– к битовым операциям необходимо прибегать при работе с некоторыми системными функциями;

– битовые операции часто приходится использовать при работе с графикой (имеется в виду программ с графикой).

В С имеются следующие битовые операции:

& – битовое и;

| – битовое или;

^ – битовое исключающая или;

~ – инвертируют каждый разряд;

<< – поразрядный сдвиг влево;

>> – поразрядный сдвиг вправо.

Таким образом, можно сказать, что битовое поле – это своеобразная структура, которая позволяет работать с отдельными битами. Не смотря на то что, при решении задач можно использовать побитовые операторы, битовые поля позволяют создавать более простые и эффективные программы.

УДК 621

Рудакова В. О.

## ОСНОВНЫЕ ПРАВИЛА ПРОГРАММИРОВАНИЯ

*БНТУ, Минск*

*Научный руководитель Дробыш А. А.*

Правила здесь довольно общего характера по своей природе, они совсем не затрагивают техники программирования на Си или Си++, а скорее рассматривают более общий процесс проектирования и разработки программы. В известном смысле, большинство правил предназначены для управленцев; программисты их часто знают, но у них нет свободы, необходимой, чтобы воспользоваться своими знаниями.

Не решайте проблем, которых не существует. Решайте конкретную проблему, а не общий случай. Интерфейс пользователя

не должен быть похожим на компьютерную программу (принцип прозрачности).

Многие слышали, что лучшим пользовательским интерфейсом из когда-либо разработанных является карандаш. Его назначение тотчас же понятно, для него не нужно руководство пользователя, он готовится к работе без особой суеты. Однако наиболее важным свойством является прозрачность. Когда вы пользуетесь карандашом, то думаете о том, что вы пишите, а не о самом карандаше. Подобно карандашу, лучшими компьютерными интерфейсами являются те, которые скрывают сам факт того, что вы обращаетесь к компьютеру.

Не путайте легкость в изучении с легкостью в использовании. После того, как вы научились, карандашом пользоваться очень легко. Главная проблема здесь состоит в том, что для опытного пользователя часто требуется совершенно другой интерфейс, чем для начинающего. Дополнительная помощь типа «горячих» клавиш не решает эту проблему; старый неуклюжий интерфейс пользователя все еще мешает продуктивности, и нет особой разницы: откроете ли вы меню при помощи «горячей» клавиши, или мышью. Здесь проблема в самом меню.

Производительность может измеряться числом нажатий клавиш. Интерфейс, требующий меньше нажатий клавиш (или других действий пользователя типа щелчков мышью), лучше того, который требует много нажатий для выполнения одной и той же операции, даже если такие виды интерфейсов обычно сложнее в освоении. Подобным образом пользовательскими интерфейсами, скрывающими информацию в меню или за экранными кнопками, обыкновенно труднее пользоваться, потому что для выполнения одной задачи необходимо выполнить несколько операций (вызвав подряд несколько спускающихся меню).

Если вы не можете сказать это по-английски, то вы не сможете выполнить это и на Си/Си++.

Акт записи на английском языке описания того, что делает программа, и что делает каждая функция в программе, является критическим шагом в мыслительном процессе. Хорошо построенное, грамматически правильное предложение – признак ясного мышления. Если вы не можете это записать, то велика вероятность того, что вы не полностью продумали проблему или решение. Плохая грамматика и построение предложения являются также показателем небрежного мышления. Поэтому первый шаг в написании любой программы – записать то, что делает программа, и как она это делает.

Начинайте с комментариев.

Если вы последовали совету в предыдущем правиле, то комментарии для вашей программы уже готовы. Для того чтобы получить документированное описание реализации, вы просто писали и добавляли вслед за каждым абзацем блоки кода, реализующие качества, описанные в этом абзаце.

Разбивайте сложные проблемы на задачи меньшего размера. Проблема должна быть хорошо продумана перед тем, как она сможет быть решена.

Хорошо спроектированная программа не только работает лучше (или просто работает), но и может быть написана быстрее и быть проще в сопровождении, чем плохо спроектированная. Кроме того, скверно спроектированные программы труднее реализовать. Тот аргумент, что у вас нет времени на проектирование, потому что вы «должны захватить рынок программ как можно скорее», просто не выдерживает никакой критики, потому что реализация плохого (или никакого) проекта требует гораздо больше времени.

Вовлекайте пользователей в процесс проектирования. Заказчик всегда прав.

Ни одной программе не добиться успеха, если ее проектировщики не общаются непосредственно с ее конечными пользователями.

Опытный проектировщик зачастую предлагает лучшее решение проблемы, чем то, что придумано конечным пользователем, в особенности, если учесть, что конечные пользователи часто предлагают интерфейсы, созданные по образцу программ, которыми они постоянно пользуются. Несмотря на это, вы должны убедить пользователя, что ваш способ лучше, перед тем, как его реализовать. «Лучший» интерфейс не является лучшим, если никто, кроме вас, не сможет (или не захочет) им пользоваться.

УДК 621.762.4

Руйчева А. П.

## СОВРЕМЕННЫЕ ЯЗЫКИ ПРОГРАММИРОВАНИЯ

*БНТУ, Минск*
*Научный руководитель Дробыш А. А.*

За последние несколько десятилетий основой программирования стали Java, C и его производные, Python, Ruby. Они проверены временем, тысячами пользователей и разработчиков. Однако время диктует новые требования к языкам: с каждым годом необходимо увеличение быстродействия, как исполняющей машины, так и оператора.

В последние годы появляется все больше и больше языков программирования. Некоторые из них стали довольно популярными, некоторые же напротив не прижились.

Go

Пожалуй, язык получившую наибольшую популярность – Go или Golang от Google.