The Basics of Calculator Programming

Lesson 1

Toby Lam 3B (13)

Introduction:

Why should you learn Calculator Programming?

- You can now harness the calculator's full potential
- You can use the programs in exams to help you do the tedious calculations
- You can make simple games with it as well.
- It's easy to learn

No experience on programming or calculators necessary.

No specific calculators are needed, a scientific calculator will do, however with some calculators you cannot type a large program. Some might not even have a pgrm function but that's very rare and common with outdated calculators

Fundamental knowledge

There are 7 variables in a standard calculator, A, B, C, D, X, Y, M. They can store integers, decimals, etc. Later on you will learn how to store numbers with it, and change what the variables stores.

There are 4 program slots that you can use, And depending on your calculator, how many characters this calculator can hold (in **all** programs)

Part 1: Configuring your calculator

To use the program mode of calculators,

1. Press the MODE button on your calculator
2. Click 6 (PGRM) or whatever button which is next to PGRM
3. Press 1(Edit)
4. Press 1(To show that you are editing program 1(out of 4))
5. Press 1(Comp) (Other modes are beyond the scope of this course)

Press the prog button on your calculator and you will see 4 commands, scroll through the pages with left and right arrow keys , and use the number key (1 to 4) to select and input the command you want.

Part 2: Basic Knowledge about commands

As discussed earlier, you can type in various commands through the prog button. We are going to discuss how you can use them. Most commands are used in a *specific combination of commands.* Here are most of the combinations and what they do.

Before I continue on, here is an example of a program, it will be useful as an example for reference if you got stuck on some topics. (I made the code in separate lines for clarity, in the calculator you type them from top to bottom, left to right, there is also no need to type in 1. 2. 3. 4. In a calculator, just type in the code)

1. ?→ A:
2. ?→ B:
3. A+B>1⇒A+B→C:
4. A+1→ A:
5. A◢
6. B◢
7. C◢

Turn to page 6 for full explanation of the code

All combination of commands must end with a ":", in order to show that the commands have ended, it will not pause the program execution

All programs execute the code from left to right

Here are some abbreviations that will be used later on:

<variable> = A,B,C,D,X,Y,M

<number> = this can be a number like 5, **or an expression, such as A/2B, A+B, A*pi,** the calculator will calculate the result of A/2B or A+B using the current values of A and B.

<statement> = a combination of commands where it doesn't involve any ⇒, and other loop commands such as if, while, for. Meaning that it can be a A+1→B, ?→ A, etc.

<logical operators> = $\geq, \neq, =, \leq$

<logic expression> = a expression in the style of <number> <logical operators> <numbers>, such as A+2B≥C, C=5…

?→ <variable>:

This will pop out a screen asking you the value of the variable when you execute the program. After you enter a number, the number will be stored in the variable, changing the value of the variable.

<number>→ <variable>:

This will assign the value of the <number> to the <variable>.

<logic expression> ⇒ <statement>:

If the logic expression is TRUE, then the statement will be executed, if the logic expression is FALSE, then the statement will NOT be executed.

For example, if A = 1

A=1⇒ A+1: A will become 2 after that line of code

A=2⇒ A-3: However at this case, A's value will not change.

<number>◢

It will display the value of the number/ expression, for example, if you typed A+B◢ It will show the value of A+B with the current values stored in A and B. **This is a line of code which does not need a ":"**

ClrMemory: (This cannot be typed by pgrm button, remind me to show you how)

Turns all variable's value into 0, basically an abbreviation of 0→ A: 0→ B: 0→ C: 0→ D: 0→ X: 0→ Y: 0→ Z:

Let's go back to our example and see what it does:

1. ?→ A:
2. ?→ B:
3. A+B>1⇒A+B→C:
4. A+1→ A:
5. A◢
6. B◢
7. C◢

First, it will ask you to input the values of A, then B.

Second, if the values of A+B are larger than 1, the value of A+B will be stored inside C.

Third, A's value will be increased by 1.

Finally, It will display the values of A, then B, then C

Part 3: Goto and Ibls

Goto and Ibls are very important in your calculator programming endeavors. I do think that it's far more efficient than if, while, for can ever be. Therefore, you must know how to use goto and Ibls before you step in to more advanced programming, this is also one of the last things that you will learn before you can start making complexed programs.

Goto makes the execution of code jumps to the corresponding Ibl, ignoring that the code should have been executed from left to right, for example.

1. Lbl 1:
2. A+1→ A:
3. A>90⇒Goto 2:
4. Goto 1
5. Lbl 2:
6. A◢

Now, when the code starts, there's a Ibl 1. The calculator will ignore it as its only purpose is for you to use it with Goto. Therefore you can put Lbl n (0 to 9 only) everywhere in the code, as long as it doesn't violate the syntax (Such as typing Ibl ATV:)

A+1 value go to A, and then if A is larger than 90, it will jump to the position of Ibl 2, that means that the code will stop executing the code right next to it, but instead of jumping to the Ibl that it is corresponded to. That means that it will ignore the goto 1, and instead show you the value of A.

If A is not larger than 90, it will not execute goto 2, but instead execute goto 1, where A +1 value go to A again and vice versa.

Part 4: Finishing up

If you are interested to the official instruction booklet, check out http://support.casio.com/storage/en/manual/pdf/EN/004/fx-50F_PLUS_EN.pdf .Go to the program mode and there will be an explanation of everything that we taught in this lesson.

http://bit.ly/2noeScp is a great source of calculator programs; they are all made by professionals and will surely give you insight to advanced programming

Exercises

1. Make a program which asks you to input A, and then showing you the result of A+1, A-1 and A*2

2. Make a program which will show 1, 2, 3, 4, 5 …… (Hint: Use Goto and Lbl)

3. Make a program which will ask you to input A and B, And then show you the result of A+B. However, if A is smaller than B, it will show you the result of B – A. (Hint: use "⇒")

Full Explanation of the program examples

1. ?→ A:
2. ?→ B:
3. A+B>1⇒A+B→C:
4. A+1→ A:
5. A◢
6. B◢
7. C◢

1. Asks the user to input the value of A
2. Asks the user to input the value of B
3. If A+B is bigger than 1, then the value of C will be changed to A+B
4. The value of A will be changed to A+1
5. Shows the value of A
6. Shows the value of B
7. Shows the value of C

Advanced Calculator Programming

Lesson 2

What this lesson is meant to teach you:

- Being efficient, use as little characters to write a program, which will result in you having the ability to type more and more complicated programs
- Advanced conditionals

In case you forgot…

<number>: a number, or an expression, such as A, A-B, C+D. It will calculate the result (Of A-B., C+D) using the current value of A,B,C,D and then show it as a number.

Bytes = characters (If you don't understand ask me, I can explain it verbally)

<logic expression> = a expression in the style of <number> <logical operators> <numbers>, such as A+2B≥C, C=5…

Part 0: Additional functions

Before we head on to the main topic of today's lesson. We will talk about some additional functions that are important but not yet covered from last lesson.

Ran# (Not typed with pgrm button, it's shift, ".")

Generates a 3 digit random number, from 0.000-0.999. Its pseudo therefore it's not truly random.

Abs (Not typed with pgrm button, its shift, ")")

Generates the magnitude of a real number without regard to its sign.

Part 1: Conditionals and abbreviations

There are actually replacements for "⇒". Here are some examples showing which is better. Before asking me how to use them, please look at the examples and try to figure it yourself, I think it's better this way, Ask me if you are really confused. The two versions convey the same meaning, but different in efficiency

If A>10: Then A+1→A: Else A-1→A: Ifend: (21 bytes, worse)

A>10⇒A+1→A: A-1→A: (17 bytes, a lot shorter visually as well)

For 1→A To 10: 2A→B: Next: (15 bytes, better)

1→A: Lbl 1: A+1→A: 2A→B: A=10⇒Goto 1: (25 bytes, used a lbl slot as well)

While A>10: A-1→A: WhileEnd: (12 bytes, better)

Lbl 1: A-1→A: A>10⇒Goto 1: (17 bytes, used a lbl slot as well)

As a conclusion, while & for are better than "⇒", however If is not. (All the commands above can be typed using Pgrm button)

Sub-Part: Abbreviation

As in calculator programming, there are many abbreviations that you can use to shorten commands, and as a result, allows you to save bytes from your program

E2, E3…

E2 means 10 to the power of 2, That means that if you wanted to type 10000, you can just type E5 (E can be typed with the EXP button)

Variable M:

There are many abbreviations for the variable M, therefore it's favored amongst professional programmers as it's much more efficient to use M compared to other variables. Here are ALL the abbreviations:

M+ (Not M & +, it can be typed as a SINGLE character by clicking "M+", which is a button on the right side of the calculator)

"<number>M+:" = "M + <number>→M:" (Saved 2 bytes)

M- (can be typed as a SINGLE character by clicking shift, "M+")

"<number>M-:" = "M - <number>→M:" (Saved 2 bytes)

10^: (This can be typed by shift, log)

"10^(" = "10 ^(" (Saved 1 bytes)

Part 2: Advanced Conditionals

In this part, one message needs to be delivered loud and clear, which is that:

<Logic expressions> = 1 or 0 = <number>

For example,

A>0 = 1, if A is bigger than 0

A>0 = 0, if A is smaller or equal to 0

So the true definition of <number>⇒<statement> is that

If the <number> = 0, the statement will not be executed

If the <number>≠ 0, the statement will be executed.

If <number> is undefined, there will be an error.

This may seem insignificant to you but it's actually very important, for example you can make a statement like:

(A>0)+(B>0) ⇒ Goto 1:

That means that either A>0 is true, or B>0 is true, or both are true, then goto 1 will be executed. IF both A>0 and B>0 is false, then goto 1 will not be executed.

(A>0)*(B>0) ⇒ Goto 1:

That means that both A>0 and B>0 must be true in order to execute goto 1, otherwise it will not execute goto 1.

Same goes for while, if and for, they all aren't about whether a statement is true or not, it's about whether the number is 0 or not.

There are also some abbreviations for "⇒"

(A>0)(B>0) ⇒ Goto 1:

A>0 ⇒ B>0 ⇒ Goto 1: (saved 3 bytes)

Part 3: Finishing up

Before we head to the various exercises, there's one more additional tip I want to show you.

If you want to show <number> in the **last line** of your program, instead of typing this:

… … A-1→A: <number> ◢

… … A-1→A: <number> (saved 1 byte)

You can also skip every ":" at the very last line of the code

Part 4: Exercises

These exercises are easy, however it's the efficiency which matters. Try to make the programs take as little bytes as possible.

1. Calculate the differences between two numbers which you input. (No Negative Outputs)

2. The user inputs the value of A and B, As A * x% = B, display the value x.

Advanced Calculator Programming

Lesson 3

What this lesson is meant to teach you:

- Functions and things which we missed out
- Tips and reminders
- Learning from examples

Part 1: Setups

There are many setups in a calculator, and you can switch them by pressing "shift", "mode" and then scroll to see the many setups while typing programs. I wouldn't explain them all but I will explain the main setups option, Fix, Sci, and Norm. After you switch to them in the program you are running. It won't be reverted (to Norm) unless you type another mode switch command. Therefore you should be careful when changing setups.

VERY IMPORTANT REMINDER: Change in setups during programs will NOT be reverted after you exit the program. Therefore I STRONGLY RECOMMEND pressing "shift", "mode", right arrow, "3", "1", to REVERT back to NORMAL setup every time you execute a SETUP-CHANGING program. Having a changed setup may result in FALSE results during normal computations.

Fix 0:

Adding Fix 0: means that every number shown through the "◢" sign will be rounded off to the nearest integer. Variable's **value** will not be changed by Fix 0: even if it's involved in A+B→C: or whatnot. Remember that it only changes the number at the last second. After the calculator does all the calculations in normal mode, the calculator rounds off the result and show it to you. It doesn't change the actual value, it just changes how it shows the number to you.

You can use Fix 1: Fix 2: etc… check out the link at next page

Rnd(<number>):

This function is only used in Fix 0: mode. What this does is round off the number to it's nearest integer. For example: Rnd(5.5) = 6

Norm 2:

There are differences between Norm 1: and Norm 2: If you want to know please check out the link at the next page. However I prefer Norm 2: for minor reasons. It's the "normal mode" which is what you have probably stuck with if you never switched setups

Sci n: (n is an integer)

Represents the number in scientific notation, the n shows how many significant digits there are. If you don't know what scientific notation is, once again, check out the link on the next page.

The link (Go to page 10) : http://support.casio.com/storage/en/manual/pdf/EN/004/fx-50F_PLUS_EN.pdf.

You can also just google "fx 50 fh II instruction booklet" or read it if you have it

Part 2: Debugging

I would've probably mentioned this verbally in lesson 1 or 2, however I would still explain it in words since it's very important.

When there's an error in your program, it's either a Syntax or Math error. When you encounter those, instead of pressing "AC", press the **right or left arrow key**. **This will direct you to the line of code which the error occurred.** Allowing you to quickly identify the error and changing it

Part 3: Learning from examples

It's crucial to learn from examples especially if you are learning about programming. Therefore I chose a slot machine game as an example. I would explain this verbally and with some notes below. (Provided by http://webcal.freetzi.com)

(Some sections in the code were replaced, as the code was using major exploits on the calculator. It does decrease the consumption in bytes but I don't think that learning how to exploit the calculator is within the scope of this course)

1. ClrMemory:
2. Lbl 1:
3. 1M-:
4. Fix 0:
5. Rnd(9Ran#→A◢
6. Rnd(9Ran#→B◢
7. Rnd(9Ran#→C◢
8. Norm 1:
9. ABC=7^3 ⇒ 92M+:
10. 3((A=B)+(A=C)+(B=C))M+:
11. M◢
12. Goto 1

Definitions of the variables:

M is money, A is the first digit of the slot machine, B is the second, C is the third.

What it actually does:

It will show the three numbers of the slot machine, A, B, C. If two of the values are the same, Then you will get 2 dollars. If three of the values are the same, Then you will get 8 dollars. If you got 7,7,7, then you will have 100 dollars. If you didn't win anything you will lose 1 dollar.

I will explain the code line by line:

1. It initializes the program by setting all variables to 0
2. It sets as a starting point as the program
3. It decreases the money by 1
4. It enables the use of Rnd(for later on
5. 9Ran# will generate a number from 0.00 to 9.99, after rounding it off through Rnd(, it will generate a integer from 0-9 (If you think about it, 0 and 9 are unlikely to be chosen compared to other numbers)
6. Same as 5.
7. Same as 5.
8. Switch back to normal mode
9. As no other combination of A,B,C can have the result of $ABC=7^3$ other than A=B=C=7 , this can be used to identify whether A=B=C=7, and if that's true, the money will be increased by 92, then 9 (at line 10.) (it sums up to 101 because 1 cash was decreased at the start of the game)
10. This is an interesting line of code, As we discussed that <logic expressions> = <number>, if two of the expressions are true, M will be increased by 3, and if all the expressions are true, M will be increased by 9. If no expressions are true, M will be the same.
11. It shows how much money you have got
12. Goes back to the starting point of the program

Exercises:

1. Try and make a game or a program of yours, you can work with your friends as a group as well. Remember what was taught in this course and use them!

The End

There's usually a lot more to learn outside than a 6 hours course, but sadly that's not the case for calculator programming. Even though there are graphical calculators which could support 3D games with lovely graphics… This is all there is to programming in a college calculator. However, all you've learned are the tools and methods, but using them to craft something elegantly is still something left to be explored. The calculator programming community in Hong Kong is although small, it can still provide a lot of elegantly built examples for you to admire and eventually, recreate them.

I would like to thank Bennie Leung, my close friend and the president of the computer society, for approving this course. I'm grateful to have the opportunity to hold a course in my first year as a committee in the computer society. I would also like to thank Cheung Chek Ka and Emerson Law for their assistance in my programming endeavors.