Особенности формирования алгоритмического мышления бакалавра по направлению подготовки «Информатика и вычислительная техника»

Клим Евгений Игоревич,студент 1-го курса

Димитровградский инженерно-технологический институт – филиал федерального государственного автономного образовательного учреждения высшего профессионального образования «Национальный исследовательский ядерный университет «МИФИ»

Введение

Понятие алгоритмического мышления

Алгоритмическим мышлением можно назвать систему мыслительных приёмов направленных на решение задач. Следует выделить две стороны понимания алгоритмического мышления. Первая, определить чужой алгоритм. Вторая, создать свой. Если при выполнении задачи необходимо взаимодействовать с чем-либо, нужно понимать, как оно устроено. Только потом можно создавать свой алгоритм. Трудно представить задачу, решая которую, не нужно ни с чем взаимодействовать. Даже если вы просто пытаетесь открыть окно, нужно знать алгоритм «окна».

Чем легче мы понимаем чужие алгоритмы и строим свои, тем лучше. Другими словами, полезно знать и понимать, как и что устроено. Такой тип мышления очень сильно помогает освоению многих знаний и навыков. Способность мыслить точно, формально, если это нужно, становится одним из важнейшим качеством человека в современном высокотехнологичном мире.

Как и всё, что требует развития, алгоритмическое мышление нужно развивать. Можно тренировать бессистемно, например, играя в стратегические игры. Но так развитие получается одностороннее. Хуже всего будет развито понимание свойств и ограничений. Пониманием и построением алгоритмов занимается информатика. Информатика так же изучает их свойства. Логично предположить, что изучение дисциплин, связанных с информатикой и программированием, разовьёт алгоритмическое мышление наилучшим образом.

Аннотация. Данная работа посвящена анализу проблемы формирования алгоритмического мышления и уточнения его роли, места, уровня развития в современном курсе информатики и ИВТ в целом.

Формирование алгоритмического мышления студентов ИВТ (бакалавр)

Технология решения задач на вычислительной технике – это не только составление программы и получение загрузочного модуля, а формирование модели, составление алгоритма, отладка программы и ее тестирование. Решение задачи на компьютере невозможно без создания алгоритма.

Умения решать задачи, разрабатывать стратегию ее решения, выдвигать и доказывать гипотезы опытным путем, прогнозировать результаты своей деятельности, анализировать и находить рациональные способы решения задачи путем оптимизации, детализации созданного алгоритма, представлять алгоритм в формализованном виде на языке исполнителя позволяют судить об уровне развития алгоритмического мышления студента. Поскольку алгоритмическая подготовка является наиважнейшей для будущего специалиста и без нее немыслимо знанием предмета на приемлемом уровне, то предлагается направить на усиление этой подготовки резервы, предусмотренные дисциплинами, устанавливаемыми вузом.

Основная идея, реализуемая по ходу всего обучения, заключается в том, что в процессе изучения очередного алгоритма студент совершает ряд учебных действий. В рамках первых тем алгоритм часто предоставляется студенту в виде, близком к итоговому тексту, реализующей его программы. От студента требуется совершить минимальный набор действий – оформить программу в соответствии с требуемой структурой, описать переменные и т. д. Постепенно формальные описания алгоритмов становятся все менее детализированными, от студента требуется уже больше именно «программистских» навыков.

Изучение алгоритмов готовых программ

Как отмечалось ранее, прежде чем составить свой алгоритм, нужно изучить и понять работу уже готового алгоритма

Рассмотрим программу, написанную в Turbo Pascal

```
1 procedure find(x,y: integer); {x и y – координаты ячейки, от которой осуществляется поиск}

2 begin

3 if (x=xk)and(y=yk) then begin {Проверяем, не является ли данная ячейка конечной.}

4 writeln;

5 writeln();

6 readkey
```

7 halt {Если проверка условия дала положительный результат, выдаем сообщение, что путь существует ‘there is a way’, и завершаем вызов процедуры. Если процедура завершится стандартным способом, значит, пути не существует, о чем и надо будет сообщить в основной программе.}

8 end;

9 lab[x,y]:=2; {Помечаем текущую ячейку как просмотренную – 2, что препятствует ее повторному просмо- тру во время поиска пути}

10 if lab[x+1,y]=1 then find(x+1,y) {Моделируем шаг вниз, если он возможен (ячейка свободна и не просмотрена) и вызываем процедуру поиска пути для данной ячейки. Аналогично на шагах 11, 12, 13 где моделируем шаги вверх, вправо и влево соответственно. Замечание: каждый вызов процедуры производится тогда и только тогда, когда предыдущие попытки вызова процедуры не были выполнены.}

11 if lab[x-1,y]=1 then find(x-1,y);

12 if lab[x,y-1]=1 then find(x,y-1);

13 if lab[x,y+1]=1 then find(x,y+1);

14 lab[x,y]:=1 {Текущая ячейка помечается как свободная, что фактически моделирует шаг назад, для случая если мы зашли в «тупик» и дальнейшее движение не возможно.}

15 end.

Обучаемому уже дан алгоритм «поиска в лабиринте», и ему необходимо лишь задать исходные данные, описать переменные и вызвать процедуру поиска пути в лабиринте от начальной ячейки.

Рассмотрим вторую задачу, решаемую с помощью рекурсии

Обобщенный алгоритм решения за- дачи о «стабильных браках».

Процедура Try (m) – алгоритм поиска супруги для мужчины m, поиск идет в порядке списка пред- почтений мужчины m:

```
1 procedure search (m: man);
2 var r: rank;
3 begin for r := 1 to n do {выбор r-й претендентки для m};
4 if {допустимо} then {запись брака};
5 if m  {не последний}
6 then search (successor(m))
7 else {записать стабильное множество}
8 end;
9 {отменить брак}
10 end
11 end
```

12 end Try.

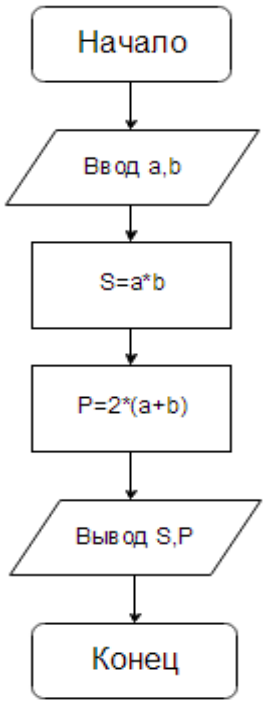
Здесь же задача перед студентом ставится более сложная, так как ему дан лишь обобщенный алгоритм и ему уже необходимо не только заполнить исходные данные, но и дописать алгоритм прежде чем переходить к выполнению программы.

Создание собственных алгоритмов

Для решения поставленных задач, необходимо составить свой алгоритм последовательности действий программы.

Для начала рассмотрим простейшую задачу на нахождение площади прямоугольника.

Первое что нужно сделать – составить блок-схему к данной задаче:



Структура программы, решающей данную задачу, тоже проста:
Описание переменных;
Ввод значений сторон прямоугольника;
Расчет площади прямоугольника;
Расчет периметра прямоугольника;
Вывод значений площади и периметра;
Конец.

```
1.  Program Rectangle;
2.  Var a,b,S,P: integer;
3.  Begin
4.  write('Введите стороны прямоугольника!');
5.  readln(a,b);
6.  S:=a*b;
7.  P:=2*(a+b);
```

```
8.      writeln('Площадь прямоугольника: ',S);
9.      write('Периметр прямоугольника: ',P);
10.     End.
```

Для развития алгоритмического мышления необходимо постоянно усложнять задачи: добавлять циклы, условия, массивы и т.д.

Так же можно изучать сразу несколько языков программирования, таких как Java, c++, Delphi и т.д.

Заключение

Алгоритмическое мышление у студента формируется постепенно, по мере усложнения задач программирования и поиска кратчайшего их решения. Не менее важным аспектом к развитию алгоритмического мышления является изучение сторонних от программирования дисциплин, таких как философия, математика, физика, психология и т.д.