How to approach learning a new SDK/API/library?

Asked 15 years, 10 months ago Modified 13 years, 3 months ago
Viewed 5k times

8

Let's say that you have to implement some functionality that is not trivial (it will take at least 1 work week). You have a SDK/API/library that contains (numerous) code samples demonstrating the usage of the part of the SDK for implementing that functionality.

How do you approach learning all the samples, extract the necessary information, techniques, etc. in order to use them to implement the 'real thing'. The key questions are:

- Do you use some tool for diagramming of the control flow, the interactions between the functions from the SDK, and the sample itself? Which kind of diagrams do you find useful? (I was thinking that the UML sequence diagram can be quite useful together with the debugger in this case).
- How do you keep the relevant and often interrelated information about SDK/API function calls, the general structure and calls order in the sample programs that have to be used as a reference - mind maps, some plain text notes, added comments in the samples code, some refactoring of the sample code to suit

your personal coding style in order to make the learning easier?

sdk

Share

Improve this question

Follow

asked Feb 19, 2009 at 11:34

Singulus

1,998 ● 2 ● 16 ● 16

3 Answers

Sorted by: Highest score (default)

7

Personally I use the prototyping approach. Keep development to manageable iterations. In the beginning, those iterations are really small. As part of this, don't be afraid to throw code away and start again (everytime I say that somewhere a project manager has a heart attack).

If your particular task can't easily or reasonably be divided into really small starting tasks then start with some substitute until you get going.

You want to keep it as simple as you can (the proverbial "Hello world") just to familiarize yourself with building, deploying, debugging, what error messages look like, the simple things that can and do go wrong in the beginning, etc.

I don't go as far as using a diagramming tool sorry (I barely see the point in that for my job).

As soon as you start trying things you'll get the hang of it, even if in the beginning you have no idea of what's going on and why what you're doing works (or doesn't).

Share Improve this answer

Follow

answered Feb 19, 2009 at 11:40

cletus

625k ● 169 ● 917 ● 945

> I have to agree. Code prototypes first aid understanding. Without them and the understanding they enable you can't hope to produce an accurate model diagrams anyway.
> – Chris Becke Mar 15, 2009 at 19:27

2

I usually compile and modify the examples, making them fit something that I need to do myself. I tend to do this while using and annotating the corresponding documents. Being a bit old school, the tool I usually use for diagramming is a pencil, or for the really complex stuff two or more colored pens.

Share Improve this answer

Follow

answered Feb 19, 2009 at 11:42

SmacL

22.8k ● 15 ● 99 ● 151

I am by no means a seasoned programmer. In fact, I am learning C++ and I've been studying the language primarily from books. When I try to stray from the books (which happens a lot because I want to start contributing to programs like LibreOffice), for example, I find myself being lost. Furthermore, when I'm using functionality of the library, my implementations are wrong because I don't really understand how the library was created and/or why things need to be done that way. When I look at sample source code, I see how something is done, but I don't understand why it's done that way which leads to poor design of my programs. And as a result, I'm constantly guessing at how to do something and dealing with errors as I encounter them. Very unproductive and frustrating.

Going back to my book comment, two books which I have ready from cover to cover more than once are Ivor Horton's Beginning Visual C++ 2010 and Starting Out with C++: Early Objects (7th Edition). What I really loved about Ivor Horton's book is that it contained thorough explanation of why something needs to be done a certain way. For example, before any Windows programming began, lots of explanation about how Windows works was given first. Understanding how and why things work a certain way really helps in how I develop software.

So to contribute my two pennies towards answering your question. I think the best approach is to pick up well written books and sit down and begin learning about that library, API, SDK, whatever in a structured approach that

offers real-world examples along with explanations as to how and why things are implemented as they are.

I don't know if I totally missed your question, but I don't think I did.

Cheers!

This was my first post on this site. Don't rip me too hard. (:

Share Improve this answer Follow

answered Aug 26, 2011 at 1:26

user898058