Контакты

Валединский Владимир Дмитриевич

сайт с материалами: v-dinsky.info

вопросы, комментарии и т.п. по почте

v-dinsky@yandex.ru

в теме письма пишите "лекции-1"
на это слово будет настроен фильтр,
чтобы письмо не затерялось в массе других

Тема 1. Обзор языка C

В этих файлах будет содержаться краткий (предварительный) конспект лекции.

Это позволит слушателям не тратить время на подробное конспектирование, однако реальная лекция может пойти и по другому сценарию, там может оказаться больше или меньше материала, могут быть рассмотрены какие-то дополнительные детали, на лекциях часто задаются вопросы, которые требуют отдельых разговоров и т.д. То есть, данные записи не дублируют лекции, а скорее задают общее направление обсуждений, которое дополняется и разъясняется при устном обсуждении.

Эти материалы иногда будут выкладываться до фактической лекции, иногда они потом корректируются по фактц прочитанного. Таким образом, для ознакомления с материалом стоит посмотреть накануне не выложен ли материал, просмотреть его, а потом прослушать именно лекцию, чтобы устранить возможные непонятные моменты.

По учебному плану курс программирования длится 2 года, и еще в течении двух лет присутствуют практикумы, связанные с применением ЭВМ для решения практических задач.

Целью первого года обучения является формирование базовых навыков программирования, изучения некоторых классических алоритмов обработки данных, осознание основных этапов решения задач с помощью вычислительной техники, в том числе основных принципов функционирования этой техники и этапов прохождения задачи через компьютер.

В начальной стадии курс ориентируется на минимальный уровень подготовки. Мы будем начинать с простых вещей и не будем лезть в дебри программистских технологий. Однако вопросы идеологии и технологии программирования будут постоянно обсуждаться. На лекция будет излагаться базовый материал, а семинары посвящены решению учебных задач на компьютере. Для получения зачета в этом семестре необходимо решить выданный набор задач с выполнением всех необходимых требований к алгоритму и оформлению кода программы, а также успешно выполнить контрольные и тестовые практические работы, предусмотренные программой обучения.

Мы пока рассматриваем императивный подход к программированию, когда программа представляет собой набор инструкций, которые должна выполнить вычислительная система, чтобы получить требуемый результат.

Парадигма программирования (см. Википедию)

Процедурное программирование:

исходные данные → процедуры → результат

объектно-ориентированное программирование:

объекты в исходном состоянии → взаимодействие объектов → объекты в требуемом состоянии

Пока что воплощаем процедурный подход к программированию.

Пусть у нас есть конкретная задача. Что значит “написать программу для решения этой задачи”?

Во-первых (в соответствии с парадигмой) мы должны определиться с тем, что считать исходными данными и что считать результатом.

Во-вторых, надо придумать “математический алгоритм решения”, который получает на вход исходные данные и на выходе производит результат при помоши выполнения “процедур, реализующизх алгоритм”.

В-третьих, нам надо записать (реализовать) этот алгоритм в виде программы на выбранном алгоритмическом языке.

В четвертых, надо “запустить” эту программу на компьюере и как-то проверить ее работоспособность.

Т.е. абстрактные шаги решения задачи (в процедурном программировании):

1. Выбор способов преставления исходный данных и результата в вашей программе.

2. Разработка математического алгоритма решения в виде набора процедур.

3. Реализация алгоритма в виде программы на алгоритмическом языке.

4. Отладка и тестирование программы.

Пункты 1–2, вообще говоря, не сильно зависят от конкретного языка программирования. Они оперируют общепринятыми понятиями (например, “целое число” или “массив” и т.д.) и часто выражаются в виде набора действий типа “присвоить значение, вычисленное по формуле”.

Поэтому при решении конкретной задачи всегда полезно сначала просто представить себе алгоритм вычислений без привязки к конкретному языку. И только потом, когда язык программирования будет лучше освоен, можно уже переходить к формированию алгоритма сразу в терминах и конструкциях языка.

Пункт 4 очень важен. Есть исследования, которые показывают, что отладка и тестирование программы занимает основное время при решении задачи (порядка 80% и даже более). К этому надо быть морально готовым и принимать на себя полную ответственность за правильную работу вашей программы.

Какой язык выбрать для обучения?

Обучение программированию предполагает использование конкретного алгоритмического языка для записи программ. В средней школе ранее был весьма популярен язык Pascal, также часто встречаются C, C++, Java, в последние годы становится популярен Python, а давным-давно для обучения использовался Basic.

Для ответа на этот вопрос надо определиться, чего хочется и что мы хотим получить в результате обучения.

Нет хорошего или плохого языка. Каждый язык предполагает определенную область применения и круг решаемых задач.

Мы выбрали C для начального обучения, поскольку этот язык наиболее близок к внутренней “кухне” вычислительного процесса, и, с другой стороны, достаточно универсален и эффективен для реализации многих задач.

Специфика задач, с которыми может столкнуться выпускник мехмата

— изощренные и математически сложные алгоритмы

— большие массивы данных самых разнообразных форматов
— сложные взаимосвязи между данными
— требование эффективности и надежности программ
— интеграция с существующими библиотеками программ

Мы будем далее использовать язык C (а потом и C++), ограничиваясь на данном этапе процедурными возможностями. Использование C++ в данном контексте оправдано тем, что, по сравнению с простым C, этот язык получил ряд дополнительных возможностей, которые упрощают запись алгоритмов в ряде случаев. Кроме того, чистый C, хоть и применяется кое-где в настоящее время, но обычно это связано с использованием специальной техники в условиях ограниченных вычислительных мощностей (например, микропроцессоры технических устройств), и подобное программирование является довольно специфичной областью.

Далее мы будем вести изложение в рамках некоторого "общего" подмножества языков C и C++, и давать отдельные комментарии к конструкциям, которые специфичны для каждого из этих языков в отдельности.

Как знакомиться с языком программирования.
— идеология и парадигма языка
— структура программы
— синтаксис
— интерфейсы и операционное окружение

В соответствии с концепциями императивного программирования и 4 пунктами по решению задачи, которые были показаны ранее, нам надо познакомиться со следующими возможностями языка:

1. базовые типы данных, простые и составные типы;
2. операции с базовыми типами;
3. операторы и управляющие конструкции;
4. структуризация записи программы (блоки, объявления, процедуры и т.п.);
5. видимость объектов (локальные, глобальные, статические и т.д.);
6. организация ввода-вывода, внешние и стандартные библиотеки;
7. прочие дополнительные возможности;
8. запуск и отладка программы в конкретной системе программирования.

Нужно позаботиться о справочном руководстве — купить книгу, найти сайты в интернете, обзавестись знающими друзьями.

В данном курсе лекций мы будем обсуждать только самые основные и принципиальные концепции простроения языка, оставляя технические детали техническим справочникам.

С практической точки зрения, обучаться можно несколькими способами. Например, сначала некоторое время потратить на "изучение теории", причем в достаточно подробно, и затем приступить к решению практических задач.

Другой подход — сразу пробовать решать задачи (от простых к сложным), опираясь на начальный "здравый смысл" и по мере продвижения осваивая новые техники.

Так как программрование как таковое есть в большой степени ремесло, то и обучаться ему полезно так же как и ремеслу, т.е. сразу приступая к решению задач и постепенно набираясь опыта и знаний под руководством опытных наставников.

Поэтому сразу возьмем простейшую (но не тривиальную) задачу — подсчитать среднее арифметическое некоторого набора вещественных чисел. И тут, до того как пытаться писать программу, придется определиться (принять решение) с множеством сопуствующих вопросов, общих для любой постановки задачи в области программирования.

Вернемся к четырем шагам решения задачи и посмотрим, какие проблемы нам предстоит решить.

1. Выбор способов преставления исходный данных и результата в вашей программе.

— что собой представляют исходные данные, откуда они берутся?

— как и откуда ваша программа получит эти данные?

— как эти даные будут представлены в вашей программе?

— что предполагается получить в качестве ответа?

— в какой форме этот ответ должен быть предоставлен пользвателю?

2. Разработка математического алгоритма решения в виде набора процедур.

— по каким формулам будет вычисляться результат?

— какие стандартные алгоритмические конструкции нам потребуются?

3. Реализация алгоритма в виде программы на алгоритмическом языке.

— тут просто записываем шаг 2 в терминах конкретного языка.

— как реагировать на возможные некорректные ситуации и что это могут быть за ситуации?

4. Отладка и тестирование программы.

— как проверить, что ваша программа работает правильно?

— какие тесты должна проходить программа?

Ответы на эти вопросы зависят от разных факторов. Например, они могут быть сформулированы заказчиком или вы должны принять соответствующие решения самостоятельно. В нашем случае мы примем следующие решения по поводу поставленных выше вопросов (уже не разбивая на пункты):

— Данные представляют собой некоторое (неизвестное заранее) количество вещественных чисел.

— Исходные данные записаны в десятичном представлении в обычном текстовом файле, записи чисел разделены пробелами или переводами строк.

— Для указанного имени файла с данными числа последовательно и автоматически вводятся в программу, пока это удается сделать. Результат вычисляется для всех введенных чисел.

— Ответ — вещественное число, равное среднему арифметическому введенных чисел.

— Ответ выводится на экран в понятной читаемой форме.

— Последовательно (в цикле) накапливается сумма и количество введенных чисел; по окончании ввода вычисляется среднее арифметическое.

— Используются стандартные функции чтения данных из файла и конкрукции цикла.

— Некорректные ситуации: не удалось ввести ни одного числа по разным причинам (файл данных не существует, недоступен, или в нем нет правильных данных) —

выводится соответствующее диагностическое сообщение и программа завершает работу.

Теперь для иллюстрации приведем простой (но почти полноценный) код этой программы на языке C. На примере этого кода будем разбираться что и почему так делается, а затем уже познакомимся с более широкими возможностями языка. Комментироваться будет устно на лекции.

И предварительно запишем наш алгоритм на “псевдокоде”, который далее переведем в конкретный алгоритмический язык.

```
необходимые объявления по правилам языка

основная функция:
    объявления необходимых переменных
    попытка открыть файл с данными
    если файл не открылся
        диагностическое сообщение
        завершить работу
    конец если
    // тут файл уже открыт

    сумма = 0
    счетчик = 0                                    | стоит оформить
    цикл пока удается прочитать очередное число | как отдельную
        накапливаем сумму                          | функцию
        увеличиваем счетчик                        |
    конец цикла                                    |
    // тут числа закончились                       |
                                                   |
    среднее = сумма / счетчик                      |
    !!! возможно деление на 0 !!!                  |

    выдать среднее
конец функции
```

Теперь можно все это переписать по правилам языка C.

```
#include <stdio.h>

double CalculateMeanValue (FILE *f);

int main(void)
{
    double res;
    FILE * f;

    f = fopen("input.txt", "r");
    if (!f) {
        fprintf(stderr, "Error opening data file\n");
        return -1;
    }
```

```
    res = CalculateMeanValue(f);

    fprintf(stdout, "Mean value is %f\n", res);
    fclose(f);
    return 0;
}

double CalculateMeanValue (FILE *f)
{
    double sum = 0, a;
    int n;
    for (n = 0; fscanf(f, "%lf", &a) == 1; ++n) {
        sum += a;
    }
    if (n != 0) { sum /= n; }
    return sum;
}
```

Здесь сразу использовано множество конструкций языка C. Можно было бы записать тот же алгоритм другими конструкциями. Что и как писать зависит от ваших знаний и опыта (ну, и от поставленных требований). Разбираем каждую строчку — зачем нужна и что делает.

Что здесь “плохо”.

1. Весь код программы записан в одном файле. Это у нас первый и последний раз!

2. Имя файла данных жестко фиксировано, нельзя применить одну программу к разным файлам.

3. Если данные не прочитались, то ответ будет 0, что совпадает со случаем, когда среднее арифметическое непустого набора чисел есть тоже 0. Ситуация с наличием или отсутствием данных в файле никак не отслеживается.