**Лабораторная работа 2**

# Программирование разветвляющихся алгоритмов

## Цель работы

Целью работы является приобретение студентами следующих навыков:

- использование метода пошаговой детализации при разработке программ,

* использование логических операций для составления «сложных» условий,
* использование операторов ***if, if-else*** и ***switch*** при программировании разветвляющихся алгоритмов,
* изображение операторов ***if, if-else*** и ***switch*** на схемах алгоритмов.

Планируемое время выполнения работы- 6 часов.

## Задание

Вывести на экран показания часов.

Исходные данные – часы и минуты в формате целых чисел. Результат должен быть представлен в смешанной текстово-числовой форме с учетом падежных окончаний, например:

2 48 – 2 часа 48 минут ночи,

5 00 – 5 часов утра ровно,

12 00 – полдень,

12 23 – 12 часов 23 минуты дня,

13 12 – 1 час 13 минут дня,

21 33 – 9 часов 33 минуты вечера,

00 00 – полночь,

03 01 – 3 часа 1 минута ночи.

Использовать следующие границы частей суток:

5 час. <= утро < 12 час.,

12 < день < 18 час.,

18 <= вечер <= 23 час.,

0 час. < ночь < 5 час.

В случае ввода невозможных показаний часов выдать соответствующее сообщение, например:

24 03 – введены недопустимые данные.

По результатам работы представить отчет.

## Рекомендации по разработке программы

Создание программы - процесс сложный, поэтому практически с любо­го этапа возможен возврат на предыдущие этапы для исправления ошибок или принятия других проектных решений. Чаще всего такого рода возвраты являются следствием ошибок, допущенных при разработке алгоритма.

Для уменьшения ошибок в алгоритмах программ при их разработке следует использовать метод пошаго­вой детализации.

При использовании данного метода разработку алго­ритмов выполняют поэтапно. На первом этапе описывают решение постав­ленной задачи, выделяя подзадачи и считая их решенными. На следующем - аналогично описывают решение подзадач, формулируя уже подзадачи следу­ющего уровня. Процесс продолжают до тех пор, пока не дойдут до подзадач, алгоритмы решения которых очевидны. При этом, описывая решение каждой задачи, желательно использовать не более одной-двух конструкций, таких как цикл или ветвление, чтобы четче представлять структуру программы.

Шаг1. Определяем общую структуру программы.

Разбиение сложного алгоритма на подзадачи следует выполнять таким образом, чтобы каждая подзадача имела физический смысл и правильность ее выполнения можно было бы проверить, не выполняя последующие действия или выполняя их небольшую часть, например, печать результатов разработанных подзадач.

Перед тем, как рисовать схему алгоритма, нужно описать используемые в алгоритме переменные, указав их назначение и тип, например:

int h – значение часов;

int m– значение минут.

На рисунке 1 показан вариант укрупненной схемы алгоритма программы.

После того, как схема алгоритма нарисована, все ее ветви нужно проверить с помощью разработанных для этих целей контрольных примеров.

Шаг 2. Детализируем операцию ввода и контроля исходных данных. После завершения разработки детального алгоритма первой подзадачи ее следует закодировать, заменяя, если необходимо, неразработанные подзадачи «заглушками», выполнить и протестировать. Например, для ввода и контроля исходных данных остальные действия не требуются и для отладки этого блока достаточно выводить на печать введенные данные. Заглушка не выполняет никаких действий, а только печатает свое имя.

***Не следует сразу кодировать всю программу, так как в этом случае значительно сложнее обнаруживать и исправлять ошибки***. Пишите и отлаживайте программу по частям, так, чтобы на каждом шаге разработки программа могла выполняться (была бы «живой»). Это позволит обнаружить и устранить возможные ошибки на ранней стадии их возникновения и существенно сократить общее время разработки программы.

Шаг 3. Детализируем подзадачу определения времени суток. Ее удобнее выполнить до обработки часов, так как при выводе результатов падежные окончания могут измениться. Например, вместо 15 часов мы должны печатать 3 часа дня. Это можно учесть, если запомнить время суток в переменной типа int, например (1 - утро, 2-день, 3- вечер, 4-ночь) и выполнить операцию h=h-12, если h>12.

Шаги 4, 5. Детализируем операции обработки часов и минут. Результаты обработки запоминаются в переменных типа int, например,

h\_padeg - падежное окончание для вывода числа часов,

m\_padeg - падежное окончание для вывода числа минут.

Возможные значения h\_padeg равны:

1, если надо печатать «час»;

2, если надо печатать «часа»;

3, если надо печатать «часов».

Для определения падежных окончаний слов при печати значений минут эффективно использовать операцию деления по модулю (эта операция определена для целых чисел и в С++ обозначается знаком **%**), а для выделения диапазона чисел с одинаковыми значениями падежных окончаний использовать операторы больше, меньше, больше равно, меньше равно (>, <, >=, <=).

Например, если ( m>4 && m<21 ), то надо печатать ***минут,*** а иначе,

если ( m%10 == 1), то есть для m=21, 31, 41 и 51 надо печатать ***минута***.



Рис.1. Укрупненная схема алгоритма

Шаг 6. При выводе результатов значения переменных h\_padeg и m\_padeg будем использовать в операторе switch в качестве ключа при печати результата с помощью операторов потокового вывода (формирование строки на экране монитора операторами потокового вывода рассматривалось в лабораторной работе 1).

Таким образом, на каждом шаге решается одна достаточно простая задача, что существенно облегчает разработку программы и является основным достоинством метода пошаговой детализации.

Например, на первом шаге мы вводим исходные данные (время в виде двух целых чисел: часы и минуты), выполняем проверку введенных данных, чтобы исключить ввод недопустимых значений, распечатываем введенные данные и выполняем тестирование, то есть проверку ***всех*** возможных вариантов ввода . На этапе отладки программы всегда следует распечатывать введенные данные, так как неверный ввод – типичный источник ошибок. После успешного тестирования отладочную печать нужно закомментировать (не удалить), ибо согласно одному из законов Мерфи «последняя ошибка в программе на самом деле предпоследняя» и как только вы удалите отладочную печать она Вам тут же потребуется.

В программировании задачу обычно можно решить несколькими способами. Какой из возможных алгоритмов следует использовать? Основными характеристиками алгоритмов являются сложность (понятность), время выполнения, занимаемая оперативная память, время разработки. Для выполнения заданий лабораторных работ (и не только) следует использовать наиболее понятный (простой) алгоритм. Такие алгоритмы (если не предъявляются специальные требования к времени выполнения или занимаемой оперативной памяти) обычно имеют лучшие обобщенные характеристики и меньшее время разработки.

Использование метода пошаговой детализации способствует разработке понятных алгоритмов и сокращает время разработки программы, делая его прогнозируемым.

Литература:

1. Р. Лафоре. Объектно-ориентированное программирование в С++. Издательство ПИТЕР, 2004 г.
2. Т.А. Павловская, Ю.А. Щупак. С/С++ Структурное программирование. Практикум. ПИТЕР, 2003 г.
3. Петрова. Основы программирования. Высшая школа. 2005 г.

## Требования к отчету.

Отчет по лабораторной работе должен состоять из 4-х разделов, отражающих основные этапы разработки программы:

- Постановка задачи;

- Разработка алгоритма;

- Кодирование (соответствующий раздел отчета называется «Текст программы»);

- Тестирование (соответствующий раздел отчета называется «Анализ результатов»).

В разделе «Постановка задачи» должен быть приведен текст задания и согласованные с преподавателем (заказчиком) уточнения, если они требуются (например, как выводить время 00:03 - 0 часов 3 минуты ночи или 12 часов 3 минуты ночи, или существует ли время 24:00).

Раздел «Разработка алгоритма» должен содержать следующую информацию:

* краткое описание (обоснование) алгоритма,
* описание входных, выходных и вспомогательных данных с указанием их идентификаторов и типов,
* схему алгоритма, состоящую из двух частей: общей (укрупненной) схемы и уточненной схемы одного из блоков:
  + Ввод и контроль исходных данных;
  + Обработка часов;
  + Обработка минут;
  + Обработка времени суток.

(Номер блока равен остатку от деления порядкового номера студента в списке группы на 4 плюс 1.)

В разделе «Текст программы» должен быть приведен листинг программы, включающий необходимые комментарии.

В разделе «Анализ результатов» должны быть приведены результаты проверки работы всех ветвей алгоритма. Результаты следует оформить в виде таблицы, а для двух наборов исходных данных должны быть приведены распечатки экранов. Для экономии краски изображения экранов должны иметь белый фон, для чего их нужно предварительно обработать в графическом редакторе (Paint).

Отчет должен быть распечатан на принтере на листах бумаги формата А4, скрепленных в левом верхнем углу с помощью степлера, и подписан исполнителем с указанием даты сдачи отчета преподавателю.

На последней странице показан титульный лист, который должен использоваться в отчетах по всем лабораторным работам курса (изменяться должны только номер, название лабораторной работы и фамилии студента и преподавателя).

**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «Основы информатики»

Отчет по лабораторной работе №2

«Программирование разветвляющихся алгоритмов»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-13 |  | преподаватель каф. ИУ5 |
| Иванов Иван |  | Петров П.И. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2012 г.