент в список
8 - Сортировать список по номеру
9 - Сортировать список по фамилии
10 - Вывести весь список
0 - Выход
Выбор: 4
Введи фамилию записи: Путин
Элемент удалён

Для возврата в меню нажмите ввод
```

Вывод

Разработанный проект программы составлен из пяти файлов: основной файл — main.cpp, класс строки — String.h и String.cpp, класс списка — Dynamo.h и файл с методами и структурой human — Human.h.

В классе списка есть одно поле — Element<T> *head, являющееся заголовком списка. Тип Element — является шаблоном класса, в котором находятся данные и указатель на следующий элемент списка. Также там есть метод show(), нужен для обеспечения вывода данных.

Разработанный класс списка позволяет не только создавать списки, но и добавлять, изменять, удалять, сортировать и выводить элементы из списка. Также данный класс позволяет создавать списки с любым типом элемента.

Код, специфичный для данной задачи, был вынесен в отдельный файл Human.h, в котором объявляется структура и методы, которые будут вызваны для списка с типом human.

В итоге получилась программа, позволяющая полноценно работать с односвязными линейными списками.