



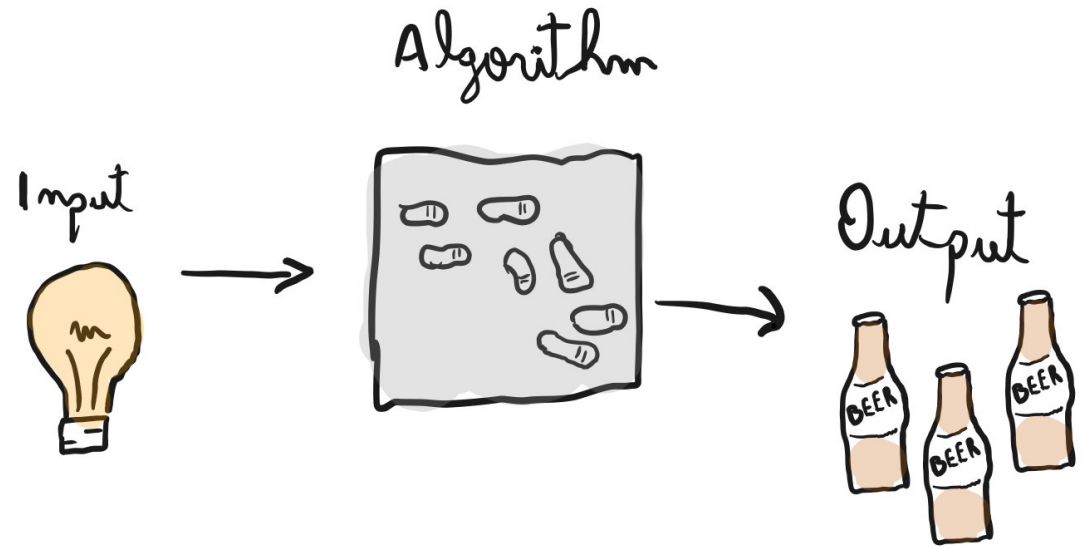
Introduction to algorithms

What is an Algorithm?

- An algorithm is a step-by-step procedure to transform a given input to the desired output and solve a computational problem. In other words, an algorithm is a tool for solving a well-specified computational problem. (Think!)
- The computational problem is a collection of questions that computers might be able to solve. For example, the problem of sorting — “Given a sequence of n integers, arrange it into increasing order.” is a computational problem.

Algorithm

Imagine it is morning, you are at home, watching some TV and eating some donuts. Then, you realize you need to buy beer because later it is going to happen a barbecue at Mark's house (by the way, imagine you have a friend called Mark). There is a sequence of steps you need to follow, since standing up from sofa until paying for the beer at the closest grocery store. This sequence of steps is called **algorithm**.

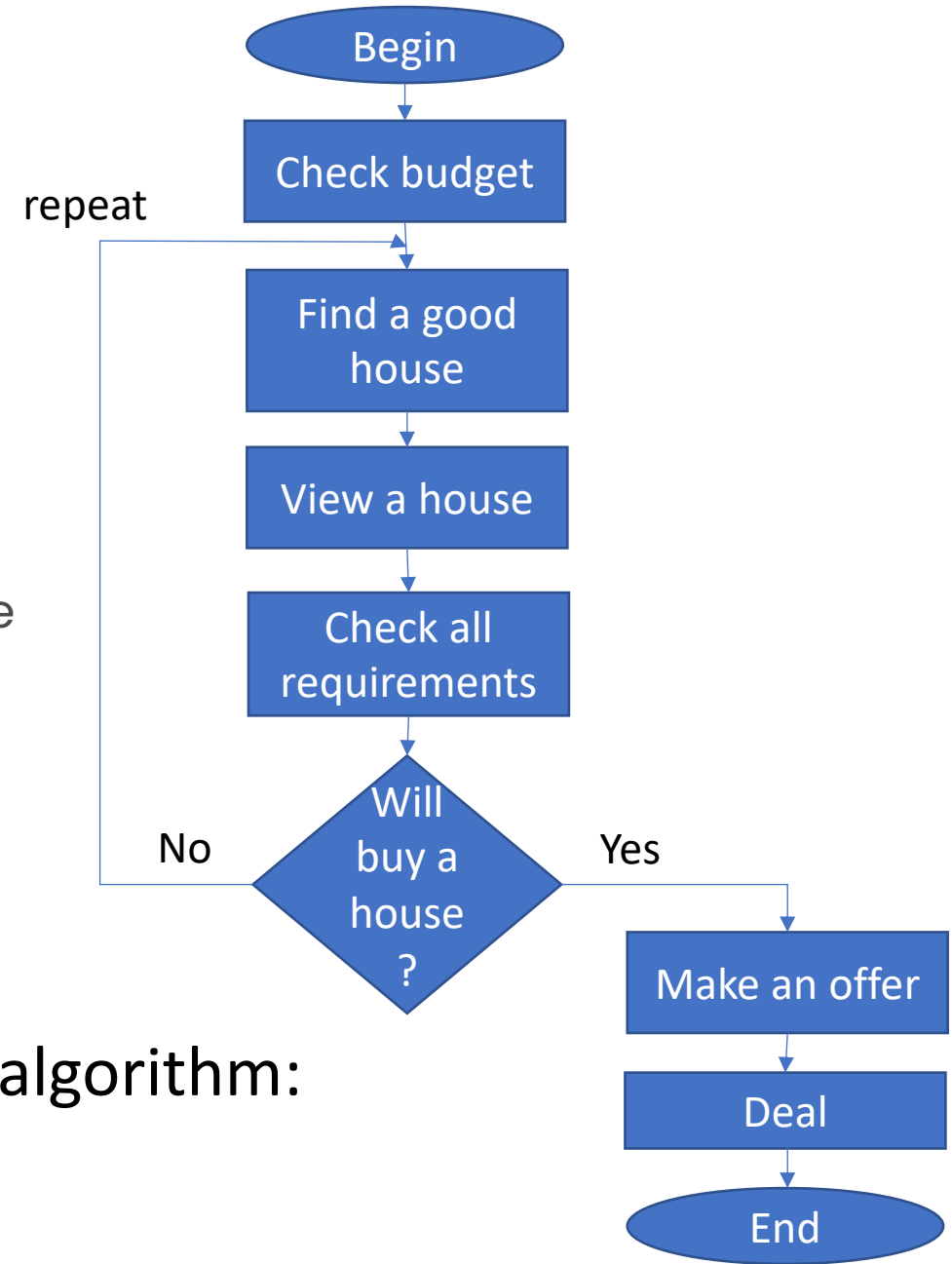


Example of an Algorithm?

An algorithm is a set of guidelines that describes how to perform a task. Think of an algorithm as step-by-step instructions that create a predictable pattern in a set of numbers or in lines of code. Mathematicians, engineers, and computer scientists develop and implement these “instructions” to provide real-world solutions.

You need to buy a house.

How will you do it: will create a simple algorithm:
linear steps + loop.



ALGORITHMS

ALGORITHMS are everywhere. They play the stockmarket, decide whether you can have a mortgage and may one day drive your car for you. They search the internet when commanded, stick carefully chosen advertisements into the sites you visit and decide what prices to show you in online shops. As Uber and Waymo will tell you, they can be the subjects of legal arguments; they cause regulatory worries too (earlier this month a group of luminaries called for a ban on battlefield robots running algorithms designed to kill people). PageRank—the algorithm that powers Google’s search results—has made its inventors very rich indeed. Algorithmically curated “filter bubbles” may even affect the way a country votes. But what exactly are algorithms, and what makes them so powerful?





Application for house buying


- Functional requirements :
want to have an application which will collect information about interesting houses:
 - address,
 - price,
 - bedrooms,
 - bathrooms,
 - floors,
 - area in sq. feet,
 - land (sq feet),
 - comments,
 - score (1..5),
 - status (New, plan to Visit, Viewed, Offer, Cancel, Done).Want to have a small application with no authorization (anyone could see it by the link), anyone could add/update/delete information about houses.
Possible to filter by status. Search by address.
- Non-Functional requirements : cloud agnostic application with simple interface.


Application



- Analog the application: Excel table with columns

Automation in Excel





1-844-759-7732
[Buy](#)
[Rent](#)
[New](#)
[Sell](#)
[Mortgage](#)
[Real Estate Agents](#)
[Feed](#)

[Log In](#)

[← Search](#)
[Overview](#)
[Property Details](#)
[Sale & Tax History](#)
[Schools](#)

A photograph of a white garage door with a brick wall and a shingled roof. A 'Street View' logo is in the bottom left corner.

585 Bluebonnet Dr, Keller, TX 76248

\$335,000

Est. \$2,298/mo [Get pre-approved](#)

3

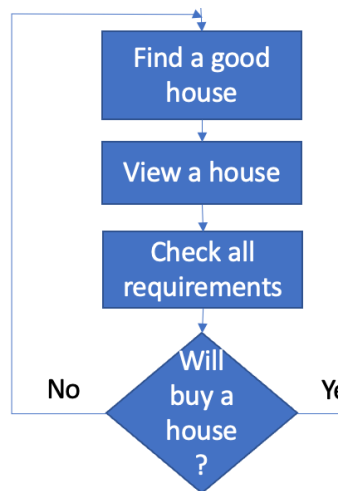
Beds

2

Baths

[illegible]

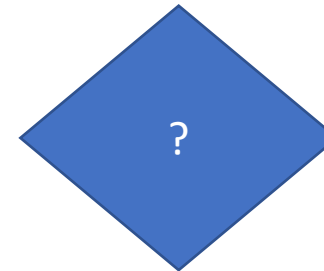
Components



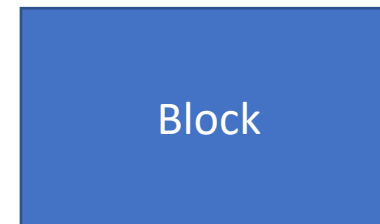
Loop



1 for each algorithm



Condition: if-else or case



Action



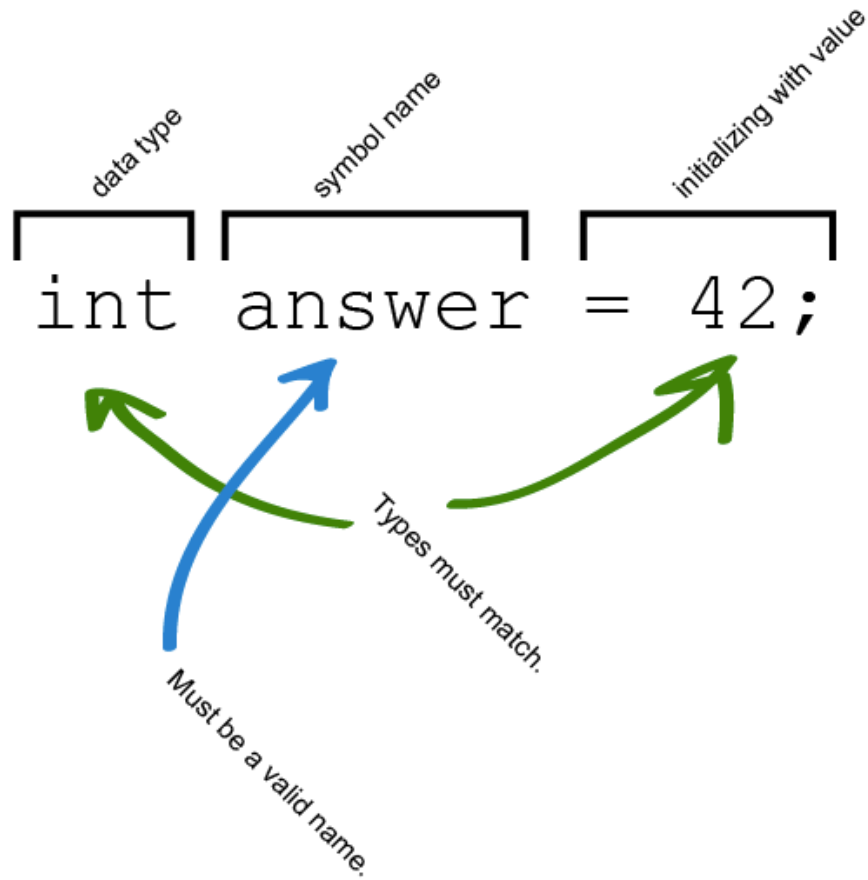
Direction



Steps

An algorithm is, essentially, a brainless way of doing clever things. It is a set of precise steps that need no great mental effort to follow but which, if obeyed exactly and mechanically, will lead to some desirable outcome.

Variable



A variable is a characteristic that can be measured and that can assume different values. **Height, age, income, province or country of birth, grades obtained at school and type of housing** are all examples of variables.

For Example, Wallet is a Variable. We could put money. It could be empty:

Wallet = null

We put 20\$ into wallet:

Wallet = 20\$

We removed 5\$

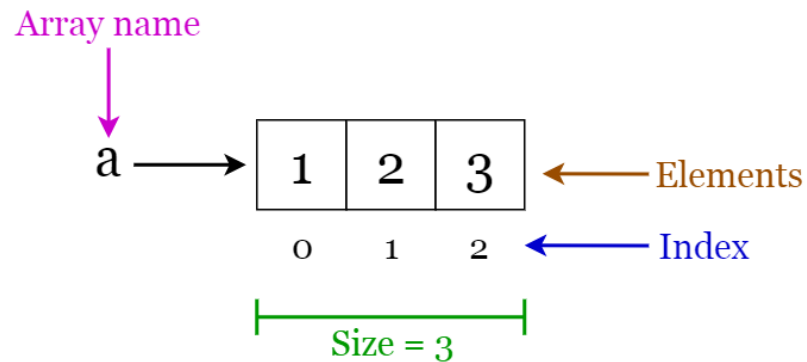
Wallet = Wallet - 5\$

Now wallet has 15\$.

print Wallet

15\$

Array

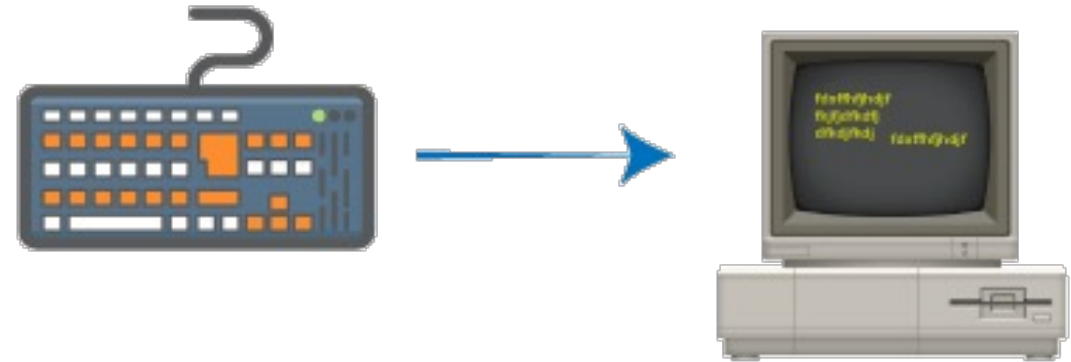


An array is a **collection of elements of the same type placed in contiguous memory locations that can be individually referenced by using an index to a unique identifier**. Five values of type `int` can be declared as an array without having to declare five different variables (each with its own identifier).

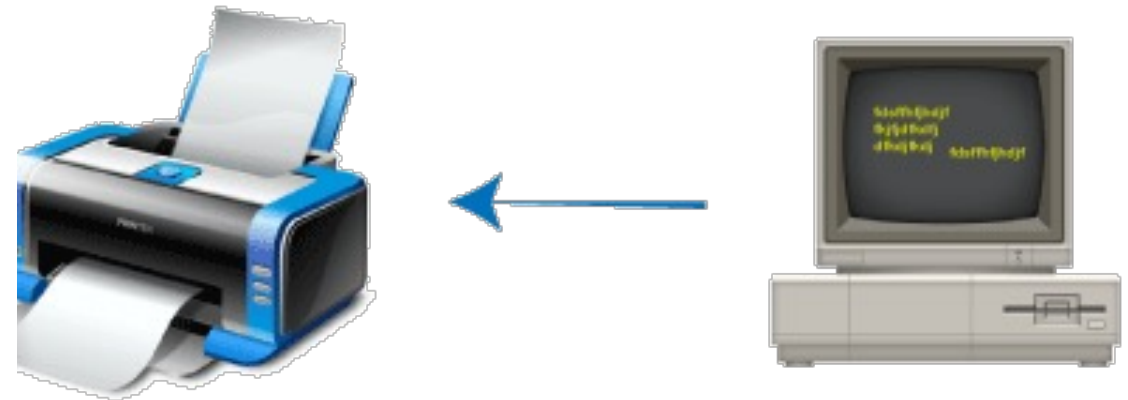
INPUT / PROCESS / OUTPUT

- An input is **data that is entered into or received by a computer**. This could include a user pressing a key on a keyboard, clicking a mouse to select something on screen or tapping a touch pad. Some inputs indicate to the computer what we want it to do, while others provide data for the computer to process.
- A process is **an instance of a program running in a computer**.
- The output is **how the computer presents the results of the process**, such as text on a screen, printed materials, or sound from a speaker.

Input Example



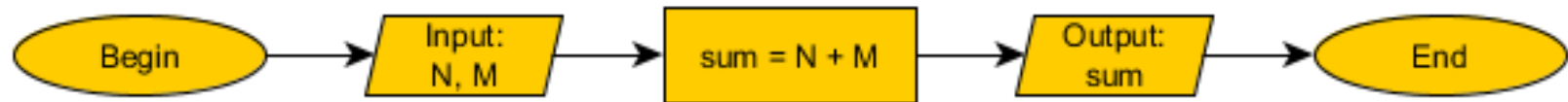
Output Example



Find the sum of two numbers N and M

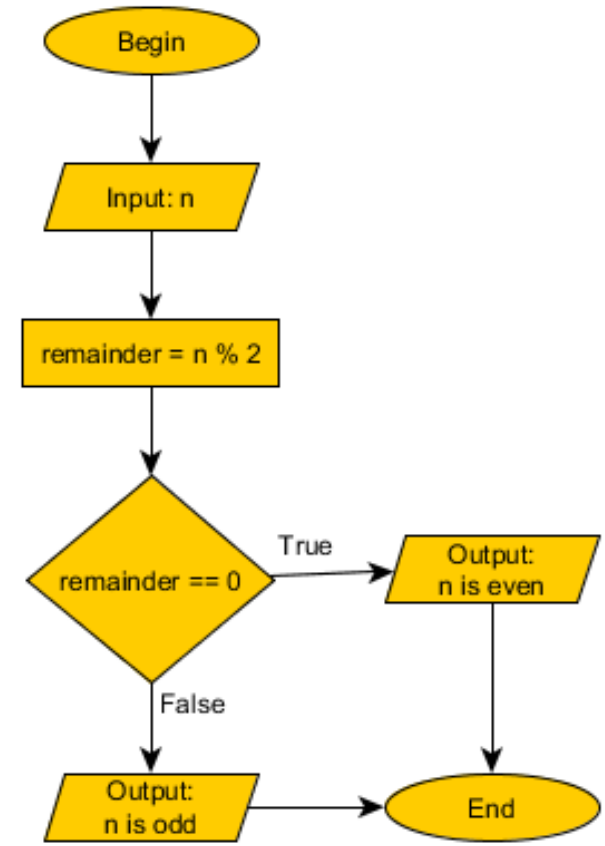
The procedure is:

1. Enter the two numbers in the variables N and M.
2. Sum them and save the result in the variable sum.
3. Output the result.

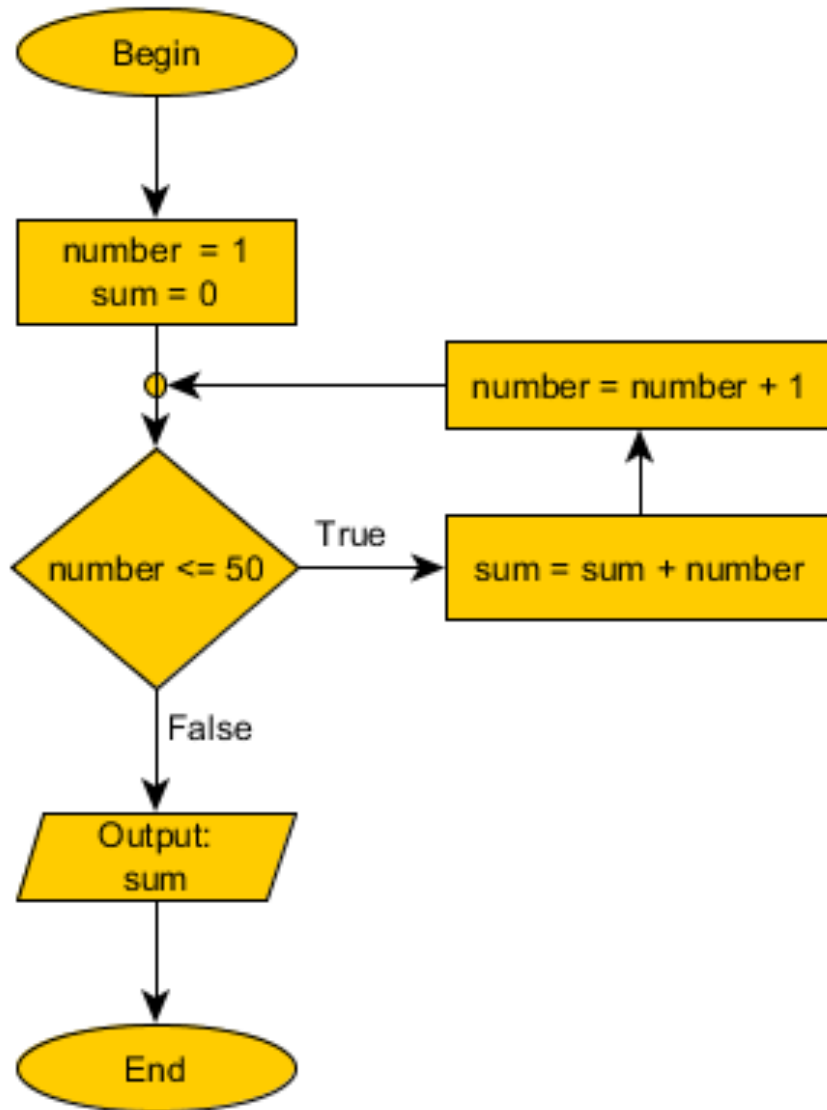


Find if a given number “n” is odd or even

A number is even if it can be divided by 2 without remainder. Such numbers are 2, 4, 6, 8.. and so on. The numbers that leave a remainder are called odd. They are 1, 3, 5, 7.. and so on. In programming we find the remainder of a division with the operator %. Also we use the double equals “==” to compare values for equality



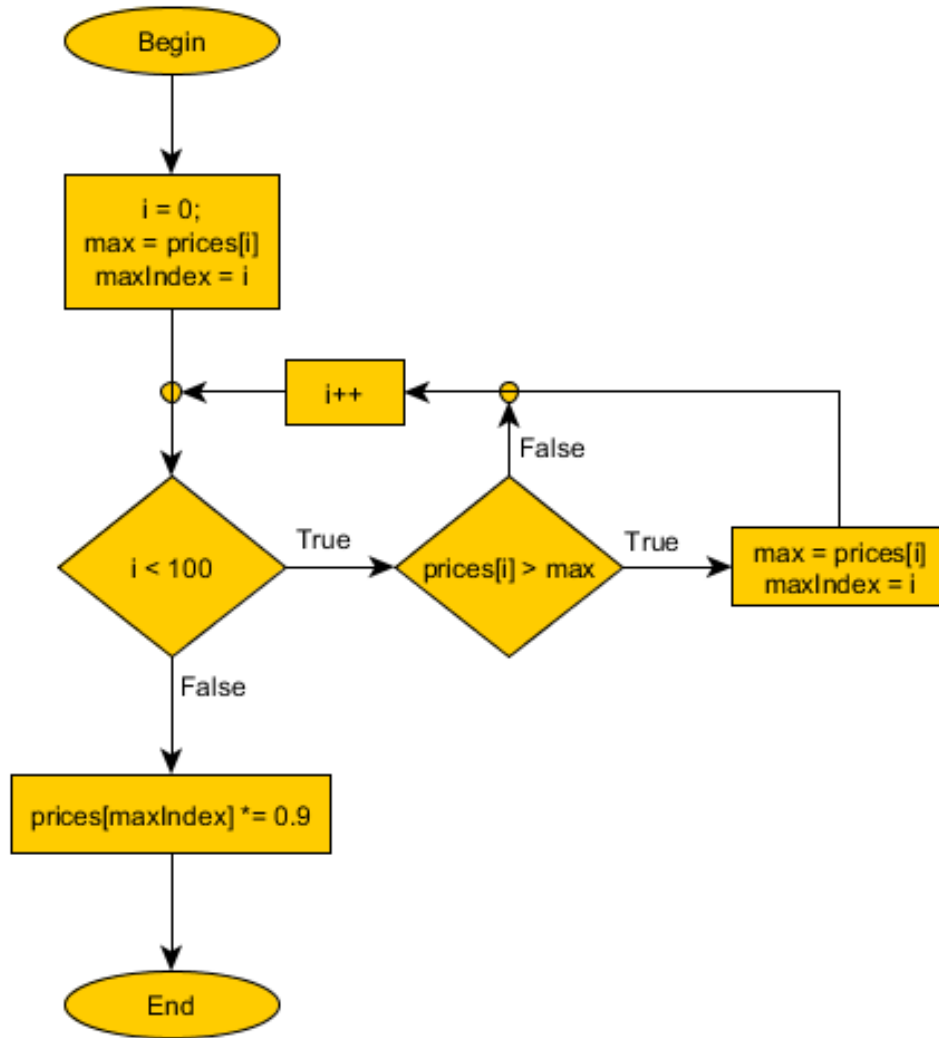
Find the sum of the first 50 numbers



Summing two numbers was easy – the calculation was one block from the flow chart. But how about 50? Do you have to write 50 blocks to solve this task? Happily – no!

You can automate this process by repeatedly incrementing the value of a variable and checking it every time if it exceeds the last value – 50. Then sum that number every step and... there you go! This construction is called a **loop**.

Find the biggest of 100 prices and reduce it by 10%



Given is the array prices with 100 elements(`prices[100]`). The problem consists of two parts

1. Find the highest price in the array
2. Reduce that price

For part 1 we iterate through the whole array, starting with index 0. We compare the first value with the next prices and when a greater price is found, we remember the new value in the variable “max” and its location in “maxIndex”.

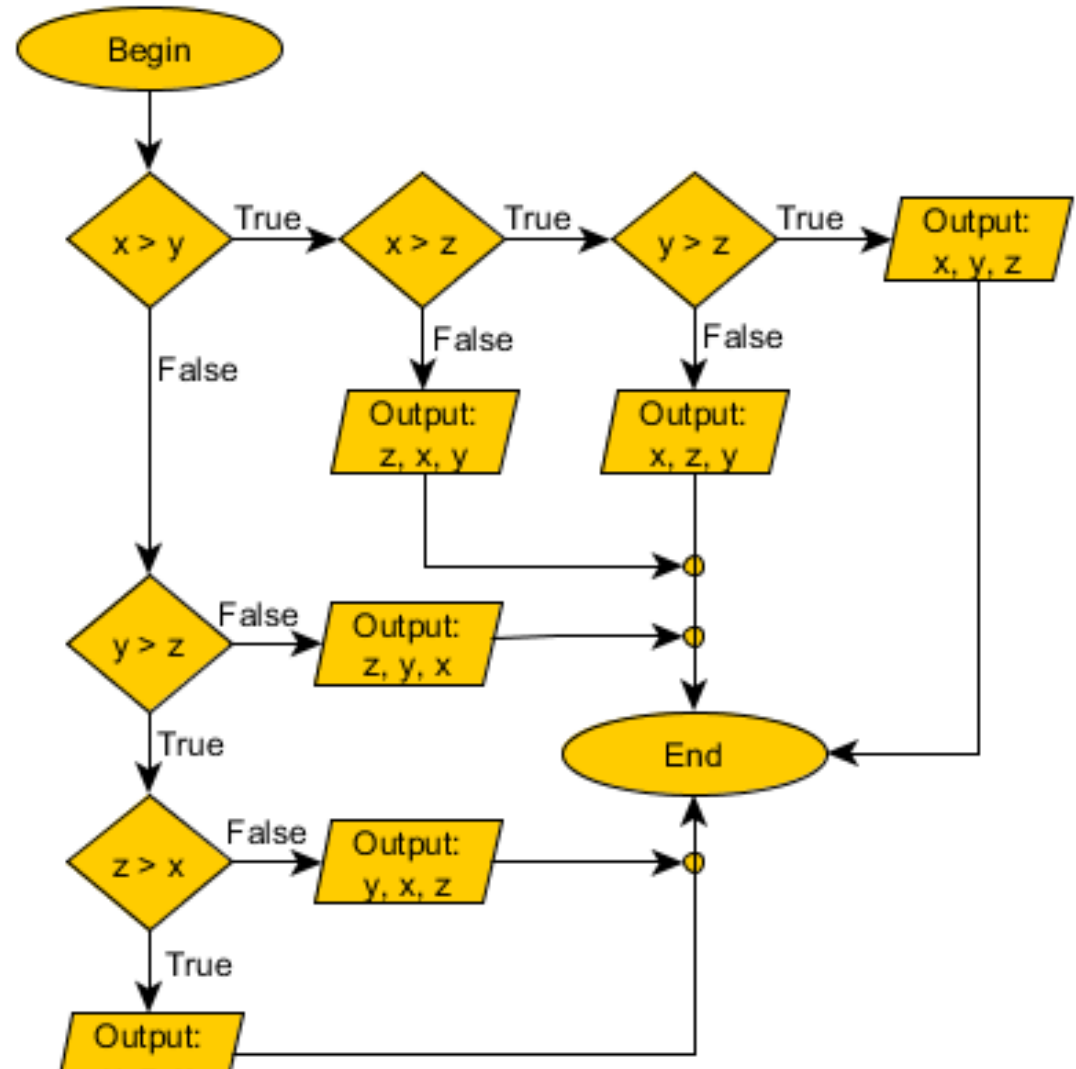
Once we compared all elements of the array we have to reduce the max price with 10%. This is the same as multiplying it by 0.9, so that is what you see in the algorithm.

The last note here – we use short version of the multiply-assign operator:

- `prices[maxIndex] *= 0.9`
is the same as
`prices[maxIndex] = prices[maxIndex] * 0.9`

Output the numbers x,y,z in descending order

The last of the algorithm examples will be more branched. As you will see, we will need to do several consecutive examinations and this will spread our flow chart a bit.



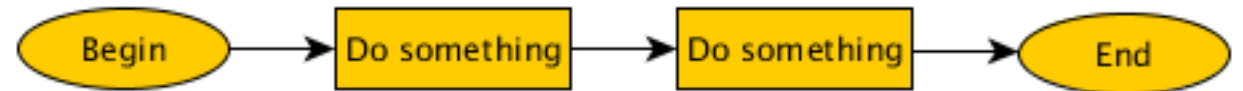
Algorithm structure types

Linear

The most simple algorithms are linear. They don't make any decisions. They just do their stuff step-by-step. Here is an example flow chart;

Here is an example in C that reads 3 numbers and then prints their sum. It does not make any decisions, but just follows the actions 1, 2, 3...

```
int main(void) {  
    int var1, var2, var3, sum;  
    printf("Please, input 3 numbers!\n");  
    scanf("%d%d%d", &var1, &var2, &var3);  
    sum = var1 + var2 + var3;  
    printf("The sum %d + %d + %d is %d.", var1, var2, var3, sum);  
    return 0; }
```

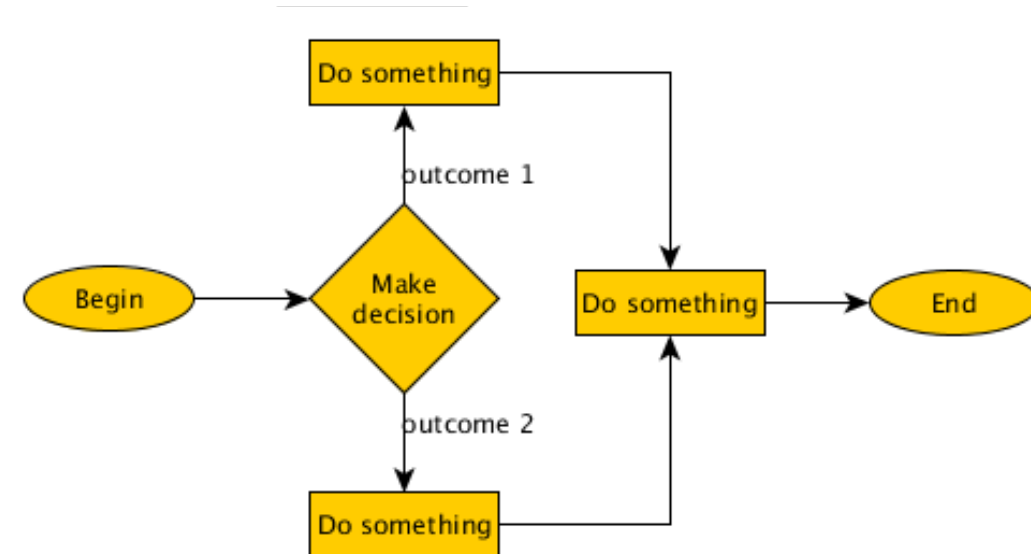


Branched Algorithm

Branched algorithms take at least one decision during its execution. They are a bit more complex than linear and more useful, too. Here is a very simple example:

And here is an example in C that finds the biggest value from 3 numbers. To make a simple decision in C, we use the [if statement](#). If we need to choose an exact value, out of many options, we can also use the [switch statement](#).

```
int main(void) {  
    int var1, var2, var3, max;  
    printf("Please, input 3 numbers!\n");  
    scanf("%d%d%d", &var1, &var2, &var3);  
    max = var1;  
    if(var2 > max) max = var2;  
    if(var3 > max) max = var3;  
    printf("%d is the max from %d, %d and %d.", max, var1, var2, var3);  
    return 0; }
```

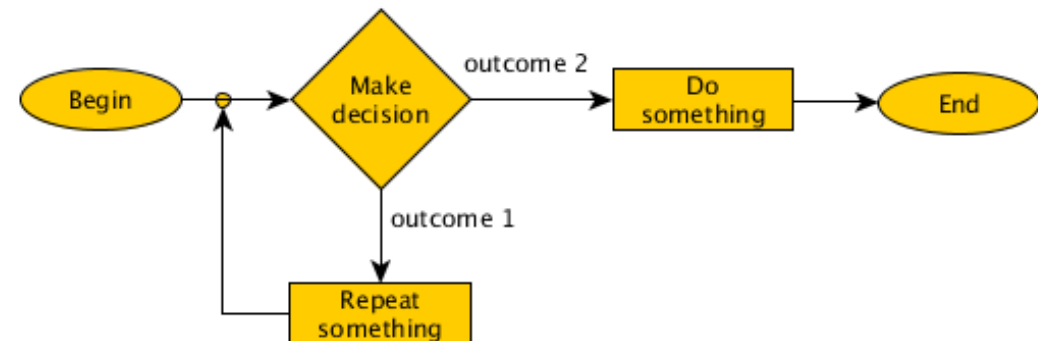


Cyclic Algorithm

Cyclic are a special type of branched ones. They take at least one decision, but one of its outcomes goes back and repeats the decision. This allows to automate a program to repeat some action many times.

Here is an example in C that reads 3 numbers and then prints their sum. It does not make any decisions, but just follows the actions 1, 2, 3...

```
int main(void) {  
  int count, temp, i, sum = 0;  
  printf("How many numbers you want to sum?\n");  
  scanf("%d", &count);  
  for(i = 1; i <= count; ++i) {  
    printf("Please input the next number: ");  
    scanf("%d", &temp); sum += temp; }  
  printf("The sum is: %d", sum);  
  return 0; }
```



Sorting

- Sorting is a very common task in programming. We may sort a table in a web page or an array of data or any other data structure. The basic reason for this is to make searching in the data faster.
- But how do we make sorting faster? It depends on the situation. Sometimes the data size is small and a simple implementation will be the best choice. In other cases we will choose a more sophisticated sorter.

HOMEWORK

1. Create an algorithm to cook an omelet.
2. Create an algorithm of Bedtime Routines.
3. Create an algorithm for Classifying Objects: based on mistakes in the homework setup a score.
4. Create an algorithm of Deciding What to Eat
5. Create an algorithm for Finding a Library Book in the Library