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 wchar_t *name, int *current_value, int *dest_value);
void inputString(const wchar_t *name, String *current_value, String
*dest_value);
bool isNumber(wchar_t *str, int length);
int wtoi(const wchar_t *str);

int main()
{
#ifdef _WIN32
	setlocale(LC_ALL, "Russian");
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
#else // _WIN32
	locale::global(locale("ru_RU.UTF-8"));
#endif // _WIN32

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

	Dynamo<human> list;

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

	wchar_t input[32];
	int input_menu;

	do {
		wcout << L"Выберите один из пунктов путём ввода цифры:" << endl;
		wcout << L"1 - Найти запись по номеру" << endl;
```

```
		wcout << L"2 - Найти запись по фамилии" << endl;
		wcout << L"3 - Удалить запись по номеру" << endl;
		wcout << L"4 - Удалить запись по фамилии" << endl;
		wcout << L"5 - Изменить запись по номеру" << endl;
		wcout << L"6 - Изменить запись по фамилии" << endl;
		wcout << L"7 - Добавить элемент в список" << endl;
		wcout << L"8 - Сортировать список по номеру" << endl;
		wcout << L"9 - Сортировать список по фамилии" << endl;
		wcout << L"10 - Вывести весь список" << endl;
		wcout << L"0 - Выход" << endl;

		wcout << L"Выбор: ";

		wcin.getline(input, 32);
		input_menu = wtoi(input);

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

					wcout << L"Введи номер записи: ";
					wcin.getline(input, 32);
					input_number = wtoi(input);

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

					backMessage();
				}
				break;
			case 2:
				{
					wchar_t *input_string = new wchar_t [32];

					wcout << L"Введи фамилию записи: ";
					wcin.getline(input_string, 32);

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

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

					wcout << L"Введи номер записи: ";
					wcin.getline(input, 32);
					input_number = wtoi(input);
```

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

                              backMessage();
                        }
                        break;
                  case 4:
                        {
                              wcout << L"Введи фамилию записи: ";
                              wcin.getline(input, 32);

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

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

                              wcout << L"Введи номер записи: ";
                              wcin.getline(input, 32);
                              input_number = wtoi(input);

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

                                    list.edit(input_elem, edit_elem);

                                    wcout << L"Элемент отредактирован" << endl;
                              }

                              backMessage();
                        }
                        break;
                  case 6:
                        {
                              wcout << L"Введи фамилию записи: ";
                              wcin.getline(input, 32);

                              Element<human> *input_elem = list.get(input);
                              if (input_elem == NULL) {
                                    wcout << L"Ошибка: элемент с фамилией '" <<
input << L"' не найден" << endl;
```

```
                        } else {
                              human edit_elem;
                              editMessage(input_elem, &edit_elem);

                              list.edit(input_elem, edit_elem);

                              wcout << L"Элемент отредактирован" << endl;
                        }

                        backMessage();
                  }
                  break;
            case 7:
                  {
                        human input_elem;

                        editMessage(NULL, &input_elem);

                        list.add(input_elem);

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

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

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

void backMessage()
{
      wcout << endl << L"Для возврата в меню нажмите ввод" << endl;
      wchar_t input[32];
      wcin.getline(input, 32);
}

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

```
            inputInteger(L"номер", &current_data->data->number, &input_data-
>number);
            inputString(L"фамилию", &current_data->data->last_name, &input_data-
>last_name);
            inputString(L"имя", &current_data->data->first_name, &input_data-
>first_name);
            inputInteger(L"возраст", &current_data->data->age, &input_data-
>age);
            inputString(L"страну", &current_data->data->country, &input_data-
>country);
            inputString(L"работу", &current_data->data->job, &input_data->job);
            inputString(L"хобби", &current_data->data->hobby, &input_data-
>hobby);
            inputString(L"телефон", &current_data->data->phone_number,
&input_data->phone_number);
      }
}

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

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

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

      *dest_value = wtoi(input);
}

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

      if (current_value == NULL) {
            do {
                  wcout << L"Введите " << name << ": ";
                  wcin.getline(input, 32);

                  *dest_value = input;
            } while (strlen(input) == 0);
      } else {
            wcout << L"Введите " << name << " [" << *current_value << "]: ";
            wcin.getline(input, 32);

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

bool isNumber(wchar_t *str, int length)
```

```
{
	return isdigit(str[0]);
}

int wtoi(const wchar_t *str)
{
	return (int)wcstol(str, 0, 10);
}
```

Содержимое файла **String.h**:

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

#ifndef STRING_H
#define STRING_H

#ifdef __WIN32

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

#endif // __WIN32

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

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

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

	friend std::wistream& operator>>(std::wistream& in, String& str);
	friend std::wostream& operator<<(std::wostream& out, String& str);

private:
	wchar_t *s;
	int n;
};

#endif
```

Содержимое файла **String.cpp**:

```
#include "String.h"

using namespace std;

String::String()
{
```

```
	s = new wchar_t [256];
	s[0] = '\0';
	n = 0;
}

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

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

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

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

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

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

void String::append(const wchar_t *str)
{
	// получаем длину str
	int str_len = 0;
	while (str[str_len] != '\0') {
		str_len++;
	}

	// создаём новую строку с бОльшим размером
	wchar_t *buf = new wchar_t [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::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 wchar_t *str, int index)
{
	// получаем длину str
	int str_len = strlen(str);

	// создаём новую строку с бОльшим размером
	wchar_t *buf = new wchar_t [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;
}

int strlen(const wchar_t *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;
```

```
}

String& String::operator=(const wchar_t *str)
{
      String *buf = new String(str);
      s = buf->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;
}

std::wistream& operator>>(std::wistream& in, String& str)
{
      in >> str.s;
      str.n = strlen(str.s);
      return in;
}

std::wostream& operator<<(std::wostream& out, String& str)
{
      out << str.s;
```

```
	return out;
}
```

Содержимое файла **Dynamo.h**:

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

using namespace std;

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

	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 wchar_t *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->next;
		delete head;
		head = temp;
	}
}

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

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

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

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

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

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

	while (temp != NULL) {
		wcout << *temp->data << endl;
		temp = temp->next;
	}
}

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 wchar_t *str)
{
	Element<T> *temp = head;
	String buf_str(str);

	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;
      }
}
```

Содержимое файла **Dynamo.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->data->number == value) {
```

```
                return temp;
            }
            temp = temp->next;
      }
      return NULL;
}

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

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

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

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

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

            temp = temp->next;
      }
}

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

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

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

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

                              p = p->next;
                        }

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

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

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

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

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

                              p = p->next;
                        }

                        elem->next = p->next;
                        p->next = 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.