Язык программирования C++

Лекция 10

Дополнительные вопросы:

- класс string
- стандартная библиотека шаблонов STL
  - контейнеры

Три основные концепции объектно-ориентированного подхода

- инкапсуляция
- наследование
- полиморфизм

Класс string

Заголовочный файл <string>

Объект класса string представляет собой последовательность символов. Он проще в использовании, чем символьный массив char[ ], и предлагает более естественное представление строки как типа данных.

```
string str1;                    // создание пустого объекта строки
string str2 = “John Smith”;     // создание инициализированного объекта строки
cin >> str1;                    // str1 увеличен для того, чтобы вместить ввод
string str3;
str3 = str1 + str2;             // присвоить str3 объединение строк str1 и str2
str1 += str2;                   // добавить str2 в конец str1
```

Класс string реализует множество удобных функций для работы со строками: удаление части или всей строки, замена части или всей строки частью другой строки, поиск, добавление и удаление данных из строки, сравнение частей строк и строк целиком, извлечение подстроки из строки и др.

Дополнительная информация - в online справочнике
www.cplusplus.com/reference/string/string

Стандартная библиотека шаблонов STL

(Standard Template Library)

Библиотека STL содержит набор шаблонов, представляющих контейнеры, итераторы, алгоритмы и объекты функций.

Контейнер - это структура данных, похожая на массив, которая может хранить несколько значений. Контейнеры STL однородны по структуре и хранят только однотипные данные.

Итераторы - это объекты, позволяющие перемещаться внутри контейнера подобно тому, как указатели позволяют перемещаться по массиву, они являются обобщением указателей.

Алгоритмы используются для обработки данных, например, сортировки контейнера или поиска конкретного значения в списке.

Объекты функций - это объекты, которые ведут себя подобно функциям, они могут быть объектами класса или указателями на функцию.

Библиотека STL позволяет создавать различные контейнеры, включая массивы, очереди и списки, и осуществлять множество операций, например, поиск, сортировку и перетасовку в случайном порядке.

Контейнеры STL

vector

<vector> - контейнер, в котором хранится последовательность однотипных объектов в виде массива смежных элементов

```
#include <vector>
using namespace std;
vector<int> vect1(5);          // вектор из 5 элементов типа int
int n;
cin >> n;
vector<double> vect2(n);       // вектор из n элементов типа double
vect1[0] = 3;                  // доступ к элементу вектора с помощью операции [ ]
```

Контейнеры STL

list

<list> - контейнер, в котором хранится последовательность длиной *N* в виде двусвязного списка из *N* узлов, в каждом из которых хранится единственный элемент вместе с адресами предыдущего и последующего элементов.
Основное преимущество двусвязного списка - его гибкость. Можно свободно вставлять и удалять элементы внутри списка, просто переписывая прямые и обратные ссылки в узлах.
Но в отличие от контейнера vector, list не поддерживает форму записи массива и произвольный доступ.

```
#include <list>
using namespace std;
list<int> one(5, 2);                          // список из 5 двоек
int stuff[5] = {1,2,4,8,6};
list<int> two;
two.insert(two.begin(), stuff, stuff+5);      // в список two вставляем элементы из
                                              // массива stuff
```

Контейнеры STL

map

<map> - ассоциативный контейнер (отображение, словарь), в котором хранится последовательность длиной *N* в виде упорядоченного двоичного дерева из *N* узлов, в каждом из которых хранится единственный элемент типа pair<const Key, T>. Последовательность упорядочена по компоненту const Key.

Разновидность <map> - <multimap> (мультиотображение, словарь с дубликатами) допускает, чтобы два (смежных) элемента имели ключи const Key с одинаковым упорядочиванием, в то время как словарь - нет. Т.е. map или multimap вместе с каждым ключом хранят “отображаемое (словарное) значение” (“mapped value”). Все ключи словаря (map) уникальны, а ключи словаря с дублированием (multimap) могут дублироваться.

Итераторы

Итераторы являются обобщением указателей: это объекты, которые указывают на другие объекты. Они имеют большое значение для таких обобщенных алгоритмов, как find(), поскольку могут использоваться для прохода по диапазону объектов. Если итератор указывает на какой-либо объект диапазона, то после инкремента он будет указывать на следующий объект этого диапазона.

```
#include <string>
#include <map>
typedef std::pair<const int, std::string> Pair;       // объявление новых типов данных
typedef std::multimap<int, std::string> MapCode;
MapCode codes;                                        // объявление словаря с дублированием
codes.insert(Pair(510, “Oakland”));                   // заполнение словаря данными
codes.insert(Pair(718, “Brooklyn”));
codes.insert(Pair(718, “Staten Island”));
codes.insert(Pair(415, “San Francisco”));
MapCode::iterator it;                                 // объявление итератора
for (it=codes.begin(); it!=codes.end(); ++it)         // вывод упорядоченных данных
    cout << (*it).first << “  “ << (*it).second << end;
```

Контейнеры STL

Member functions

(constructor)	Construct vector (public member function)
(destructor)	Vector destructor (public member function)
operator=	Assign content (public member function)

Iterators:

begin	Return iterator to beginning (public member function)
end	Return iterator to end (public member function)
rbegin	Return reverse iterator to reverse beginning (public member function)
rend	Return reverse iterator to reverse end (public member function)
cbegin	Return const_iterator to beginning (public member function)
cend	Return const_iterator to end (public member function)
crbegin	Return const_reverse_iterator to reverse beginning (public member function)
crend	Return const_reverse_iterator to reverse end (public member function)

Capacity:

size	Return size (public member function)
max_size	Return maximum size (public member function)
resize	Change size (public member function)
capacity	Return size of allocated storage capacity (public member function)
empty	Test whether vector is empty (public member function)
reserve	Request a change in capacity (public member function)
shrink_to_fit	Shrink to fit (public member function)

Element access:

operator[]	Access element (public member function)
at	Access element (public member function)
front	Access first element (public member function)
back	Access last element (public member function)
data	Access data (public member function)

Modifiers:

assign	Assign vector content (public member function)
push_back	Add element at the end (public member function)
pop_back	Delete last element (public member function)
insert	Insert elements (public member function)
erase	Erase elements (public member function)
swap	Swap content (public member function)
clear	Clear content (public member function)
emplace	Construct and insert element (public member function)
emplace_back	Construct and insert element at the end (public member function)

пример из online справочника cplusplus.com со списком функций для контейнера std::vector

1. Инкапсуляция

см. также лекцию 6

Инкапсуляция – это механизм, который объединяет данные и методы (т.е. алгоритмы), манипулирующие этими данными, а также защищает и то, и другое от внешнего вмешательства или неправильного использования. Когда данные и работающие с ними методы объединяются таким образом, то создается объект.

Внутри объекта методы и данные могут быть закрытыми (private). Закрытые методы и данные доступны только для других частей этого объекта. Т.е. закрытые методы и данные недоступны для тех частей программы, которые существуют вне объекта. Если методы и данные являются открытыми (public), то, несмотря на то, что они заданы внутри объекта, они доступны и для других частей программы.

Общепринятым является стиль программирования, когда открытая часть объекта используется для того, чтобы обеспечить контролируемый интерфейс к закрытым элементам объекта.

2. Наследование

см. также лекцию 9

Наследование – это процесс, посредством которого один объект может приобретать свойства другого. Точнее, объект может наследовать основные свойства другого объекта и добавлять к ним черты, характерные только для него. Дочерний класс наследует все связанные с родителем качества и добавляет к ним свои собственные характеристики.

Без использования механизма наследования для каждого объекта пришлось бы задать все характеристики, которые бы исчерпывающе его определяли. Однако при использовании наследования можно описать объект путем определения того общего класса (или классов), к которому он относится, с теми специальными чертами, которые делают этот объект уникальным.

3. Полиморфизм

см. также лекцию 4

Полиморфизм – это свойство, которое позволяет одно и то же имя использовать для решения двух или более схожих, но технически разных задач. Применительно к объектно-ориентированному программированию это позволяет использовать одно имя для задания общих для класса действий. Выполнение каждого конкретного действия будет определяться типом данных.

Например, в языке С нахождение абсолютной величины числа требует трех различных функций: abs(), labs() и fabs(). Они возвращают абсолютную величину целых, длинных целых и чисел с плавающей запятой соответственно.

В С++ каждая из этих функций может быть названа abs(). Тип данных, который используется при вызове функции, определяет какая конкретная версия функции действительно выполняется. Этот тип полиморфизма называется перегрузкой функций (function overloading)

3. Полиморфизм (2)

см. также лекцию 7

Полиморфизм может применяться также и к операторам. Фактически во всех языках программирования ограниченно применяется полиморфизм, например, в арифметических операторах. Так, в языке С символ + используется сложения целых, длинных целых, символьных переменных и чисел с плавающей запятой. В этом случае компилятор автоматически определяет, какой тип арифметики требуется.

В С++ можно применить эту концепцию подмены операторов и к другим, заданным самим программистом, типам данных. Такой тип полиморфизма называется перегрузкой операторов (operator overloading).

Практическое задание

Маша и Петя приглашают своих друзей в гости. Надо написать программу, которая делает следующее:

1. позволяет Маше ввести список имен ее друзей. Имена сохраняются в контейнере, а затем выводятся на экран в отсортированном (алфавитном) порядке

2. позволяет Пете ввести список имен его друзей. Имена сохраняются во втором контейнере, а затем выводятся на экран в отсортированном порядке

3. создает третий контейнер, который объединяет эти два списка, исключает дубликаты и выводит содержимое этого контейнера в отсортированном порядке