РАБОТА НА ЭВМ И ПРОГРАММИРОВАНИЕ

с.н.с. В. В. Борисенко

2 курс, отделение математики

Осенний семестр, 48 часов.

1. Обзор алгоритмических языков: традиционные языки — Си, Фортран, объектно ориентированные — Java, C#. Основные различия между традиционными и объектно ориентированными языками. Размещение переменных и объектов в памяти в случае традиционных языков. Три вида памяти — статическая, стековая и динамическая. Организация динамической памяти и способы размещения объектов в объектно ориентированных языках. Промежуточное положения языка C++ между традиционными и объектно ориентированными: по способу размешения объектов в памяти — традиционный язык, однако поддерживает ряд понятий, свойственных объектно ориентированным языкам.

2. Знакомство с языком C++. Основные понятия — класс, члены класса, методы класса. Типы переменных: обычные переменные, указатели и ссылки, модификатор “const”. Важность правильного задания прототипов методов класса и функций. Переопределение операторов для классов. Класс `R2Vector` (вектор на плоскости) как пример класса. Десять способов неправильного описания прототипов методов сложения (`operator+`) и “увеличить на” (`operator+=`) и единственный правильный способ. Реализация классов “вектор”, “точка” и “прямоугольник”, используемых в двумерной графике. Два этапа реализации класса: задание интерфейса и реализация методов. Реализация других классов, представляющих математические объекты: многочлен от одной переменной, ортогональное преобразование двух- и трехмерного пространства и т.п.

3. Исключения “exception” и их использование в современных языках. Конструкция “`try — catch`”. Использование исключений для обработки ошибочных ситуаций. Преимущества использования исключений вместо анализа кодов возврата. Общее понятие исполнителя и класс в C++ как его конкретное воплощение. Язык “Псевдокод” как неформальный язык для описания алгоритмов.

4. Структуры данных: общие понятия. Стек и реализация стека на псевдокоде и на C++. Обратная польская запись формул и ее использование (байткод Java, язык Postscript и т.п.). Проект “Стековый калькулятор”. Аппаратная реализация стека: процессор, оперативная память, регистры процессора, роль регистров `PC`, `SP` и `FP`. Размещение локальных переменных функции в аппаратном стеке. Механизм вызова функций и передачи параметров функции.

5. Статические члены и методы класса. Наследование и виртуальные методы. Различия в механизме вызова обычных и виртуальных методов. Зачем нужны виртуальные методы.

6. Программирования в оконных средах, общие принципы. Событийная модель программирования: события, сообщения о событиях, очередь сообщений, цикл обработки событий и обработчики событий. Преимущества объектно-ориентированного подхода при программировании в оконных средах. Использование виртуальных методов класса “Окно” для реализации обработчиков событий. Понятие API операционной системы и библиотека классов как надстройка над API. Конкретные библиотеки классов для работы в оконных средах: MFC в случае MS Windows, Qt в случае X-Windows (Unix).

7. Введение в программирование под X11. Понятие X-клиента и X-сервера, переменная `DISPLAY`. Программирование над Xlib (нижний уровень). Класс “Графическое окно” `GWindow` и его использование для создания графических программ. Статические методы и переменные класса `GWindow`. Типы событий в X11, виртуальные методы класса `GWindow`, используемые для обработки событий. Создание окна и рисование в окне. Проект “`GWindow`”.

8. Структуры данных: последовательного доступа — стек, очередь, дек, двусвязный и односвязный списки; прямого доступа — массив (вектор), динамический массив, матрица, множество, нагруженное множество. Реализация стека и дека (очереди) на базе массива. Битовая реализация ограниченного множества целых чисел. Способы реализация циклов “для каждого элемента”: итераторы, методы типа “elements”.

9. Проект “Выпуклая оболочка”. Его реализация на языке C++ с использованием классов, поддерживающих двумерную графику (`R2Vector`, `R2Point` и т.п.). Графическая иллюстрация

работы с исполнителем "Выпуклая оболочка" с использованием класса "Графическое окно" (`GWindow`). Проект "`R2Convex`".

10. Схема построения цикла с помощью инварианта на примерах расширенного алгоритма Евклида, быстрого возведения в степень и вычисления логарифма без использования разложения в ряд.

11. Применение схемы инварианта цикла в алгоритме бинарного поиска. Реализация множества с использованием бинарного поиска на языке C++: проект "`BinSet`".

12. Оценка снизу минимального числа операций в произвольном алгоритме сортировки. Алгоритмы пирамидальной сортировки, сортировки слиянием и быстрой сортировки.

13. Ссылочные реализации. Реализации Л1 и Л2-списков на языке C и C++.

14. Идея реализации множества с помощью хеширования. Два способа реализации множества с помощью хеширования: внешнее разрешение коллизий с помощью вектора множеств, использование дополнительного массива ("кучи"). Ссылочная реализация вектора множеств на языке C++ и и хеш-реализация множества на его основе. Проект "`HashSet`".

15. Проект "Текстовый редактор" (`class TextEdit`), реализация на языке C++ с использованием Л2-списка строк и класса "Графическое окно" (`GWindow`).

16. Графы и деревья, их использование в программировании. Рекурсивный обход дерева. Бинарный поиск с использованием деревьев. Сбалансированное дерево. Реализация множества с помощью сбалансированного дерева. Проект "`TreeSet`".

17. Формальные грамматики (рассматриваются только контекстно свободные грамматики). Левые и правые выводы. Дерево вывода. Детерминированные языки. Лемма о разрастании для КС-языков.

18. Конечные автоматы. Детерминированные и недетерминированные КА. Построение детерминированного КА, эквивалентного заданному недетерминированному. Язык, задаваемый КА. Лемма о разрастании для автоматных языков. Регулярные выражения. Теорема Клини о совпадении классов автоматных и регулярных языков.

19. Инструментальные средства программирования. Утилита LEX и ее использование для создания лексических анализаторов (сканеров). Примеры написания сканеров с использованием JLex.

20. Задача определения принадлежности заданной цепочки языку, заданному формальной грамматикой, и восстановления вывода цепочки: разбор. Противоположность задач вывода и разбора. Методы разбора: нисходящий, или рекурсивный разбор, и восходящий, или LR-разбор. Смысл названий LL(1) разбор и LR(1) разбор. Преимущества восходящего разбора по сравнению с рекурсивным.

21. Восходящий разбор, или LR-разбор. Определение понятий ситуации грамматики и состояния разбора. Состояния LR(0)-разбора и LR(1)-разбора. Построение множества возможных состояний и переходов. Построение таблиц для LR(0) и LR(1) разбора. Алгоритм работы парсера.

22. Утилита YACC и ее использование для построения синтаксических анализаторов (парсеров). Этапы разработки компилятора, парсер как первый шаг в разработке компилятора. Преимущества использования YACC'а по сравнению с "ручным" программированием. Примеры построения парсеров с использованием YACC'а. Проект "Интерпретатор формул".