

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-2"
на это слово будет настроен фильтр,
чтобы письмо не затерялось в массе других

Лекция 1. Обзор языка C

В 2021–2022 учебном году курс лекций, читающихся на механико-математическом факультете МГУ, проходит в дистанционном режиме. Здесь приводится конспект сказанного на этой лекции. Это позволит слушателям не тратить время на 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 ключевые 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 алгоритмов 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 курса 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 вычисли-

тельная 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 помощи выполнения “процедур, реализующих алгоритм”.

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

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

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

1. Спецификация входных данных задачи и требуемого результата.
2. Разработка математического алгоритма решения в виде набора процедур.
3. Реализация алгоритма в виде программы на алгоритмическом языке.
4. Отладка и тестирование программы.

Пункты 1-3 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 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++, поскольку 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

- изоощренные и математически сложные алгоритмы
- большие массивы данных самых разнообразных форматов
- сложные взаимосвязи между данными
- требование эффективности и надежности программ

— интеграция с существующими библиотеками программ

Упомянутые выше языки:

Basic - простой, не приспособен для серьезных задач

Pascal - учебный, не поддерживается для серьезных задач

C C++ - универсальный, поддерживается везде, разные уровни использования

Java - универсальный, поддерживается везде, сложен при начальном изучении, неэффективен в сложных ситуациях

Python - прост в изучении, удобен для простых но не пригоден для сложных задач и есть еще много других языков ...

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 построения языка, оставляя технические детали техническим справочникам.

Краткий и неполный обзор построения языков C и C++

1. Базовые типы данных

целый:		модификаторы	диапазоны
char	- 1 байт (k=8)	signed	$-2^{(k-1)}$ $+2^{(k-1)} - 1$
short	- 2 байта (k=16)	unsigned	0 $2^k - 1$
long	- 4 байта (k=32)		
int	- 4 байта		

также возможны и более "длинные" представления

вещественный:	стандарт IEEE-754 floating point representation
float	- 4 байта
double	- 8 байтов

также возможны и более "длинные" представления

логический

bool true false (также для любого базового типа 0 - false, не 0 - true)

Про указатели и ссылки будет отдельный разговор

константы

целые

321 -7 +2345

0x10 0x2F 0x341CD4FA

'A' '2' 's' - целый код данного символа в текущей кодировке

int c = 'A' + 2; codetable

вещественные (double)

1.25 3.1415926 -7. 0.00001 +.123

123.e-3 1.e+20 -345.678e+2

1.33f (float)

"строки" - массив кодов символов в определенной кодировке

"the world of code" ASCII string

2. Простые переменные и составные типы

простые переменные

```
int x, y, z;                - значения могут быть не определены
                            но иногда могут быть = 0   (обсудим позже)

int i, j, k, _a, d_35;
double f, u123, result;
char r2d2, c3po;
float val;
unsigned long abc, size;
signed short m;                j = m;
```

массивы

```
int a[100];                a[0] a[1] ... a[99]  !!!! int k;    k = 1000;
double b, c[1024];        c[0] ... c[1023]  !!!! error    z = a[k] + 1;
                            !!!! a[100] - в массиве нет такого элем
```

!!! проверка индексов на попадание в границы не делается !!!

инициализация в объявлении

```
int x = 10, y = -3;        - значения определены
double e = 2.71828;
int r[4] = {3, 2, -7, 6};
double a[] = {3.3, 4.1, -2.22222};    создается массив на указанное количество
                                        значений, т.е. здесь на 3
```

Есть еще структуры, объединения, битовые поля — о них будет отдельный разговор.