Junior Developper

Pytanie 1: Co to jest struktura w iOS?

Struktury to nazwane typy, które pozwalają na grupowanie elementów w jedną zmienną. Struktura w języku Swift wyglądałaby następująco:

struct SomeStruct {

var name: String

var attribute: String

}

podczas gdy w Objective-C wyglądałoby to tak:

struct City {

NSString *name;

NSString *attribute;

}

Dobrym pytaniem uzupełniającym dla kandydata jest wyjaśnienie różnicy między klasami i strukturami oraz wyjaśnienie, gdzie można je efektywniej wykorzystać. W języku Swift struktury są kopiowane, a klasy są przywoływane. Tak więc w przypadku dużych obiektów będą przekazywane jako wartości, a nie jako referencje. Swift używa struktur, aby poprawić wydajność, nawet w przypadku obiektów String i Array. Struktury są lepsze, jeśli

• Celem jest hermetyzacja kilku wartości danych.

• Struktura nie musi dziedziczyć wartości z innych istniejących typów.

Pytanie 2: Co to jest framework w iOS?

Framework to „szkielet", który został zapakowany w pakiet, który ma być używany i rozpowszechniany w innych aplikacjach. Struktura może zawierać szeroką gamę dodatkowych zasobów (obrazy, napisy, które zostały już zlokalizowane, pliki danych, obiekty interfejsu użytkownika itp.). O ile framework nie został udostępniony publicznie, zazwyczaj zawiera pliki .h potrzebne do jego uruchomienia. Dobrym pytaniem do zagłębienia się w ten temat jest zadanie kandydatowi różnicy między frameworkiem a biblioteką. Bibliotekę można zdefiniować jako zbiór funkcjonalności gotowych do połączenia i użycia w aplikacji. Framework może być biblioteką, zestawem zawierającym wiele bibliotek, zbiorem skryptów lub czymkolwiek, czego potrzebujesz do stworzenia aplikacji. Jest to termin bardziej abstrakcyjny. Jeśli kandydat zostanie poproszony

Na konkretnych przykładach biblioteka mogłaby być biblioteką sieciową. Framework może być menedżerem wtyczek lub systemem GUI

Pytanie 3: Jak możesz przechowywać informacje w aplikacji na iOS?

iOS zapewnia natywnie kilka różnych sposobów utrwalania danych w aplikacji. Zawierają

Keychain: Uważa się, że służy do przechowywania poufnych danych w bezpieczny sposób (powtarzające się przykłady to loginy i hasła).

Baza danych SQLite: jeśli dane, które mają być przechowywane, są złożone lub wymagają dużej ilości ustrukturyzowanych danych, najlepszym rozwiązaniem jest użycie bazy danych SQLite.

Podstawowe dane: to jest struktura trwałości Apple. Opiera się na wykresie obiektów, który opisuje obiekty, które powinny zostać zapisane. Następnie możesz bezpośrednio pracować z tymi obiektami, a framework zajmuje się dbałością o zapisanie go w bazie danych.

Pliki: można je zapisać bezpośrednio w aplikacji (ze względów bezpieczeństwa aplikacja może uzyskać dostęp tylko do własnego kontenera aplikacji). Jeśli chcesz zagłębić się głębiej, potencjalny kandydat może na przykład omówić powody wyboru oprogramowania SQLite zamiast danych podstawowych. Jednym z powodów może być to, że SQLite może być współużytkowany przez różne aplikacje na innych platformach, podczas gdy Core Data jest strukturą dostępną wyłącznie dla Apple.

Ustawienia domyślne użytkownika: można ich użyć do utrwalenia niewielkiej ilości danych. Typowe przykłady są częścią konfiguracji wymaganej do uruchomienia aplikacji. Wartości domyślne użytkownika mogą utrwalać typy pierwotne w systemie iOS (String, Data, Number, Date, Array, i Dictionary).

Pytanie 4: Co to jest słownik? Czy jest podobny do innych struktur w innych językach programowania?

Słownik przechowuje skojarzenia między kluczami należącymi do tego samego typu, a wartościami również należącymi do tego samego typu w kolekcji bez zdefiniowanej kolejności. Każda wartość jest powiązana z unikalnym kluczem, który działa jako identyfikator tej wartości w słowniku. Jest podobny do tego, co jest znane w innych językach jako HashMap, z pewnymi subtelnymi różnicami.

Pytanie 5: Co to jest profil informacyjny?

Profil informacyjny to zbiór jednostek cyfrowych, które w wyjątkowy sposób łączą programistów i urządzenia z autoryzowanym zespołem programistów iPhone'a i umożliwiają używanie urządzenia do testowania. Na każdym urządzeniu, na którym chcesz uruchomić kod aplikacji, należy zainstalować programistyczny profil informacyjny. Jeśli chcesz dokładniej omówić to z kandydatem, możesz zapytać go o proces podpisywania aplikacji na iOS

Pytanie 6: Co to jest ARC?

Automatyczne liczenie odwołań (ARC) to funkcja kompilatora, która zapewnia automatyczne zarządzanie pamięcią dla wszystkich obiektów Objective-C. Zamiast ręcznie zachowywać i zwalniać operacje, ARC pozwala skupić się na części kodu, która jest bardziej interesująca, na grafach obiektów i relacjach między obiektami w aplikacji. Dyskusja na temat ARC może skierować się w bardziej interesujących kierunkach. Jeśli kandydat wyświetla solidny profil, możesz chcieć omówić różnice między tradycyjnymi metodami usuwania elementów bezużytecznych. W przypadku ARC nie ma procesu w tle zwalniającego obiekty w czasie wykonywania, jak w innych technologiach

Pytanie 7: Co to jest układ automatyczny?

Układ automatyczny to sposób, za pomocą którego programiści mogą tworzyć interfejsy użytkownika, definiując relacje między elementami. Zapewnia elastyczny i wydajny system, który opisuje, w jaki sposób widoki i elementy sterujące interfejsu użytkownika są ze sobą powiązane. Korzystając z Auto Layout, możesz uzyskać niesamowity stopień kontroli nad układem, z szerokim zakresem dostosowań i uzyskać doskonały interfejs.

Pytanie 8: Jak zarządzasz zależnościami w iOS?

W przeciwieństwie do innych platform, Apple oficjalnie nie zapewnia żadnego sposobu obsługi zależności w projekcie iOS. Jako de facto standard, CocoaPods jest szeroko stosowany w społeczności iOS. Zależności dla projektów są określone w pojedynczym pliku tekstowym, zwanym Podfile, za pomocą którego CocoaPods rozwiązuje zależności między bibliotekami, pobiera wynikowy kod źródłowy, a następnie łączy go w obszarze roboczym Xcode w celu skompilowania projektu. Bardziej doświadczony kandydat z pewnością mógłby wymienić inną alternatywę dla CocoaPods, a mianowicie dość popularną Kartaginę, napisaną w języku Swift. Chcesz omówić to dalej? Następnie rozpocznij dyskusję na temat zalet Carthage vs CocoaPods.

Pytanie 9: Jak debugować i profilować w systemie iOS?

Jest to bardzo ogólne pytanie, które polega na uzyskaniu jednej odpowiedzi od potencjalnego kandydata. Aby było ciekawie, kandydat powinien umieć omawiać instrumentację w systemie iOS i dzienniki awarii. Jeśli masz znany błąd w swojej aplikacji, dobrą praktyką może być pozwolenie programistom na otwarcie twojego środowiska i sprawdzenie, jak porusza się po różnych instrumentach (debugowanie procesora itp.).

Pytanie 10: Jaka jest różnica między identyfikatorem aplikacji a identyfikatorem pakietu w systemie iOS?

Identyfikator pakietu służy do precyzyjnej identyfikacji pojedynczej aplikacji. Ciąg identyfikatora pakietu musi być jednolitym identyfikatorem typu (UTI), który zawiera tylko znaki alfanumeryczne (A – Z, a – z, 0–9), łączniki (-) i kropki (.). Ciąg powinien mieć odwrotny format DNS. Identyfikator aplikacji to dwuczęściowy ciąg używany do identyfikowania jednej lub więcej aplikacji z jednego zespołu programistów. Ciąg składa się z identyfikatora zespołu i ciągu wyszukiwania identyfikatora pakietu z kropką (.) Oddzielającą dwie części. Identyfikator zespołu jest dostarczany przez Apple i jest unikalny dla określonego zespołu programistów, podczas gdy ciąg wyszukiwania identyfikatora pakietu jest dostarczany przez Ciebie w celu dopasowania albo do identyfikatora pakietu pojedynczej aplikacji, albo zestawu identyfikatorów pakietu dla grupy aplikacji. Dlaczego Apple miałby wybierać tak mylącą nomenklaturę, to tajemnica, której nikt jeszcze nie ujawnił.

Pytanie 11: Jak działa podpisywanie kodu?

Podpisywanie kodu to proces jednoznacznego podpisywania aplikacji w celu ustawienia własnych poświadczeń (przy użyciu klucza prywatny-publiczny). Używany klucz jest przechowywany lokalnie na komputerze Macintosh, a jeśli chcesz używać innych stacji roboczych do wykonywania prac programistycznych, tożsamość również musi być udostępniona. Stary żart mówi: „Samouczek, jak łatwo podpisać aplikację na iOS w 75 krokach”. Proces nie jest prosty i bez względu na to, jak bardzo jesteś doświadczony, zawsze łatwo jest zapomnieć o niektórych krokach. Jednak programista powinien być w stanie nakreślić kluczowe etapy procesu. Uwaga: ten proces może ulec zmianie w przyszłości, zwłaszcza poszczególne etapy.

1. Najpierw musisz utworzyć plik CSR na swoim komputerze lokalnym. W tym celu musisz użyć pliku CSR, używając programu dostępu do pęku kluczy. W tym procesie należy określić pewne informacje (takie jak nazwa zwyczajowa, adres e-mail, typ szyfrowania itp.). Ten plik musi być bezpiecznie przechowywany.

2. Po wykonaniu tej czynności musisz utworzyć certyfikat w witrynie sieci Web programu dla programistów. W pewnym momencie podczas procesu asystent zapyta o utworzony wcześniej plik CSR. Następnie zostanie utworzony i pobrany certyfikat. Po dwukrotnym kliknięciu zostanie

automatycznie dodany do pęku kluczy. Aby z nim pracować, ten certyfikat musi znajdować się na komputerze.

Pytanie 12: Jaka jest różnica między ramką a granicami?

Ramka UIView w iOS jest wyimaginowanym prostokątem względem lokalizacji nadzoru, w którym się znajduje. Z drugiej strony, granica jest względnym prostokątem do własnego układu współrzędnych. Oba elementy są wyrażone jako lokalizacja, w której punktami początkowymi są x, y, a rozmiar określany jest przez szerokość i wysokość.

Pytanie 13: Jak rzutować między typami?

W przypadku Objective-C i Swift odpowiedź będzie inna. Ogólnie i ponieważ Objective-C jest nadzbiorem języka C, rzutowanie na typ działa w podobny sposób. Aby rzutować typ, możemy wykonać następujące czynności:

int i = (int) 42,69f;

W przypadku C utraciłaby precyzję podczas tego rzutu. Rzucanie w Swift jest nieco bardziej złożone, aczkolwiek potężne. Używamy tego, co jest określane jako operator sprawdzania typu (jest i jako), aby sprawdzić, czy dana zmienna należy do określonego typu. Oto przykład:

var appleCount = 0

var orangeCount = 0

for item in Shop {

if item is Apple {

appleCount += 1

} else if item is Orange {

orangeCount += 1

}

}

W tym przykładzie, jeśli warunek jest spełniony (element jest jabłkiem lub elementem pomarańczowym), operator zwraca wartość true, ocena zakończy się pomyślnie, a kod będzie kontynuowany. Zwróć uwagę, że może to być bardziej złożone, jeśli należy wziąć pod uwagę wiele typów. Operator jest bardzo podobny. Rozważ następujący fragment kodu:

if let _ = item as? Apple ...

if item is Apple ...

Oba są bardzo podobne, ale w tym przypadku operator as przekształci wynikową klasę w celu jej późniejszego użycia (co często ma miejsce). Jeśli chcesz odpowiedzieć na to pytanie, zapytaj kandydata o różnicę między używaniem jako? i jako!

Pytanie 14: Jaką metodę użyłbyś, aby znaleźć typ obiektu?

Tutaj znowu mamy różne opcje, w zależności od używanego języka. W Objective-C musielibyśmy wywołać następującą funkcję:

[name isKindOfClass:[NSString class]]

podczas gdy w Swift moglibyśmy postępować w następujący sposób:

if object is String

W Swift mamy kilka dodatkowych opcji, które również mogą nam pomóc.

type(of:)

isKind(of:)

isMember(of:)

conforms(to:)

Pytanie 15: Jaka jest różnica między #if i #ifdef?

#if działa jak zwykle, jeśli.

#if __IPHONE_OS_VERSION_MAX_ALLOWED >= 30200

if (UI_USER_INTERFACE_IDIOM() ==

UIUserInterfaceIdiomPad) {

return YES;

}

#endif

return NO;

#ifdef oznacza „jeśli zdefiniowano - jakąś wartość lub makra”.

#ifdef RKL_APPEND_TO_ICU_FUNCTIONS

#define RKL_ICU_FUNCTION_APPEND(x) _RKL_CONCAT(x, RKL_APPEND_

TO_ICU_FUNCTIONS)

#else // RKL_APPEND_TO_ICU_FUNCTIONS

#define RKL_ICU_FUNCTION_APPEND(x) x

#endif // RKL_APPEND_TO_ICU_FUNCTIONS

Pytanie 16: Co to są kompilatory iOS?

Kompilatory to obszerny temat i żaden młodszy programista nie może ich szczegółowo wyjaśnić. Jeśli jednak pracujesz z systemem iOS, powinieneś mieć świadomość, że istnieją różne kompilatory, których można używać w systemie iOS i że przeszły one przez różne okresy dojrzałości. Są to GCC, LLVM z Clang i LLVM z GCC. Ta rozmowa może być przedłużana w nieskończoność. Na pytanie o aktualny stan wiedzy, kompetentny kandydat powie Ci, że w skrócie nie ma potrzeby korzystania z GCC, a zamiast tego LLVM z Clang jest kompilatorem, który zaspokoi wszystkie Twoje potrzeby. Można również podać szczegóły: LLVM kompiluje się szybciej niż GCC; wygenerowany kod będzie generalnie szybszy i zapewnia również dokładniejsze komunikaty o błędach podczas kompilacji.

Pytanie 17: Jak możesz zachować różne smaki dla wersji produkcyjnych i deweloperskich?

Można tu zastosować różne podejścia. Pierwszym krokiem mogłoby być stworzenie różnych celów, z których każdy wykorzystywałby inne informacje. pliki plist. Za każdym razem, gdy cel zostanie wybrany, zostanie użyty inny Info.plist, co umożliwi rozróżnienie między różnymi zmiennymi (na przykład tokenami, adresami URL itp.). Można również pomyśleć o użyciu identyfikatorów pakietów. Zdefiniowanie różnych makr preprocesora będzie sterować warunkową kompilacją różnych fragmentów kodu. Alternatywnie lub dodatkowo możesz umieścić ustawienia konfiguracji kompilacji (w tym zmianę lokalizacji pliku Info.plist) w plikach * .xcconfig i odwołać się do tych w obszarach projektu, informacji i konfiguracji. Następnie możesz stworzyć inną wersję swojej aplikacji, po prostu zmieniając schemat. Umieszczanie ustawień konfiguracji kompilacji w plikach to również ogromna korzyść dla kontroli konfiguracji.

Pytanie 18: Jaka jest różnica między viewDidLoad i viewDidAppear? Którego użyjesz do załadowania danych ze zdalnego serwera i wyświetlenia ich na ekranie?

viewDidLoad () jest wywoływana tylko raz, kiedy kontroler widoku został początkowo załadowany do pamięci. Generalnie, gdy chcemy utworzyć instancje zmiennych i zbudować widoki, które będą żywe przez cały cykl życia kontrolera widoku, robimy to tutaj. Jednak widok nadal nie jest widoczny! viewDidAppear () jest wywoływana, gdy widok jest wyświetlany po raz pierwszy na ekranie. W rzeczywistości ta metoda może być wywoływana kilka razy podczas cyklu życia kontrolera widoku. Pomyślmy na przykład, kiedy modalny kontroler widoku jest ładowany, a później odrzucany. Widok został już załadowany, ale w rezultacie pojawi się dwukrotnie. W tej metodzie zazwyczaj wykonujemy czynności związane z układem, takie jak rysowanie w interfejsie użytkownika, prezentowanie ekranów modalnych itp. W tym miejscu chcesz utworzyć instancję dowolnych zmiennych instancji i zbudować dowolne widoki, które istnieją przez cały cykl życia tego kontrolera widoku. Jednak widok zwykle nie jest w tym momencie widoczny. Aby odpowiedzieć na pytanie, chcesz załadować dane ze zdalnego serwera w metodzie viewDidLoad. W ten sposób jest ładowany raz, a nie za każdym razem, gdy pojawia się widok. Aby skontaktować się z kandydatem, możesz zapytać go / ją, jaki jest efekt zachowania rzeczy w widoku Widoczny. W rezultacie wystąpią bardzo prawdopodobne wycieki pamięci, jeśli elementy nie zostaną zwolnione po zniknięciu widoku.

Pytanie 19: Jak śledzisz błędy? Jakie narzędzia preferujesz?

W tym miejscu kandydat będzie chciał porozmawiać o narzędziach, które może już znać. Kilka z nich do zacytowania (bez domniemanego poparcia dla żadnej konkretnej firmy) to Apteligent (dawniej Crittercism), Crashlytics / Fabric, HockeyApp lub sama platforma Apple (po dystrybucji można zobaczyć awarie w iTunes Connect).

Pytanie 20: Co to jest NSUserDefaults?

NSUserDefaults to klasa, która umożliwia programiście zapisywanie ustawień, właściwości i informacji związanych z aplikacją lub danymi użytkownika. NSUserDefaults przechowuje klucze i skojarzoną z nimi wartość. W NSUserDefaults można przechowywać następujące typy:

- NSData
- NSString
- NSNumber
- NSDate
- NSArray

• NSDictionary

Możliwe pytania uzupełniające w tym miejscu mogą być następujące: Jakie są różnice w stosunku do innych mechanizmów przechowywania w systemie iOS? Czy można przechowywać obraz za pomocą NSUserDefaults? Odpowiedź brzmi tak. Obraz można przekształcić w Base64 i przechowywać jako NSString. Kolejne pytanie dotyczy tego, czy byłoby to efektywne wykorzystanie NSUserDefaults. Ogólnie rzecz biorąc, kandydat powinien zostać poproszony o omówienie zalet i wad korzystania z NSUserDefaults w przeciwieństwie do innych typów pamięci masowej.

Pytanie 21: Jak testujesz swój kod? Jak sprawić, by kod był testowalny?

Są to również ważne pytania, które mogą mieć wiele interesujących elementów i mamy nadzieję, że pozwolą nam poznać sposób myślenia kandydata i wiedzę teoretyczną. Odpowiednia dyskusja może dotyczyć korzyści płynących z testowania ręcznego i testowania automatycznego. Każdy rodzaj testu ma inne zalety i wady. Porozmawiajmy o testowaniu automatycznym.

• Testy mogą przebiegać dość szybko, ponieważ nie jest potrzebny człowiek.

• Mogą być opłacalne, jeśli odpowiednio wykorzystamy narzędzia do automatyzacji. Mogą być drogie w krótkim okresie, ale zdecydowanie niższe koszty w dłuższej perspektywie.

• Każdy może zobaczyć wyniki. Testy można odtwarzać, nagrywać i zapisywać.

• Z drugiej strony, narzędzia mogą mieć ograniczenia i możemy być ograniczeni do rozwoju naszego narzędzia testowego lub API.

Testowanie ręczne może mieć również swoje zalety i wady.

• W krótkim okresie obniża koszty.

• Bardziej prawdopodobne jest ujawnienie rzeczywistych problemów użytkowników. Ponieważ testy automatyczne są zautomatyzowane, nie będą testować specjalnych przypadków użycia krawędzi ani improwizować i testować czegoś, czego nie ma w skrypcie.

• Z drugiej strony, niektóre zadania mogą być raczej trudne do wykonania ręcznie, w najgorszym przypadku powtarzalne, nudne i wcale nie stymulujące. Kandydat może również zostać poproszony o pytanie, z jakich frameworków testowych korzystasz? XCTest jest ściśle powiązany z Xcode, dlatego większość programistów powinna mieć przynajmniej minimum teoretyczne wyobrażenie o frameworku. Inne frameworki testowe to Appium i Calabash. Aby zanurkować głębiej, możesz omówić zalety i wady każdego frameworka. Może to być wygodny sposób na pogodzenie wiedzy kandydata z wiedzą firmy, chociaż kandydata nigdy nie powinno się zatrudniać wyłącznie na podstawie znajomości posiadanych przez siebie ram, ale na podstawie jego ogólnej postawy i rozumowania. Ramy dzisiejszego dnia znajdą się w muzeach jutra. Na koniec interesująca dyskusja na ten temat dotyczy tego, jak sprawić, by kod był testowalny. Zasady tworzenia testowalnego kodu zawsze sprzyjają interesującej rozmowie, a wśród nich SOLID jest bardzo dobrym punktem wyjścia.

Pytanie 22: Jaka jest różnica między właściwościami atomowymi i nieatomowymi? Która jest wartość domyślna dla zsyntetyzowanych właściwości? Kiedy użyłbyś jednego kontra drugiego?

Atomic jest domyślnym zachowaniem. Niepodzielna właściwość zapewni, że obecny proces zostanie zakończony przez CPU, zanim inny proces uzyska dostęp do zmiennej. Oczywiście nie jest to szybkie, ponieważ zapewnia całkowite zakończenie procesu. Z drugiej strony zachowanie nieatomowe nie jest zachowaniem domyślnym. Jest szybszy dla kodu zsyntetyzowanego, to znaczy dla zmiennych utworzonych przy użyciu @property i @synthesize. Nie są bezpieczne dla wątków i ich używanie może

spowodować w nieoczekiwanym zachowaniu, gdy dwa różne procesy uzyskują dostęp do tej samej zmiennej w tym samym czasie.

Pytanie 23: Co to są „mocne" i „słabe" odniesienia? Dlaczego są ważne i jak można ich używać do kontrolowania zarządzania pamięcią i unikania wycieków pamięci?

Silne odniesienie to takie, które pojawia się domyślnie za każdym razem, gdy tworzona jest zmienna. Istnieje ważna właściwość z wzajemnymi silnymi odniesieniami, a kiedy tak się dzieje, następuje cykl utrzymania. W tej sytuacji ARC nigdy nie będzie w stanie zniszczyć obiektów. Jest to opisane jako wyciek pamięci. Zawsze należy unikać tego rodzaju wzajemnych i silnych odniesień. Gdy nie jest to możliwe, z pomocą może przyjść użycie słabych odniesień. Jeśli jedno z tych odniesień zostanie zadeklarowane jako słabe, cykl przechowywania zostanie przerwany, a zatem uniknie się wycieku pamięci.

Pytanie 24: Jaki jest twój proces śledzenia i naprawiania wycieku pamięci?

Najlepszym podejściem do tego pytania jest dostarczenie kandydatowi istniejącego projektu i znanego wycieku pamięci oraz poproszenie go o jego debugowanie.

Pytanie 25: Jakie sześć instrumentów jest częścią standardowego zestawu iOS?

Wśród nich kandydat powinien wymyślić nazwy przecieków, wielordzeniowy, profilowanie czasu, zombie, użycie systemu, rejestrator interfejsu użytkownika, monitor aktywności, alokacje, animacja rdzenia. Aby kontynuować, funkcjonalność i cechy szczególne każdego narzędzia to dobre tematy do zagłębienia.

Pytanie 26: Jak dodać zasoby do mojej aplikacji?

W grupach zasobów po prostu przeciągnij do niego plik i wybierz „Utwórz odniesienia do folderów dla wszystkich dodanych folderów". W ten sposób plik zostanie dodany automatycznie.

Pytanie 27: Co to są bloki?

Bloki to funkcja na poziomie języka dodana do C, Objective-C i C ++, która umożliwia tworzenie odrębnych segmentów kodu, które można przekazywać do metod lub funkcji tak, jakby były wartościami. Bloki są obiektami Objective-C, co oznacza, że można je dodawać do takich kolekcji jak NSArray czy NSDictionary. Mają także możliwość przechwytywania wartości z otaczającego zakresu, upodabniając je do domknięć lub lambd w innych językach programowania. Kandydat, który chciałby dokładniej zgłębić ten temat, może również powiedzieć, że t odpowiednikiem bloków w Swift są domknięcia. Zamknięcia są obiektami pierwszej klasy, więc można je zagnieżdżać i przekazywać (jak bloki w Objective-C). W Swift funkcje są tylko specjalnym przypadkiem domknięć.

Pytanie 28: Jak wstawić kontrolę poprawności, która zostanie wyłączona w kompilacjach wydań?

Można użyć kodu potwierdzenia, który działa tylko w kompilacji do debugowania. Poniższy fragment kodu jest przykładem:

```
#if defined(NDEBUG)

{

// The assertion code below should be compiled out of

existence in a release

// build. Log an error and abort the program if it is not.
```

```
bool ok = true;

NSCAssert(ok = false, @"NS assertions should be disabled

but are not");

if (!ok)

{

NSLog(@"Detected release build but NS_BLOCK_ASSERTIONS

is not defined");

return -1;

}

}

#endif
```

Pytanie 29: Kiedy let i var są odpowiednie w Swift?

Słowo kluczowe let definiuje stałą.

```
let theAnswer = 42
```

theAnswer nie może być później zmienione. Dlatego niczego opcjonalnego lub słabego nie można zapisać za pomocą let. Pierwsza z nich musi się zmienić w trakcie działania i musi być napisana przy użyciu var.

var definiuje zwykłą zmienną.

```
var president = "Lincoln"
```

Pytanie 30: Co to jest protokół, jak definiujesz swój własny i kiedy jest używany?

Protokoły to sposób na określenie zestawu metod, które klasa ma zaimplementować, jeśli chce współpracować z jedną z klas. Delegaci i źródła danych, takie jak UITableViewDelegate i UITableViewDataSource, są rzeczywiście protokołami.

```
@protocol MyProtocol <NSObject>

- (void)aRequiredMethod;

@required

- (void)anotherRequiredMethod;

@optional

- (void)anOptionalMethod;

@end
```

@required specifies that the method must be implemented, whereas

@optional lets the developer decide.

Następnie można określić, że klasa „jest zgodna” z protokołem (implementuje wymagane metody) w interfejsie klasy w następujący sposób:

@interface MyClass <MyProtocol>

@end

Pytanie 31: Co to jest MVC, jak jest zaimplementowany w iOS i czy ma jakieś alternatywy?

MVC to dobrze znany wzorzec projektowy, który definiuje, w jaki sposób każdy komponent musi zostać zaimplementowany, jak komponenty komunikują się między sobą i gdzie musi znajdować się każda funkcja. Domyślnie firma Apple zapewnia implementację wzorca MVC: widoki UIV to widoki, w których znajduje się interfejs użytkownika; UIViewControllers obsługują kontroler, który nasłuchuje zdarzeń i dzięki temu może aktualizować widok zgodnie z naszymi potrzebami; a model to dane, których używa aplikacja i które mogą znajdować się w dowolnym obiekcie, który tworzymy do przechowywania przydatnych informacji. Istnieje wiele alternatyw dla MVC. Powszechnym może być używanie MVVM z reaktywnym kakao. Inne alternatywy mogą obejmować Viper i używanie funkcjonalnego kodu reaktywnego. Dobrze zorientowany kandydat powie Ci, jaka jest różnica między wzorem a architekturą. Ogólnie rzecz biorąc, wzorzec jest modelem służącym do rozwiązywania powtarzającego się problemu w dziedzinie informatyki, podczas gdy architektura to zestaw reguł i organizacji, które definiują strukturę kodu, aby można go było konserwować i rozszerzać.

Pytanie 32: Jakie są różne sposoby określenia układu elementów w UIView?

Istnieje kilka różnych technik określania układu w aplikacji na iOS.

• Możemy użyć InterfaceBuilder, aby dodać plik XIB do naszego projektu. Później ten plik XIB można załadować z poziomu naszego kodu aplikacji. InterfaceBuilder pozwala nam również stworzyć storyboard.

• Możesz napisać własny kod, aby używać NSLayoutConstraints, aby elementy w widoku były uporządkowane według automatycznego układu.

• Możesz stworzyć CGRects opisujące dokładne współrzędne dla każdego elementu i przekazać je do metody ramki UIView - (id) initWithFrame: (CGRect).

Pytanie 33: Jaki kod formatu jest używany do drukowania sformatowanej wiadomości za pomocą NSString?

Użyłbyś następujących:

NSLog (@ "Message ==% @", msg);