

This is a list of things you're supposed to learn in *Introduction to Programming*. This list is probably not complete, but does serve as a starting point. It is important to understand that this list is intended to emphasize general concepts over specifics of language: if all you learn is the particulars of a single language, you have done yourself no favors.

Not all sections of Intro will cover all these things. In some cases, this is because of the language being used to teach the class (some languages don't have iteration or variable typing); in others, time near the end of the semester may run out. If you don't learn binary searching of arrays, don't worry about it. It's more important to get a good understanding of the more basic material. Binary searching will likely be an extra for classes which master the easier material quickly. Your instructor can tell you which sections don't apply to your class at all.

The items are arranged by category, and are general. The section numbers are not chosen to match those of any particular textbook. Some items may not apply to all languages. If you are unsure whether you understand some topic, the best solution is to write a small program which will demonstrate it working, and perhaps also one that does NOT work. For example, for the item 'difference between string types and numerical types', you could write a small program that declares a number and a string, and attempts to assign one to the other. (Note that in some languages, including many dialects of BASIC, types are not strictly enforced; assigning a numerical value to a string type will result in an automatic conversion.)

Some items are accompanied by suggested programming problems which exercise your knowledge of the feature in question. If you are unsure about an item—or even if you are confident about it—you may wish to write the program and get it working to verify that you understand that element before checking it off.

Section 1: Basic Syntax

- _____ 1.1 How to insert comments
- _____ 1.2 Style: writing good comments
- _____ 1.3 Restricted keywords used by a language
- _____ 1.4 What elements, if any, are case-sensitive
- _____ 1.5 The statement terminator or line-continuation character, if the language uses either
- _____ 1.6 How to enter a string literal
 Exercise: write a program which prints out 'Hello World!'.
- _____ 1.7 How to enter character literals, if different than strings
- _____ 1.8 How to enter numeric literals, both integer and floating point (if the language distinguishes between them)
- _____ 1.9 Style: when to indent subsections of code

Section 2: Variables

- _____ 2.1 What names are legal variable names
- _____ 2.2 Declaring variables of different types
- _____ 2.3 Difference between string types, character types, and numerical types (if they are different)
- _____ 2.4 Difference between integer and floating-point numerical types
- _____ 2.5 Style: choosing understandable and clear variable names

Section 3: Operations

- _____ 3.1 Assigning values to variables of all types
- _____ 3.2 Mathematical operations
Exercise: write a program which prints out two integers, and prints out their sum, product, difference, quotient & remainder after dividing. Do the same with two floating-point numbers. Can you compute remainder for floating-point division?

Section 4: Decision Making

- _____ 4.1 **if** statements
- _____ 4.2 Two-way **if** statements using **else**
Exercise: write a program which allows the user to enter a number. The program should print out whether the number is positive or negative. Once this works, add another **if-else** statement which prints out whether the number is odd or even.
- _____ 4.3 Multi-way **if** statements using **else if** (sometimes done with a special keyword such as **elsif**)
Exercise: write a program which allows the user to enter a number. Then print out whether the number is positive, negative, or zero.
- _____ 4.4 **case** statements (sometimes called **switch** or **select**)
- _____ 4.5 How to set a **default** case if no other case matches (sometimes done with the keyword **else** or **others**)
Exercise: write a program which allows the user to enter a letter. The program should print out whether the letter is a consonant, a vowel, or sometimes a vowel (as in the case of 'y'). Use the language's default case for consonants.

Section 5: Iteration

_____ 5.1 Simple **for** loops

Exercise: write a program which prints out all the numbers between 15 and 30, and prints out whether each number is odd or even. (Requires an **if-else** inside the **for**.)

_____ 5.2 Nested **for** loops

Exercise: write a program which prints out a multiplication table with 4x15 in the upper-left and 10x3 in the lower right. Make the value in each column increase by two from the column before it, and the value in each row decrease by three from the preceding row. It should look like this:

```
4 x 15 = 60  6 x 15 = 90  8 x 15 = 120  10 x 15 = 150
4 x 12 = 48  6 x 12 = 72  8 x 12 = 96   10 x 12 = 120
4 x 9  = 48  6 x 9  = 54   8 x 9  = 72   10 x 9  = 90
4 x 6  = 24  6 x 6  = 36   8 x 6  = 48   10 x 6  = 60
4 x 3  = 12  6 x 3  = 18   8 x 3  = 24   10 x 3  = 30
```

You'll need a **for** loop for each row, and within each row you'll need a loop to do all the columns. Once you have it working, make it step by different amounts in the rows and columns. Change the start and stop points.

_____ 5.3 **while** loops

Exercise: write a program which asks the user to type in numbers, and prints out whether a number is odd or even, and whether it is positive or negative. Stop when the user enters 0. Then print out how many numbers were odd, how many were even, how many were positive, and how many were negative. Print out the largest and the smallest. Note that you can't put any limit on how many numbers might be typed in. (Requires counter variables and **if** statements inside loop.)

Section 6: Procedures & Functions

_____ 6.1 Difference between a procedure and a function, if any

_____ 6.2 Difference between formal and actual parameters

_____ 6.3 What a local variable is, and why it can only be seen from within the procedure/function in which it is declared

_____ 6.4 Declaring & calling a procedure

Exercise: write a program which uses a procedure called *swap()*. The procedure should take two integer arguments and switch them. Your program should print out the numbers before and after calling the *swap()* procedure.

_____ 6.5 Declaring & calling a function

- _____ 6.6 Returning a value from a function
Exercise: write a program which uses a function called *larger()*. The function should take two integer arguments and return the larger value. Get two numbers from the user and print the larger one.
- _____ 6.7 Useful built-in functions of a language (square roots, converting to upper- and lower- case, finding the length of a string of characters, choosing a character from the middle of a string, and so on)

Section 7: Arrays

- _____ 7.1 Declaring an array
- _____ 7.2 Legal subscripts for an array
- _____ 7.3 Using a **for** loop to print out the contents of an array
Exercise: write a program which declares an array of 15 numbers, and uses a **for** loop to get 15 numbers from the user. Then use another loop to print out the entire array, the largest number in the array, the smallest number in the array, and the arithmetic mean of all the numbers. (You will need variables to store the largest and smallest numbers and the sum, as well as tests on each number.)
- _____ 7.4 Searching an unsorted array with sequential search
Exercise: write a program which declares an array of 15 numbers, and gets 15 numbers from the user. Then allow the user to enter number to search for, and report back whether that number is in the array, along with its location if it was found.
- _____ 7.5 Searching a sorted array with binary search
Exercise: write a program which declares an array of 15 numbers, and fills in the array with numbers in order (-22, -15, -3, 0, 5, 12, and so on). Then have the user enter a number to search for, and report back whether that number is in the array, along with its location if it was found.
- _____ 7.6 Two-dimensional arrays
Exercise: write a program which declares a two-dimensional array of integers, 3 by 4. This will be used to store the vote tallies for three candidates in four districts. (The user will enter the values.) Have a one-dimensional array of size three which will store the total votes for each candidate. Print out the 2-D array, then print out the total votes, and finally print out the number of the winning candidate.