Discussion 11.08.19

void fibFill (int arr[], const int n)

{  
 if (n < 0)

return;

switch (n)

{

case 2:

arr[1] = 1;

case 1:

arr[0] = 1;

case 0:

break;

default:

for (int i = 2; i < n, i++)

{

arr[i] = arr[i-1] + arr[i-2];

}

}

//other methods

for (int k = 0; k < 2 && k < n; k ++)

arr [k] = 1;

}

int main ()

{

const int SIZE = 20;

int fibSeq[SIZE];

fibFill(fibSeq, SIZE);

cout << fibSeq[SIZE-1] << endl;

}

Why array?

* Grouping information (variable)
* Less code in general
* Easy to loop w/ multiple variables
* Easy to read

Properties

* Fixed size (vanilla array, not vector)
* Same type
* No size function
* Contiguous block of memory

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Variables | 1D | 2D | Nth D |
| Declaration | int foo; | int foo [5] | int foo [2][5] | int foo [][][][]… |
| Declaration + Initialization | foo <box>  String pets = “cat”; | foo <5 boxes>  String pets [4] = {“cats”, “dogs”, “turtles”, “elephants”}; | foo <10 boxes stack in two>  String pets [2][2] = {{“cats”, dogs”}, {“turtles”, “elephants”}} | foo <how ever many there is>  String pets [4][2]…[5] = {  {N-1 Array}…  }”/ |
| As Parameters | void func (string pets)  {} | void function (strint pets [], int n = 0) | void function (strint pets [][0], int n = 0, int m = 0)  size of the array -> no compilation error, n = constant error) | void fuction (string pets [][2][3][5}…  define all brackts except het fistone |
| Passed By | Pass by values (unless &, pass by reference) | Pass by reference | Pass by reference | Pass by reference |

# number of slices -> each with own grid (stacking)

Array: pass by refence – save memory (too much process with pass by values)