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

Лекция 1. Обзор языка C

Здесь будет даваться краткий конспект сказанного на лекциях. Это позволит слушателям не тратить время на подробное конспектирование, однако пояснения и обсуждения многих моментов проводится именно на лекции и не всегда отражается в ее конспективной записи. Таким образом, для ознакомления с материалом стоит прослушать именно лекцию, а содержимое данного файла сохранит основные ключевыет 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 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, предумотренные 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 $\rightarrow$ процедуры $\rightarrow$ результат

объектно-ориентированное программирование:

объекты в исходном состоянии $\rightarrow$ взаимодействие объектов $\rightarrow$ 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. Спецификация входных данных задачи и требуемого результата.
2. Разработа математического алгоритма решения в виде набора процедур.
3. Реализация алгоритма в виде программы на алгоритмическом языке.
4. Отладка и тестирование программы.

Пункты 1–3 этого плана требуют знакомства с конкретным языком.

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.

Как всегда, для этого 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 и C++, покольку это языки, наиболее близкие к внутренней "кухне" вычислительного процесса, и, с другой стороны, достаточно универсальные и эффективные 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 или язык C++, ограничиваясь на данном этапе его 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;
2. простые переменные и составные типы;
3. операции с базовыми типами;
4. операторы и управляющие конструкции;
5. структуризация записи программы (блоки, объявления, процедуры и т.п.);
6. видимость объектов (локальные, глобальные, статические и т.д.);
7. организация ввода-вывода, внешние и стандартные библиотеки;
8. прочие дополнительные возможности;
9. 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 (и 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.

1. что собой представляют исходные данные, откуда они берутся?
2. по какой сжеме осуществляется ввод этих данных в работающую программу?
3. что предполагается получить в качестве ответа?
4. в какой форме этот ответ должен быть предоставлен пользвателю?
5. как будут представлены данные в вашей программе и по каким алгоритмам они будут обрабатываться?

6. как вы (ваша программа) должна реагировать на возможные некорректные ситуации и что это могут быть за ситуации?

и только после решения всех этих вопросов можно уже выбирать язык программрования и приниматься за кодирование:

7. реализовать код вашей программы на выбранном алгоритмичском языке.

Ответы на эти вопросы зависят от разных факторов. Например, они могут быть сформулированы заказчиком или вы должны принять соответствующие решения самостоятельно. Мы решим это так:

1. исходные заднные записаны в десятичном представлении в обычном текстовом файле, записи чисел разделены пробелами или переводами строк.

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

3. ответ — среднее арифметическое введенных чисел.

4. ответ выдается на экран в понятной читаемой форме, также диагностические сообщения при некорректных ситуациях.

5. ну, тут писать долго, проговорим устно.

6. некорректные ситуации: файл данных не существует, не читается, в нем фактически нет чисел; на экран выводятся соответствующие сообщения.

Теперь для иллюстрации приведем примитивный (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.

```
#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. Все записано 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.