Challenge 1: Getting Started

By Steven (Sviatoslav) Lytovka

**Part 1**

**1. Imagine asking a friend to meet you at a restaurant. What instructions would a computer program need that your friend would not in order to understand how to get there?**

A: In order to give instructions to computer we will need to break down complex idea into smallest individual instructions using programming language to write it. We would need to define variables, functions or methods, initial coordinates, exact length of each segment of the route, coordinates of turning points etc.

**2. What is the difference between source code and machine code? What does the CPU do? Where are instructions and data stored?**

A: Machine code is set of instructions and data executed directly by computer’s CPU (Central Processing Unit). Source code is set of computer instructions written in a programming language, usually as a plain text. We would need interpreter or complier to convert source code into machine code.

CPU process data into computer using ones and zeros. Once CPU receives those ones and zeros it creates commands for the computer based on user’s request. When CPU process some instructions or data it stores it in system RAM memory or the CPU cache, so the CPU can access that information quicker.

**3. What is an IDE and why is it useful?**

IDE (Integrated Development Environments) is large software that provides advanced functions for program developers, including source code editor, built automation tools and debugger. This tool helps programmer to do all development in one place. Examples: Xcode, Microsoft Visual Studio, Eclipse.

**4. What kind of language is Objective-C? Why do you think we use it for making device specific applications? Why wouldn't we want to use an interpreted language for making iPhone and iPad apps?**

Objective-C is compiled program language that supports object-oriented programming. Compiled languages usually faster than interpreted, also source code is private and they don’t require interpreter to run it. That’s why we are using compiled language. Also historically first Apple software was built on Objective-C language.

**Part 2**

**Chapter 3. Two Floats**

// main.c

// TwoFloats

//

// Created by Sviatoslav Lytovka on 8/27/13.

// Copyright (c) 2013 Sviatoslav Lytovka. All rights reserved.

//

#include <stdio.h>

int main(int argc, const char \* argv[])

{

//Declare 2 float variables called "A" and "B"

float a;

a=156.28;

float b;

b=17.33;

printf("Variable A is %f.\n", a);

printf("variable B is %f.\n", b);

//Declare 'double' variable C and assigning it sum of the "A" and "B"

double sumA\_B;

sumA\_B=a+b;

//Log that to the user

printf("The sum of A and B is equal %f.\n", sumA\_B);

return 0;

}

Output:

Variable A is 156.279999.

variable B is 17.330000.

The sum of A and B is equal 173.610001.

**Chapter 4. if/else**

Both expressions in the code snippet are true, but on the console we will have only one statement “First expression is true”. Since first statement evaluates true cycle stops.

**Chapter 5. Functions**

// main.c

// Triangle

//

// Created by Sviatoslav Lytovka on 8/28/13.

// Copyright (c) 2013 Sviatoslav Lytovka. All rights reserved.

//

#include <stdio.h>

float remainingAngle(int A, int B)

{

int C=180-A-B;

return C;

}

int main(int argc, const char \* argv[])

{

float angleA=30.0;

float angleB=60.0;

float angleC=remainingAngle(angleA, angleB);

printf("The third angle is %.2f\n", angleC);

return 0;

}

output: The third angle is 90.

**Chapter 6. Numbers**

printf("Sine of 1 radian is %.3f", sin(1));

output: Sine of 1 radian is 0.841

**Chapter 7. Loops**

// main.c

// Coolness

//

// Created by Sviatoslav Lytovka on 8/29/13.

// Copyright (c) 2013 Sviatoslav Lytovka. All rights reserved.

//

#include <stdio.h>

int main(int argc, const char \* argv[])

{

int i;

for (i=99; i>=0; i-=3) {

printf("%d\n", i);

if(i%5==0) {

printf("Found one!\n");

}

}

return 0;

}

Output:

99

96

93

90

Found one!

87

...

0

Found one!

**Chapter 8. Addresses and pointers.**

// main.c

// Addresses

//

// Created by Sviatoslav Lytovka on 8/29/13.

// Copyright (c) 2013 Sviatoslav Lytovka. All rights reserved.

#include <stdio.h>

int main(int argc, const char \* argv[])

{

printf("A float consumes %zu bytes\n", sizeof(float));

return 0;

}

A float is 4 bytes.

Since one bit is used to hold the sign, short integer can hold 215=32,768, so the smallest is -32,768, largest 32,767 (we have one zero)

Unsigned short can cold 216-1=65535.

**Chapter 10. Structs**

// main.c

// Challenge10

//

// Created by Sviatoslav Lytovka on 8/29/13.

// Copyright (c) 2013 Sviatoslav Lytovka. All rights reserved.

//

#include <time.h>

#include <stdio.h>

int main(int argc, const char \* argv[])

{

long secondsSince1970=time(NULL);

struct tm now;

localtime\_r(&secondsSince1970, &now);

printf("The time is %d:%d:%d on %d:%d:%d\n", now.tm\_hour, now.tm\_min, now.tm\_sec, now.tm\_mon, now.tm\_mday, now.tm\_year+1900);

long secondsInFourMillion=time(NULL)+4000000;

struct tm will;

localtime\_r(&secondsInFourMillion, &will);

printf("The date in 4 million seconds will be %d:%d:%d\n", will.tm\_mon+1, will.tm\_mday, will.tm\_year+1900);

return 0;

}

Output:

The time is 16:6:55 on 7:29:2013

The date in 4 million seconds will be 10:14:2013