LabVIEW for Physicists

Daniel Janse van Rensburg

June 2022

Contents

Chapter 1

Introduction

1.1 Welcome

Welcome to the LabVIEW course, for 3rd year physics students, at UJ.
In your first semester of 3^{rd} year, you have been exposed to a programming language called `C++`, an *Object-Oriented* text based computer language.
With this mighty tool in hand, you made a little microcontroller do your bidding, specifically an `ATmega328` found on the Arduino boards.
You may have noticed how terse statements in `C++` can get, sometimes obfuscating the programmers intent, see listing 1.2 on page 2.
In this course however, we will not be dealing with text based statements, instead you will make a computer bend to your will by means of pictures and threads!

Before we jump into the details, let us first compare two programs performing the same task, one written in `C`, and one assembled in `LabVIEW`.
Suppose we want to capture two decimal numbers from a user in order to compute the sum of said numbers:

C Programming

In listing 1.1, on page 2, the first line imports the required libraries for `C` to handle our input and output, or `IO`. We then have to declare what type of variables we are storing and give them names, in this case `numberA & numberB`. In this case, the values are initialised to 0, not required for this example, but good practice regardless. The numbers are then captured from the user on line 7 & 8. Finally, the results are printed so that the user may see that their system can indeed do basic arithmetic.

```c
#include<stdio.h>

int main() {

double numberA = 0;
double numberB = 0;
scanf("%f", &numberA);
scanf("%f", &numberB);

printf("%d + %d = ",numberA, numberB, numberA + numberB);
return 0;

}

```

Listing 1.1: A simple C program to add two user typed numbers

LabVIEW Programming

Without having any idea what LabVIEW is, I am sure you are able to follow what figure **??**, on page **??**, is attempting to convey.
In figure **??**, on page **??**, you will find the *Graphical User Interface*, or GUI, for the summing program. The two figures are one program developed simultaneously, we will dive into the details throughout the course.

Before you even ask, yes it is really that simple. You will find that creating programs in LabVIEW is fun and rewarding, so relax and take it easy.

By the end of this course, you will be able to create complex programs to help you solve problems in physics, as well as bridge the gap between the computer world and the real world.

```
B,i,y,u,b,I[411],*G=I,x=10,z=15,M=1e4;X(w,c,h,e,S,s){int t,
 o,L,E,d,O=e,N=-M*M,K
=78-h<<x,p,*g,n,*m,A,q,r,C,J,a=y?-x:x;y^=8;G++;d=w||s&&s>=h
 &&v 0,0)>M;do{_ o=I[
p=O]){q=o&z^y _ q<7){A=q--&2?8:4;C=o-9&z?q["& .$  "]:42;do{
 r=I[p+=C[l]-64]_!w|p
==w){g=q|p+a-S?0:I+S _!r&(q|A<3||g)||(r+1&z^y)>9&&q|A>2){_
 m=!(r-2&7))P G[1]=O,
K;J=n=o&z;E=I[p-a]&z;t=q|E-7?n:(n+=2,6^y);Z n<=t){L=r?l[r
 &7]*9-189-h-q:0 _ s)L
+=(1-q?l[p/x+5]-l[O/x+5]+l[p%x+6]*-~!q-l[O%x+6]+o/16*8:!!m
 *9)+(q?0:!(I[p-1]^n)+
```

```
7  !(I[p+1]^n)+l[n&7]*9-386+!!g*99+(A<2))+!(E^y^9)_ s>h||1<s&s
    ==h&&L>z|d){p[I]=n,O
8  [I]=m?*g=*m,*m=0:g?*g=0:0;L-=X(s>h|d?0:p,L-N,h+1,G[1],J=q|A
    >1?0:p,s)_!(h||s-1|B
9  -O|i-n|p-b|L<-M))P y^=8,u=J;J=q-1|A<7||m|!s|d|r|o<z||v
    0,0)>M;O[I]=o;p[I]=r;m?
10 *m=*g,*g=0:g?*g=9^y:0;}_ L>N){*G=O _ s>1){_ h&&c-L<0)P L _!
    h)i=n,B=O,b=p;}N=L;}
11 n+=J||(g=I+p,m=p<O?g-3:g+2,*m<z|m[O-p]||I[p+=p-O]);}}}}Z!r&
    q>2||(p=O,q|A>2|o>z&
12 !r&&++C*--A));}}}Z++O>98?O=20:e-O);P N+M*M&&N>-K+1924|d?N
    :0;}main(){Z++B<121)*G
13 ++=B/x%x<2|B%x<2?7:B/x&4?0:*l++&31;Z B=19){Z B++<99)putchar
    (B%x?l[B[I]|16]:x)_
14 x-(B=F)){i=I[B+=(x-F)*x]&z;b=F;b+=(x-F)*x;Z x-(*G=F))i=*G
    ^8^y;}else v u,5);v u,
15 1);}}
16
```

Listing 1.2: This listing is executable C code [?]

1.2 The LabVIEW Interface

1.2.1 Introduction

LabVIEW usually greats you with a launcher, figure **??**. From here you may open your recent projects, start a fresh project, or open a virtual instrument, known as a VI. For now, from the "File" drop down menu, select "New VI" or press `Ctrl+N` on your keyboard.

Two windows will pop up, a "Front Panel" and a "Block Diagram". You will need to familiarise yourself with the LabVIEW interface, this is best done by exploration, trail and error. Simply mousing over any button should give you some clue as to what the button does or what is contained in the menus. You probably would not break your computer or the LabVIEW installation by playing around with the interface.

In the next few sections we will go over what is meant by a VI, the difference between the block diagram and the front panel, how to cycle between these views, and how to place objects on these windows.

1.2.2 The VI

As stated before, VI is short for virtual instrument. The idea is that you create an interface which may be operated by a human, usually with a mouse and keyboard. This interface is called the "Front Panel" in LabVIEW. You then glue the elements of the front panel together in the "Block Diagram", this is where the LabVIEW magic happens, or the LabVIEW logic execution engine, if you do not believe in magic.

Figure **??**, on page **??**, shows one of the first front panels I have ever created along with the block diagram in figure **??**. By the end of this course you will be able to understand exactly what is going on there so do not let it intimidate you. Assert dominance over your computer, lest it assert dominance over you.

The Front Panel

Figure **??** shows an empty front panel.

1.3 LabVIEW variables and you

1.3.1 Overview

From your knowlage of mathematics, you understand the concept of a variable. This concept is of fundamental importance to all fields of computer science so let us take a slight detour before we dig further into LabVIEW.

Variables are named containers in which we may store information. It is then possible to read from, or write to, this container using it's name. At any moment in time, there are several named variables in your head, for example: `cellphoneNumber, currentDate, amountOfCoffeeHadToday,` etc.

With these examples, it is clear that some variables are not of the same type, i.e. saying "I had 3 June 2022 coffees today" makes no sense. You should know by now, or if you do not you will learn very soon, that computers are incredibly stupid and they will happily add 3 June 2022 to 072 543 7711 and give you a result. (The answer is 29 May 2045 by the way).

In general, it is your job to tell the computer what type a variable is. This is not strictly true as many programming languages can infer the data type

from your input, however for LabVIEW and programming languages such as `C` and `C++`, you are responsible.

1.3.2 Variable Types

Boolean

Perhaps the most fundamental variable type is the boolean, or bool for short. It consists of either yes or no, true or false, `1` or `0`, you get the picture. In LabVIEW, bools form the basis of nearly all your decision making code. It is represented as green icons, seen in figure **??**, on page **??**.

Floating Point

Floating point variables hold decimal numbers, in an application this may be the value of a magnetic field, the temperature of a thermocouple, basically any number that you can think of that would fit into 64bits of memory (more on that if you do computational physics in honours physics). All floating point values are represented in Labview as orange icons, seen in figure **??**, on page **??**.

Integer

The integer type is self explanatory. This is the only type of number which may be represented in LabVIEW as exact numbers and is ideal for comparison and counting. It is worth mentioning briefly that integers exist as two types namely signed and unsigned. Negative numbers are contained in signed integers, unsigned integers are strictly positive. The nuances of signedness are best left for the computational physics coarse in honours. Integer types are represented as blue icons in LabVIEW, see figure **??** on page **??**.

Strings

This line that you are reading is a string. Strings are made up of characters, itself a type of variable. Characters are not as important in LabVIEW as in other programming languages so they do not deserve their own subsection. Strings may be used to convey information to users, it may also be used to store information on your hard-disk. Strings are pink in LabVIEW, reference **??** on page **??**.

Arrays

Arrays are homogenous sections of sequential memory. Any of the above data types may be formed into an array. This variable type allows you to reference a collection of data points as a single item. Unless you derive joy from naming thousands of individual variables, you should use arrays when working with more than a handful of data points. Arrays are indexed from 0 in LabVIEW, important if you have programmed in Fortran before. Arrays follow the colour of their contents in LabVIEW, i.e. a blue array contains only integers.

Aggregate Types

Variable types may be grouped together to form a new variable type, usually with some relationship among one another. In LabVIEW, these types are known as clusters, similar to structs from the `C` family of languages. We will leave clusters for now, they are only really useful once your programs start getting big.

Chapter 2

The Second Chapter