

# 資料編

## 目次

図 1	教材の趣旨	..... P 2
図 2	教材の目次	..... P 3
図 3	1. ソースプログラムの基本記号	..... P 4
図 4	2. 基本プログラム作成	..... P 5
図 5	3. 動作確認	..... P 6
図 6	4. 必要用語解説	..... P 7
図 7	Xcode Ver3.2 と Xcode Ver4 の表記の違い	..... P 8
図 8	2つの画面がボタンを押すことによってループするアプリケーション	..... P 9
図 9	Storyboard の解説	..... P10
図 10	ワークスペースの名称設定	..... P11
図 11	BMI 計算機	..... P12
図 12	Storyboard におけるプログラムソースを組み込む方法	..... P13
図 13	BMI 計算機プログラムソース画面	..... P14
図 14	プログラム解説	..... P15
図 15	応用に使う予定だった電卓のプログラムソース	..... P16
図 16	電卓のシミュレータ画面	..... P17
図 17	改善後の BMI 計算機プログラムソース画面	..... P18
図 18	応用問題の作成	..... P19
図 19	応用問題作成後の画面推移見込み	..... P20
図 20	四角形面積計算機	..... P21
図 21	四角形面積計算機プログラムソース画面	..... P22
図 22	三角形面積計算機	..... P23
図 23	三角形面積計算機プログラムソース画面	..... P24
図 24	数当てゲーム	..... P25
図 25	数当てゲームプログラムソース画面	..... P26,27

図1 教材の趣旨

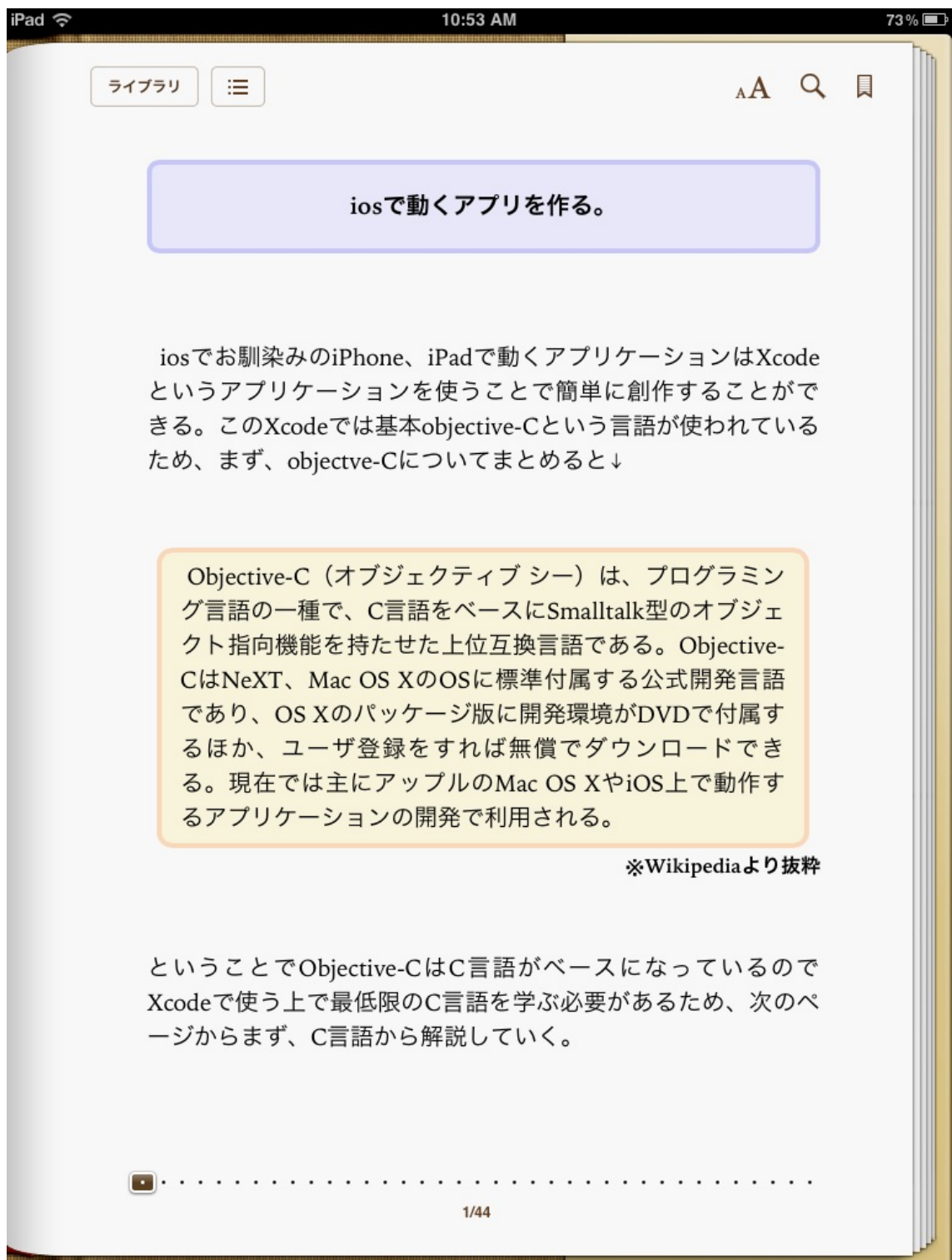


図 2 教材の目次

iosで動くアプリを作る。	1
C言語入門	2
Xcode入門1 ～ファイルの作成～	9
Xcode入門2 ～Story board～	12
Xcode入門3 ～BMI計算機～	20
Xcodeプログラム解説	34
Xcode応用編 ～面積計算機～	39
Xcode応用編 ～面積計算機2～	49
Xcode応用編 ～数当てゲーム～	52

図3 1. ソースプログラムの基本記号

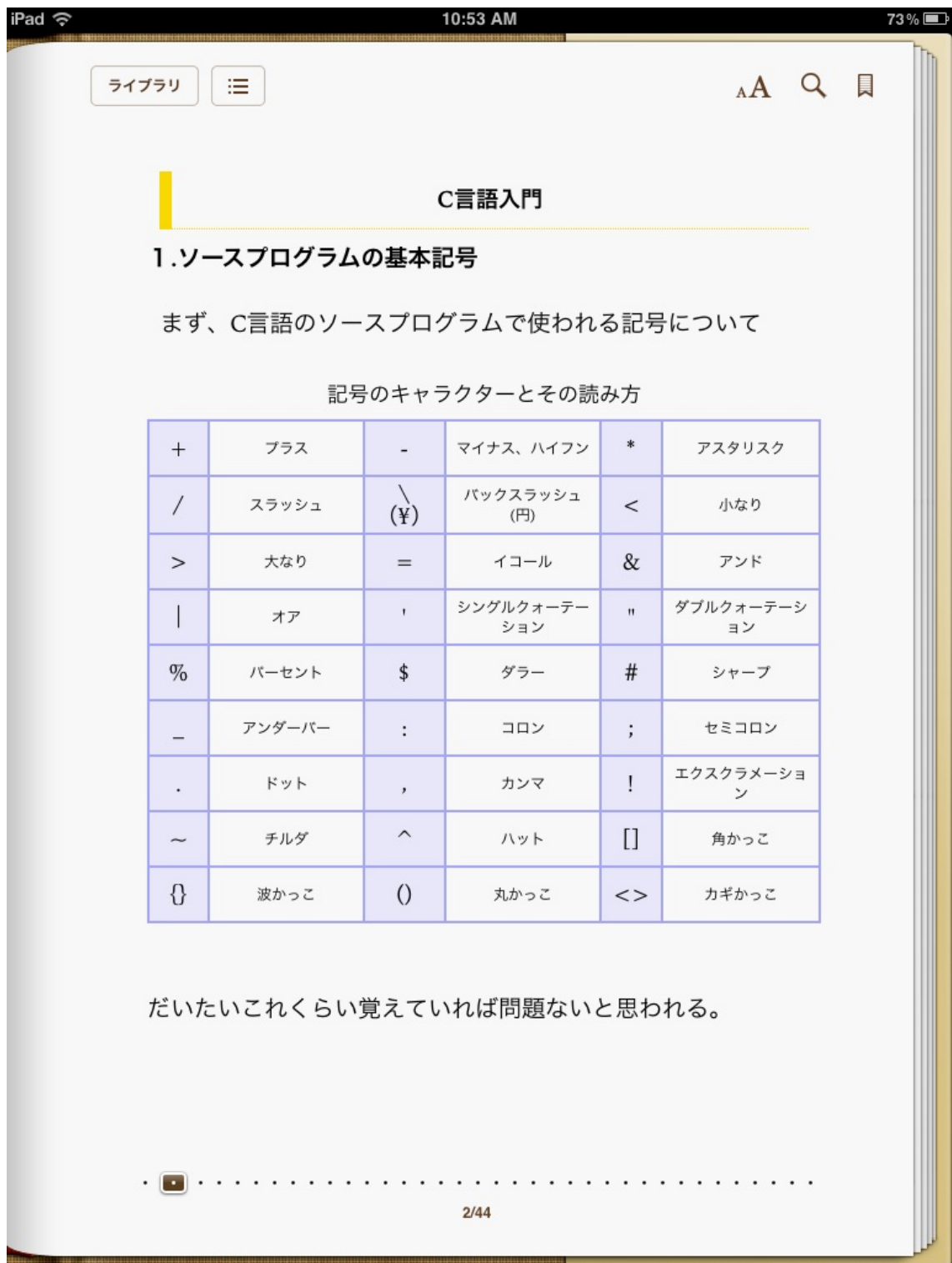


図4 2. 基本プログラム作成

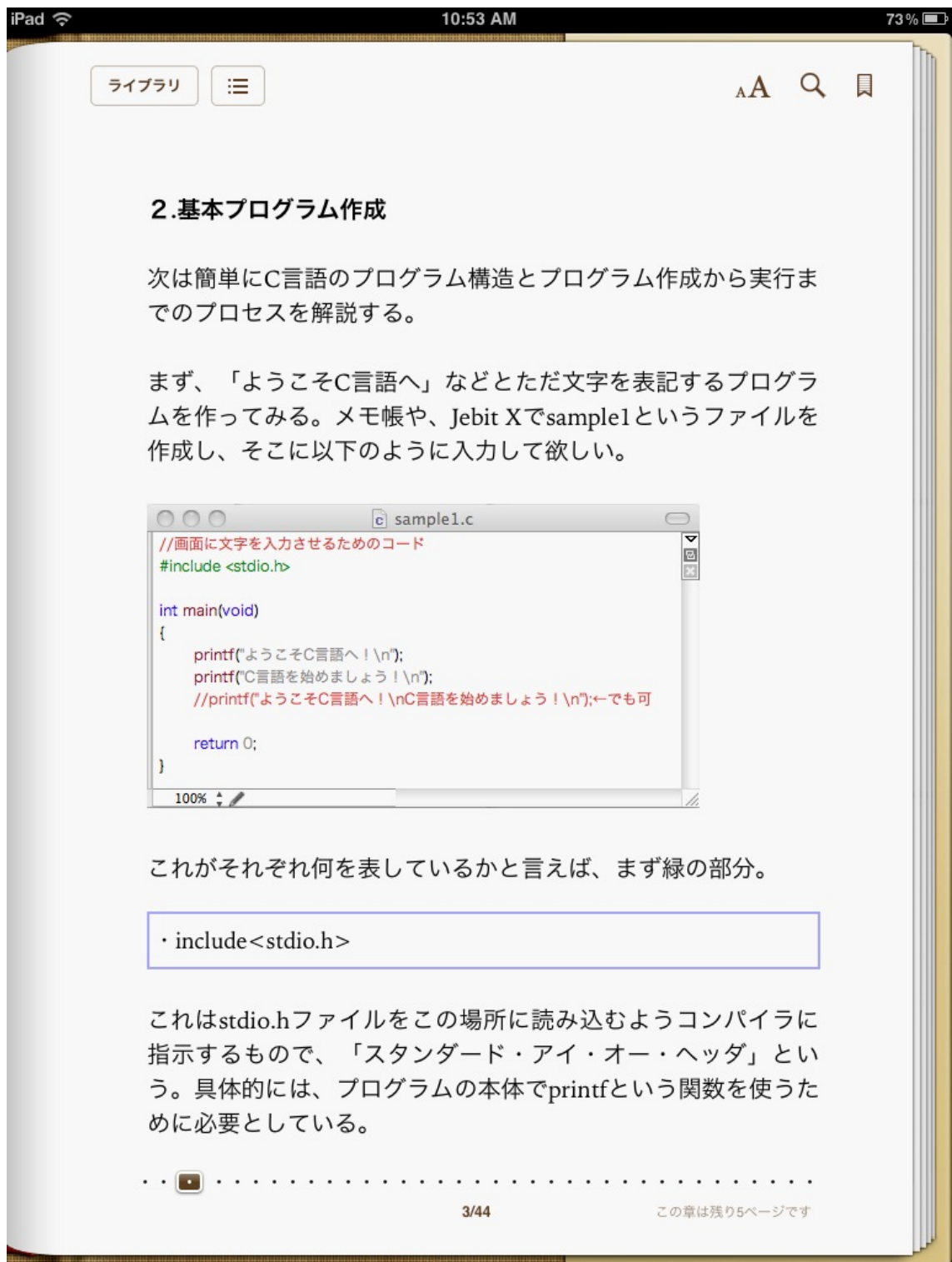


図5 3. 動作確認

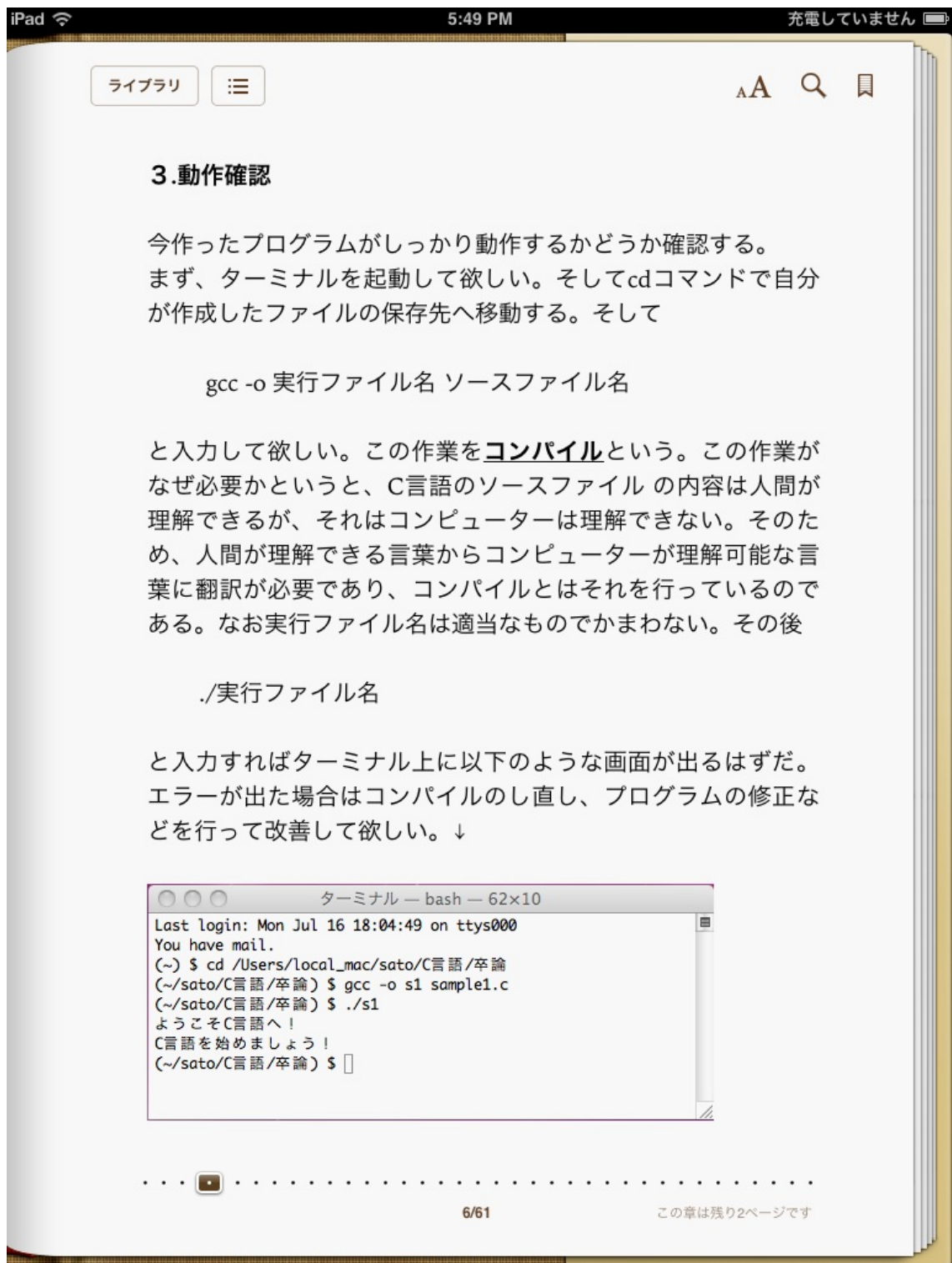


図 6 4. 必要用語解説

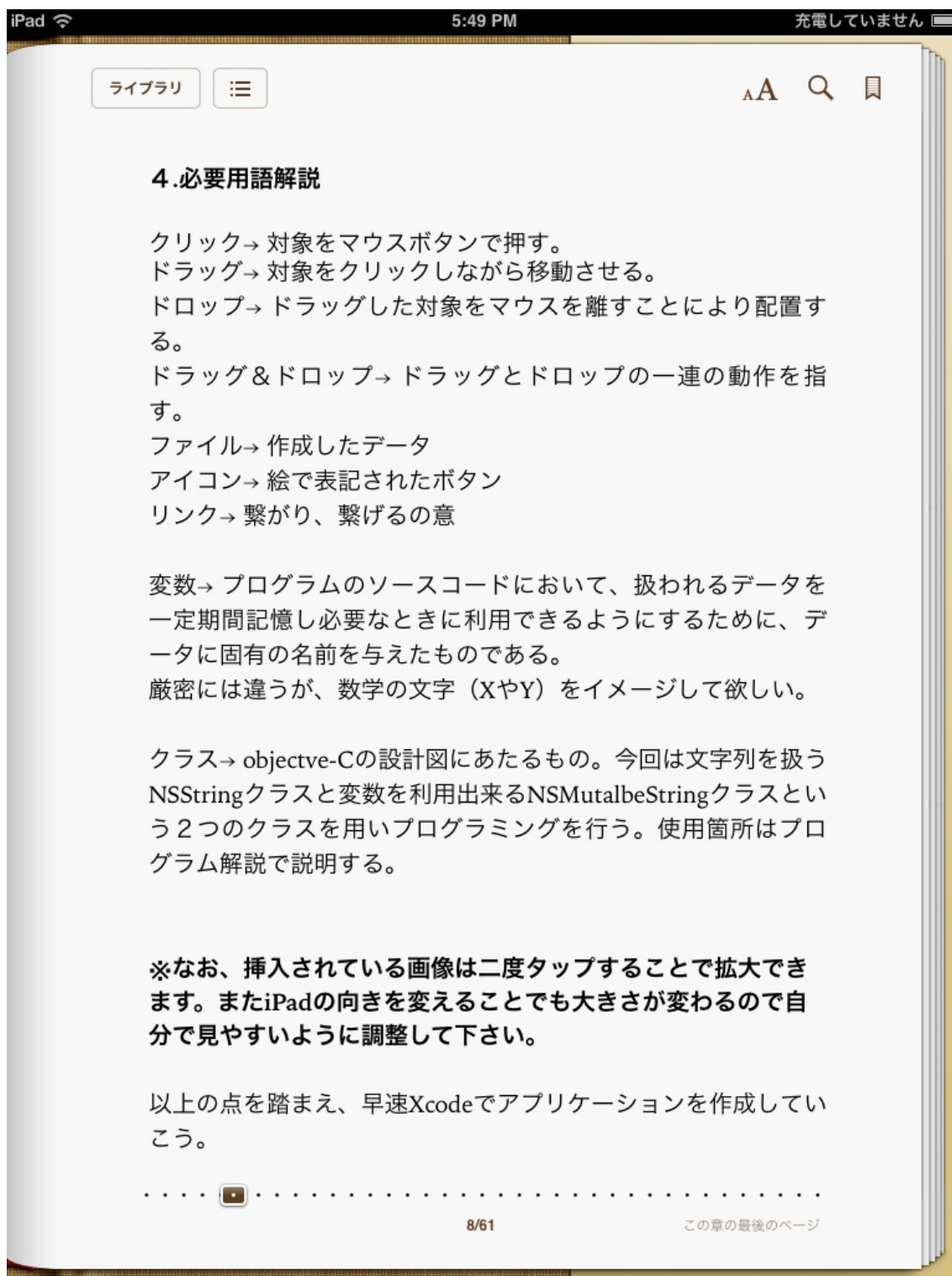


図7 Xcode Ver3.2 と XcodeVer4 の表記の違い



上が Ver3 シリーズ。

下が Ver 4 シリーズ。

表記が日本語から英語に変わっている。



図 8 2つの画面がボタンを押すことによってループするアプリケーション

画面 A

画面 B



画面 A と B がボタンを押すことでループする。

図9 Story board の解説

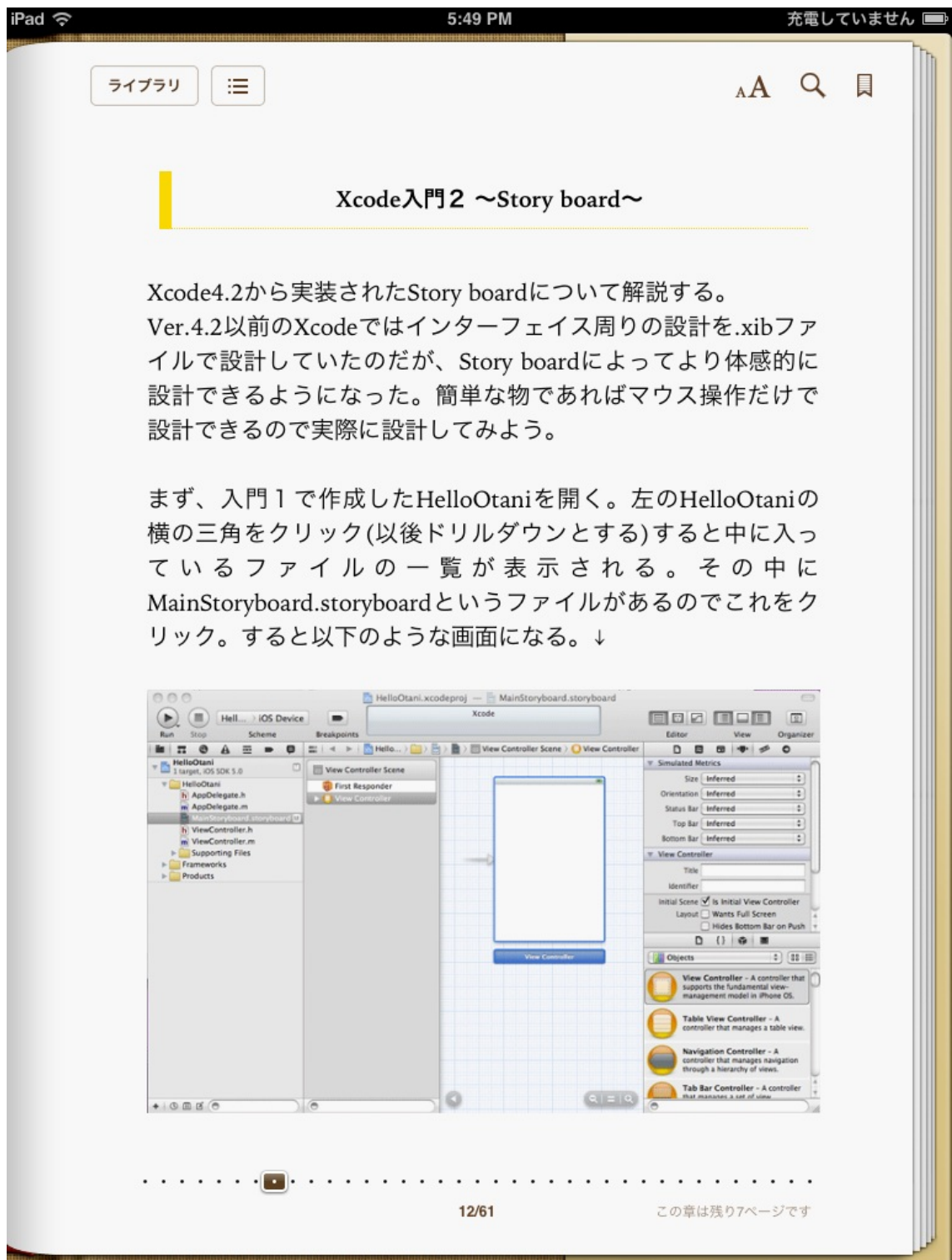
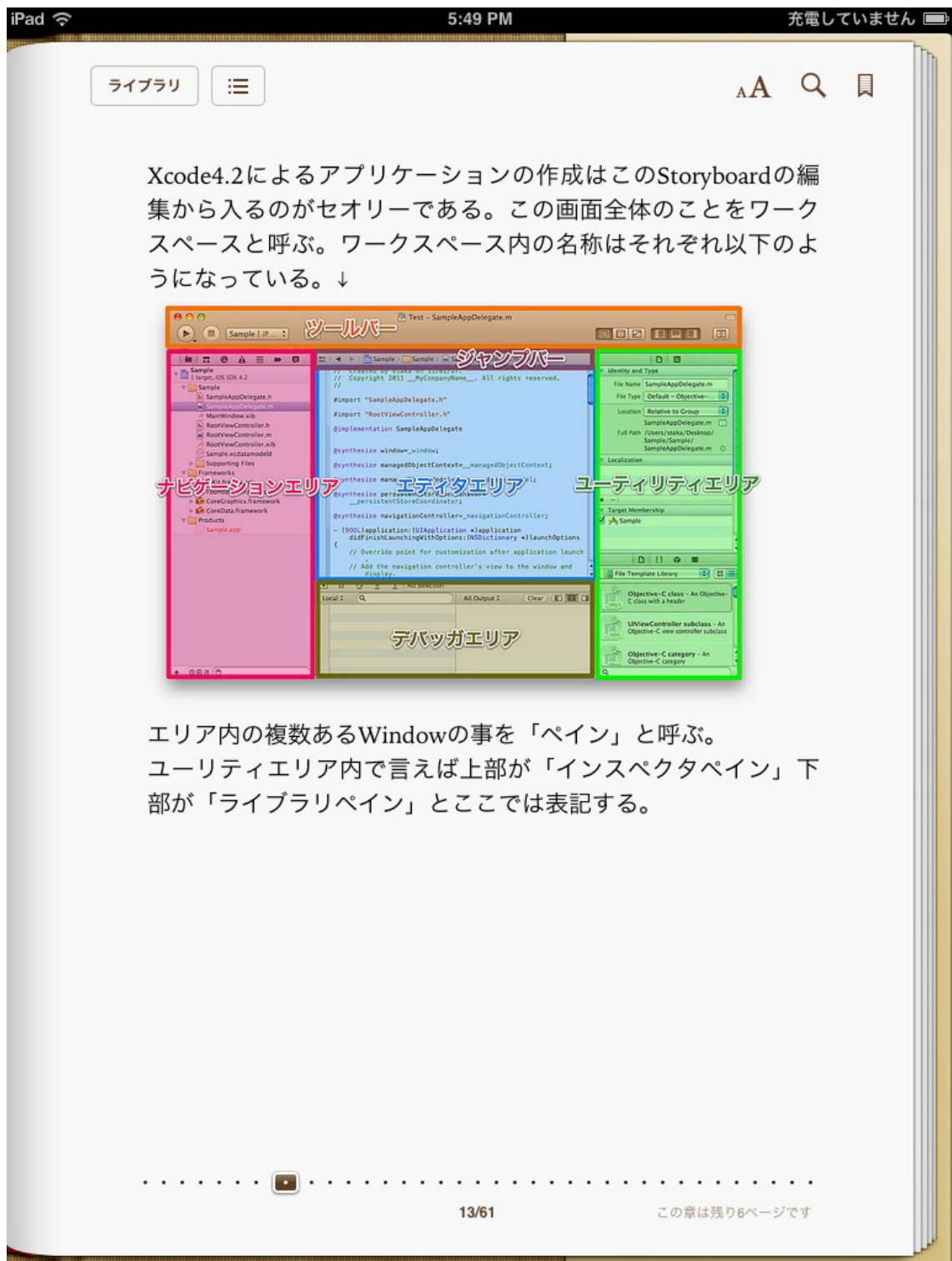


図 1 0 ワークスペースの名称設定



Xcode4.2によるアプリケーションの作成はこのStoryboardの編集から入るのがセオリーである。この画面全体のことをワークスペースと呼ぶ。ワークスペース内の名称はそれぞれ以下のようになっている。↓

エリア内の複数あるWindowの事を「ペイン」と呼ぶ。ユーティリティエリア内では上部が「インスペクタペイン」、下部が「ライブラリペイン」とここでは表記する。

図 1 1 BMI 計算機



図 1 2 Story board におけるプログラムソースを組み込む方法

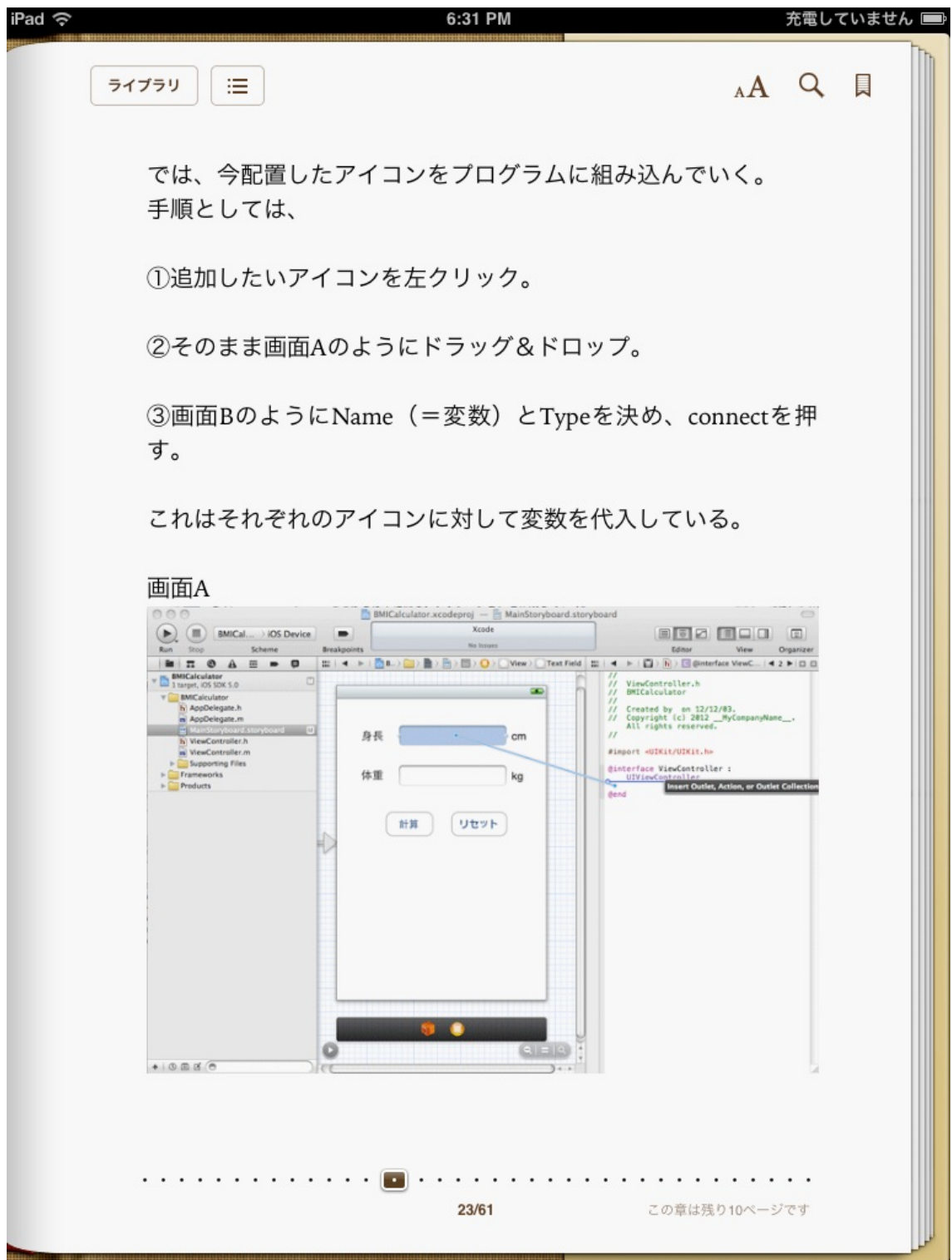


図 1 3 プログラムソース画面

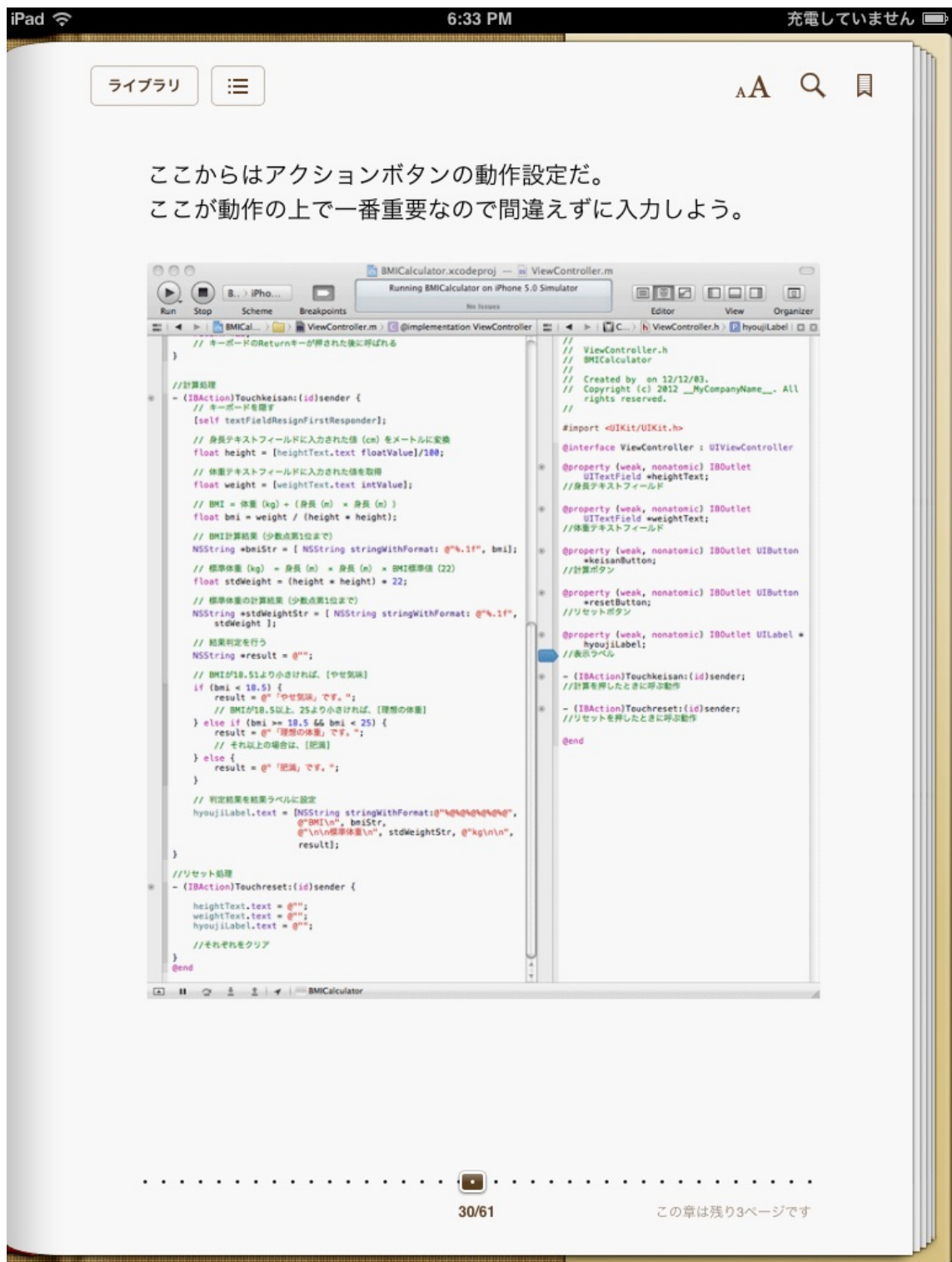


図 1 4 プログラム解説

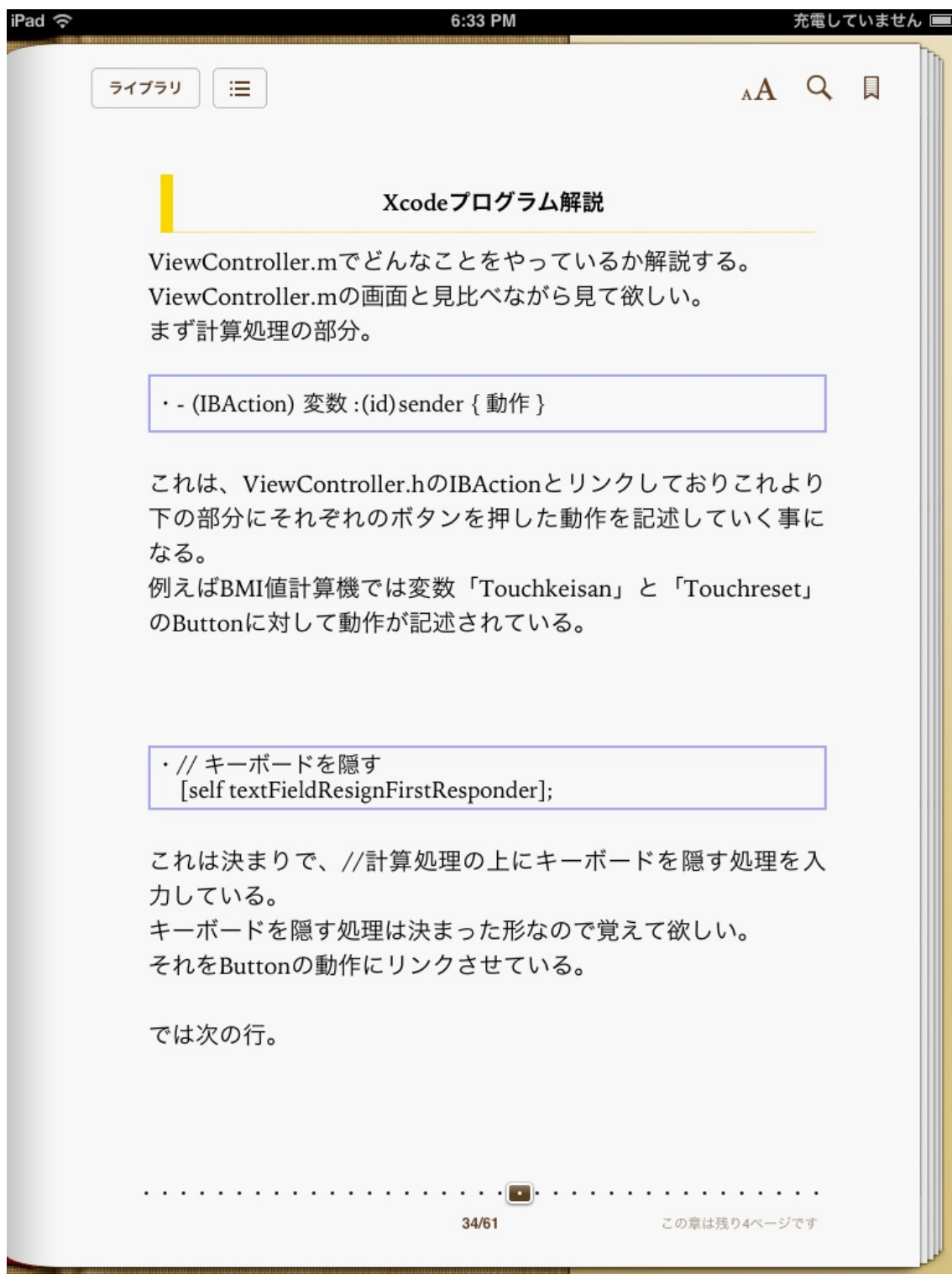


図 1 5 応用に使う予定だった電卓のプログラムソース

```

//
//  dentakucore.m
//  dentaku
//
//  Created by on 12/10/22.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import "dentakucore.h"

@interface dentakucore()
@property (nonatomic, strong) NSMutableArray *operandStack;
@end

@implementation dentakucore

@synthesize operandStack = _operandStack;

-(NSMutableArray *)operandStack
{
    if (_operandStack == nil) _operandStack = [[NSMutableArray alloc] init];
    return _operandStack;
}

-(void)setOperandStack:(NSMutableArray *)operandStack
{
    _operandStack = operandStack;
}

-(double)popOperand
{
    NSNumber *operandObject = [self.operandStack lastObject];
    if (operandObject) [self.operandStack removeObject];
    return [operandObject doubleValue];
}

-(void)pushOperand:(double)operand
{
    [self.operandStack addObject:[NSNumber numberWithInt:operand]];
}

-(double)performOperation:(NSString *)operation
{
    double result = 0;

    if ([operation isEqualToString:@"+"]){
        result = [self popOperand] + [self popOperand];
    } else if ([@"*" isEqualToString:operation]){
        result = [self popOperand] * [self popOperand];
    } else if ([operation isEqualToString:@"-"]){
        double subtrahend = [self popOperand];
        result = [self popOperand] - subtrahend;
    } else if ([operation isEqualToString:@"/"]){
        double divisor = [self popOperand];
        if (divisor) result = [self popOperand] / divisor;
    } else if ([operation isEqualToString:@"AC"]){
        result = 0;
    }

    [self pushOperand:result];
    return result;
}

@end

```

```

//
//  dentakuViewController.m
//  dentaku
//
//  Created by on 12/10/15.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import "dentakuViewController.h"
#import "dentakucore.h"

@interface dentakuViewController()
@property (nonatomic) BOOL
    userIsInTheMiddleOfEnteringANumber;
@property (nonatomic, strong) dentakucore *core;
@end

@implementation dentakuViewController

@synthesize gamen = _gamen;
@synthesize userIsInTheMiddleOfEnteringANumber =
    _userIsInTheMiddleOfEnteringANumber;
@synthesize core = _core;

-(dentakucore *)core
{
    if (!_core) _core = [[dentakucore alloc] init];
    return _core;
}

-(IBAction)botan:(UIButton *)sender
{
    NSString *digit = sender.currentTitle;
    if (self.userIsInTheMiddleOfEnteringANumber){
        self.gamen.text = [self.gamen.text
            stringByAppendingString:digit];
    } else {
        self.gamen.text = digit;
        self.userIsInTheMiddleOfEnteringANumber = YES;
    }
}

-(IBAction)enterPress
{
    [self.core pushOperand:[self.gamen.text doubleValue]];
    self.userIsInTheMiddleOfEnteringANumber = NO;
}

-(IBAction)operationPressed:(UIButton *)sender
{
    if (self.userIsInTheMiddleOfEnteringANumber){
        [self enterPress];
    }

    NSString *operation = [sender currentTitle];
    double result = [self.core performOperation:operation];
    self.gamen.text = [NSString stringWithFormat:@"%g",
        result];
}

@end

```



図 1 6 電卓のシミュレータ画面



図 1 7 改善後の BMI 計算機プログラムソース画面

```

//ここから書き足し
- (void)textFieldResignFirstResponder {
    [heightText resignFirstResponder];
    [weightText resignFirstResponder];
    // 身長・体重テキストのキーボードを隠す処理
}

//計算処理
- (IBAction)Touchkeisan:(id)sender {
    // キーボードを隠す
    [self textFieldResignFirstResponder];

    // 身長テキストフィールドに入力された値 (cm) をメートルに変換
    float height = [heightText.text floatValue]/100;

    // 体重テキストフィールドに入力された値を取得
    float weight = [weightText.text floatValue];

    // BMI = 体重 (kg) ÷ (身長 (m) × 身長 (m))
    float bmi = weight / (height * height);

    // BMI計算結果 (実数を文字列に変換)
    NSString *bmiStr = [NSString stringWithFormat:@"%g", bmi];

    // 標準体重 (kg) = 身長 (m) × 身長 (m) × BMI標準値 (22)
    float stdWeight = (height * height) * 22;

    // 標準体重の計算結果 (実数を文字列に変換)
    NSString *stdWeightStr = [NSString stringWithFormat:@"%g", stdWeight];

    // 結果判定を行う
    NSString *result = @"";

    // BMIが18.5より小さければ、「やせ気味」と表示する。
    if (bmi < 18.5) {
        result = @"「やせ気味」です。";
        // BMIが18.5以上、25より小さければ、「理想の体重」と表示する。
    } else if (bmi >= 18.5 && bmi < 25) {
        result = @"「理想の体重」です。";
        // それ以上の場合は、「肥満」と表示する。
    } else {
        result = @"「肥満」です。";
    }

    // 判定結果を結果ラベルに設定
    hyoujiLabel.text = [NSString stringWithFormat:@"%g%%@%g%%@%g%%@%g%%@%",
        @"BMI値は ", bmiStr, @"です。",
        @"\n\n標準体重は ", stdWeightStr, @"kgなので\n\n",
        result];
}

//リセット処理
- (IBAction)Touchreset:(id)sender {
    // キーボードを隠す
    [self textFieldResignFirstResponder];

    heightText.text = @"";
    weightText.text = @"";
    hyoujiLabel.text = @"";

    //それぞれをクリア
}
@end

```

```

//
//  ViewController.h
//  BMI Calculator
//
// Created by on 12/12/03.
// Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface ViewController : UIViewController

@property (weak, nonatomic) IBOutlet UITextField *heightText;
//身長テキストフィールド

@property (weak, nonatomic) IBOutlet UITextField *weightText;
//体重テキストフィールド

@property (weak, nonatomic) IBOutlet UIButton *keisanButton;
//計算ボタン

@property (weak, nonatomic) IBOutlet UIButton *resetButton;
//リセットボタン

@property (weak, nonatomic) IBOutlet UILabel *hyoujiLabel;
//表示ラベル

- (IBAction)Touchkeisan:(id)sender;
//計算を押したときに呼ぶ動作

- (IBAction)Touchreset:(id)sender;
//リセットを押したときに呼ぶ動作

@end

```

図 1 8 応用問題の作成

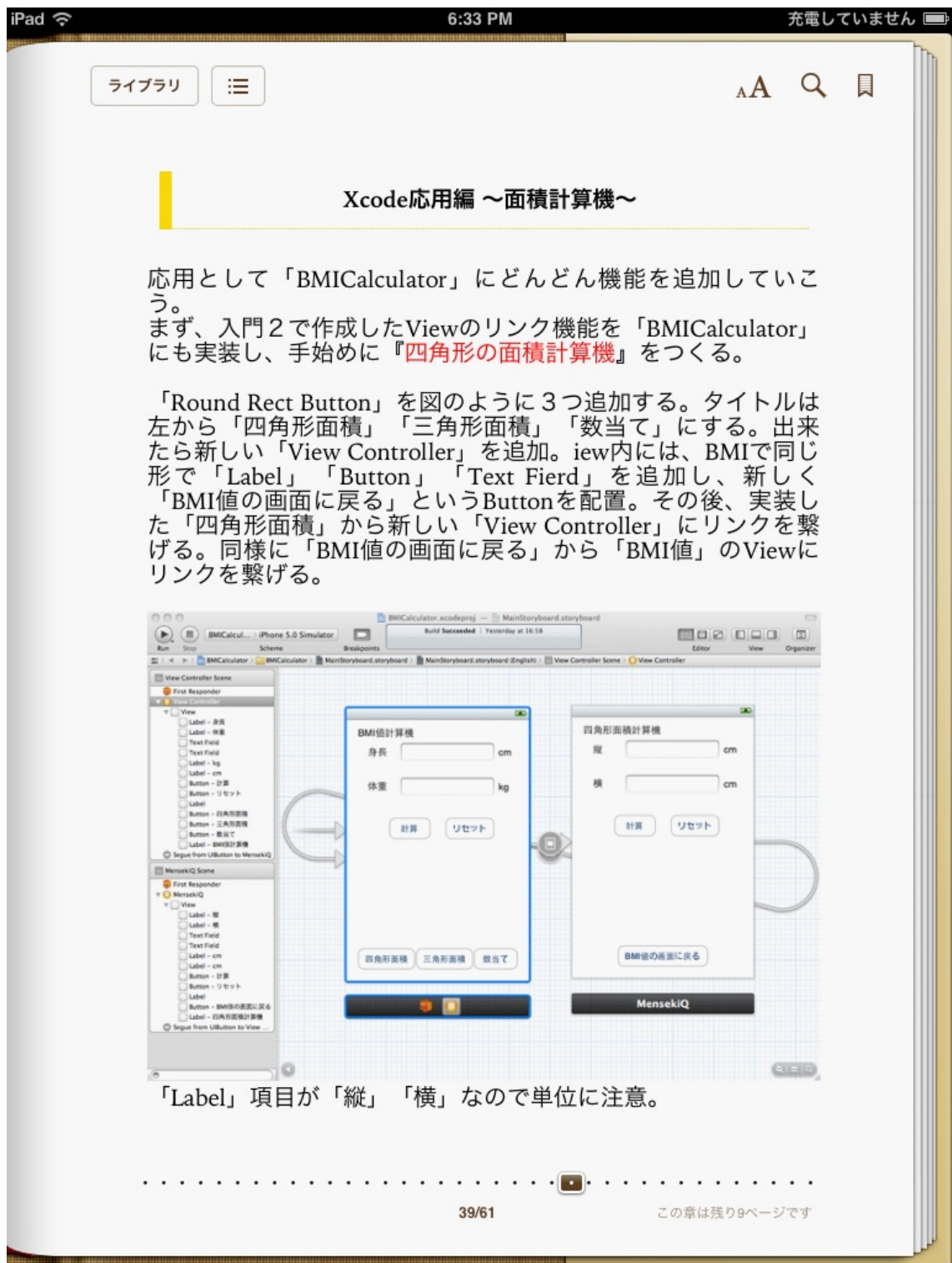


図 1 9 応用問題作成後の画面推移見込み



画面 A の 3 つのボタンを押すことで対応した画面 B、画面 C、画面 D の 3 種類の画面に変わる。

図 2 0 四角形面積計算機



図 2 1 四角形面積計算機プログラムソース画面

```

//
//  menseki_S_Q
//  BMI Calculator
//
//  Created by  on 12/12/15.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface menseki_Q : UIViewController

@property (weak, nonatomic) IBOutlet UITextField *tateText;
@property (weak, nonatomic) IBOutlet UITextField *yokoText;
@property (weak, nonatomic) IBOutlet UIButton *keisan_Q;
@property (weak, nonatomic) IBOutlet UIButton *reset_Q;
@property (weak, nonatomic) IBOutlet UILabel *hyouji_Q;

- (IBAction)Touchkeisan_Q:(id)sender;
- (IBAction)Touchreset_Q:(id)sender;

@end

@implementation menseki_Q

- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)
interfaceOrientation
{
    // Return YES for supported orientations
    return (interfaceOrientation == UIInterfaceOrientationPortrait);
}

//ここから書き足し
- (void)textFieldResignFirstResponder {
    [tateText resignFirstResponder];
    [yokoText resignFirstResponder];
    // 縦・横テキストのキーボードを隠す処理
}

- (IBAction)Touchkeisan_Q:(id)sender {
    // キーボードを隠す
    [self textFieldResignFirstResponder];

    // 縦テキストフィールドに入力された値 (cm) を実数と文字列に変換
    float tate = [tateText.text floatValue];
    NSString *tateStr = [NSString stringWithFormat:@"%g", tate];

    // 横テキストフィールドに入力された値を実数と文字列に変換
    float yoko = [yokoText.text floatValue];
    NSString *yokoStr = [NSString stringWithFormat:@"%g", yoko];

    // 四角形の面積 = 縦 (cm) * 横 (cm)
    float menseki_Q = tate * yoko;

    // 四角形の面積計算結果 (実数を文字列に変換)
    NSString *menseki_QStr = [NSString stringWithFormat:@"%g", menseki_Q];

    // 判定結果を結果ラベルに設定
    hyouji_Q.text = [NSString stringWithFormat:@"%g×%g×%gで、",
        tateStr, tateStr, yokoStr,
        "\n\n", yokoStr, yokoStr,
        "\n\n四角形の面積は", menseki_QStr, "cm2 です。"];
}

- (IBAction)Touchreset_Q:(id)sender {
    // キーボードを隠す
    [self textFieldResignFirstResponder];

    tateText.text = @"";
    yokoText.text = @"";
    hyouji_Q.text = @"";

    //それぞれをクリア
}

@end
    
```

図 2 2 三角形面積計算機



図 2 3 三角形面積計算機プログラムソース画面

```

// menseki_T.m
[self setTeihenText:nil];
[self setTakasaText:nil];
[self setKeisan_T:nil];
[self setReset_T:nil];
[self setHyouji_T:nil];
[super viewDidLoad];
// Release any retained subviews of the main view.
// e.g. self.myOutlet = nil;
}

- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)
interfaceOrientation
{
    // Return YES for supported orientations
    return (interfaceOrientation == UIInterfaceOrientationPortrait);
}

//ここから書き足し
- (void)textFieldResignFirstResponder {
    [teihenText resignFirstResponder];
    [takasaText resignFirstResponder];
    // 底辺・高さテキストのキーボードを隠す処理
}

- (IBAction)Touchkeisan_T:(id)sender {
    // キーボードを隠す
    [self textFieldResignFirstResponder];

    // 底辺テキストフィールドに入力された値 (cm) を実数と文字列に変換
    float teihen = [teihenText.text floatValue];
    NSString *teihenStr = [NSString stringWithFormat:@"%g", teihen];

    // 高さテキストフィールドに入力された値を実数と文字列に変換
    float takasa = [takasaText.text floatValue];
    NSString *takasaStr = [NSString stringWithFormat:@"%g", takasa];

    // 三角形の面積 = 底辺 (cm) *高さ (cm) / 2
    float menseki_T = teihen * takasa / 2;

    // 三角形の面積計算結果 (実数を文字列に変換)
    NSString *menseki_TStr = [NSString stringWithFormat:@"%g", menseki_T];

    // 判定結果を結果ラベルに設定
    hyouji_T.text = [NSString stringWithFormat:@"%g%g%g%g%g%g%g%g%g%g",
        @"底辺", teihenStr, @"cmで",
        @"\n\n高さ", takasaStr, @"cmの",
        @"\n\n三角形の面積は", menseki_TStr, @"cm2 です。"];
}

- (IBAction)Touchreset_T:(id)sender {
    // キーボードを隠す
    [self textFieldResignFirstResponder];

    teihenText.text = @"";
    takasaText.text = @"";
    hyouji_T.text = @"";

    //それぞれをクリア
}

@end
// menseki_T.h
//
// menseki_T.h
//
// BMI Calculator
//
// Created by on 12/12/15.
// Copyright (c) 2012 __MyCompanyName__. All
// rights reserved.
//

#import <UIKit/UIKit.h>

@interface menseki_T : UIViewController

@property (weak, nonatomic) IBOutlet
UITextField *teihenText;

@property (weak, nonatomic) IBOutlet
UITextField *takasaText;

@property (weak, nonatomic) IBOutlet UIButton
*keisan_T;

@property (weak, nonatomic) IBOutlet UIButton
*reset_T;

@property (weak, nonatomic) IBOutlet UILabel *
hyouji_T;

- (IBAction)Touchkeisan_T:(id)sender;

- (IBAction)Touchreset_T:(id)sender;

@end
    
```



図 2 4 数当てゲーム



図 2 5 数当てゲームプログラムソース画面

```

//ここから書き足し
- (void)textFieldResignFirstResponder {
    [ichiText resignFirstResponder];
    [juText resignFirstResponder];
    // 一の位・十の位テキストのキーボードを隠す処理
}
- (IBAction)Touchkeisan_K:(id)sender {
    // キーボードを隠す
    [self textFieldResignFirstResponder];

    // 一の位のテキストフィールドに入力された値を取得
    float ichi = [ichiText.text floatValue];
    // 十の位のテキストフィールドに入力された値を取得
    float ju = [juText.text floatValue];

    // 結果判定を行う
    //一の位は@"である
    NSString *result_I = @"";

    // 一の位が1より小さければ、「上だよ」と表示する。
    if (ichi < 1) {
        result_I = @"「上だよ」";
        // 一の位が1より上、10より小さければ、「下だよ」と表示する。
    } else if (ichi > 1 && ichi < 10) {
        result_I = @"「下だよ」";
        // 一の位が1なら「ビンゴ!」と表示する。
    } else if (ichi == 1) {
        result_I = @"「ビンゴ!」";
        // それ以外は、「一桁の数字を入れてね」と表示する。
    } else {
        result_I = @"「一桁の数字を入れてね」";
    }

    //十の位は@"である
    NSString *result_J = @"";

    // 十の位が4より小さければ、「上だよ」と表示する。
    if (ju < 4) {
        result_J = @"「上だよ」";
        // 十の位が4より上、10より小さければ、「下だよ」と表示する。
    } else if (ju > 4 && ju < 10) {
        result_J = @"「下だよ」";
        // 十の位が4なら「ビンゴ!」と表示する。
    } else if (ju == 4) {
        result_J = @"「ビンゴ!」";
        // それ以外は、「一桁の数字を入れてね」と表示する。
    } else {
        result_J = @"「一桁の数字を入れてね」";
    }

    //判定結果は@"である
    NSString *result = @"";

    // 一の位が1、十の位が4でない場合「おいしい。あと十の位だけだ!」と表示する。
    if (ichi == 1 && ju > 4 && ju < 4) {
        result = @"「おいしい。あと十の位だけだ!」";
        // 一の位が1、十の位が4ならば、「大正解! 答は14だよ。」と表示する。
    } else if (ichi == 1 && ju == 4) {
        result = @"「大正解! 答は41だよ。」";
        // 一の位が1でなく、十の位が4なら「おいしい。あと一の位だけだ!」と表示する。
    }
}
}

//
// kazuate.h
// BMI Calculator
//
// Created by on 12/12/17.
// Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface kazuate : UIViewController

@property (weak, nonatomic) IBOutlet UITextField * ichiText;
@property (weak, nonatomic) IBOutlet UITextField * juText;
@property (weak, nonatomic) IBOutlet UIButton * keisan_K;
@property (weak, nonatomic) IBOutlet UIButton * reset_K;
@property (weak, nonatomic) IBOutlet UILabel * hyouji_K;

- (IBAction)Touchkeisan_K:(id)sender;
- (IBAction)Touchreset_K:(id)sender;

@end

```

```
BMI Calculator.xcodeproj -- kazuate.m
Build Succeeded | Yesterday at 14:57
No Issues
Run Stop Scheme Breakpoints Editor View Organizer

BMI Calculator > BMI Calculator > kazuate.m > Touchreset_K:
kazuate.h @interface kazuate

// 判定結果は@""である
NSString *result = @"";

// 一の位が1、十の位が4でない場合『おいしい。あと十の位だけだ!』と表示する。
if (ichi == 1 && ju > 4 && ju < 4) {
    result = @"『おいしい。あと十の位だけだ!』";
    // 一の位が1、十の位が4ならば、『大正解! 答は14だよ。』と表示する。
} else if (ichi == 1 && ju == 4) {
    result = @"『大正解! 答は41だよ。』";
    // 一の位が1でなく、十の位が4なら『おいしい。あと一の位だけだ!』と表示する。
} else if (ju == 4 && ichi < 1 && ichi > 1) {
    result = @"『ピンゴ!』";
    // それ以外は、『もう一度チャレンジ!』と表示する。
} else {
    result = @"『もう一度チャレンジ!』";
}

// 判定結果を結果ラベルに設定
hyouji_K.text = [NSString stringWithFormat:@"%d%d%d",
                result_J,
                @"\\n一の位は", result_I,
                @"\\n\\n", result];
}

- (IBAction)Touchreset_K:(id)sender {
    // キーボードを隠す
    [self textFieldResignFirstResponder];

    ichiText.text = @"";
    juText.text = @"";
    hyouji_K.text = @"";

    //それぞれをクリア
}
@end

//
// kazuate.h
// BMI Calculator
//
// Created by on 12/12/17.
// Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//
#import <UIKit/UIKit.h>

@interface kazuate : UIViewController
@property (weak, nonatomic) IBOutlet UITextField * ichiText;
@property (weak, nonatomic) IBOutlet UITextField * juText;
@property (weak, nonatomic) IBOutlet UIButton * keisan_K;
@property (weak, nonatomic) IBOutlet UIButton * reset_K;
@property (weak, nonatomic) IBOutlet UILabel * hyouji_K;

- (IBAction)Touchkeisan_K:(id)sender;
- (IBAction)Touchreset_K:(id)sender;

@end
```