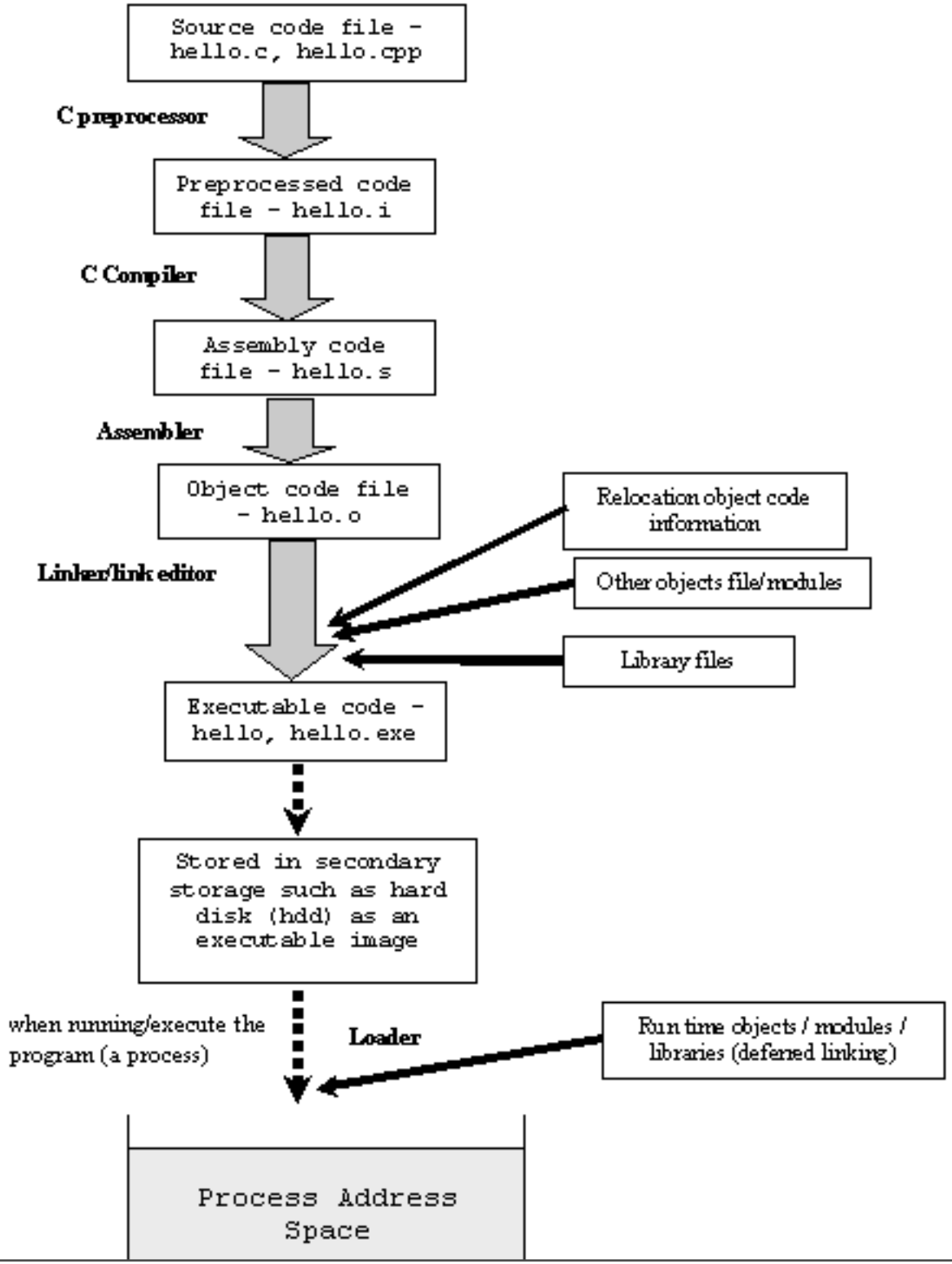
CSE 251 - Arrays

Arend Hintze

homework

```c
#include <stdio.h>
#include <math.h>

int main(int argc, const char * argv[]){
    int r,i,j;
    float dx,dy;
    printf("enter a radius between 0 and 9:");
    scanf("%d",&r);
    for(i=0;i<20;i++){
        for(j=0;j<20;j++){
            dx=(float)(10-j);
            dy=(float)(10-i);
            if(sqrt(pow(dx,2.0)+pow(dy,2.0))<r)
                printf("1");
            else
                printf("0");
        }
        printf("\n");
    }
    return 0;
}
```

compiler command -lm missing?

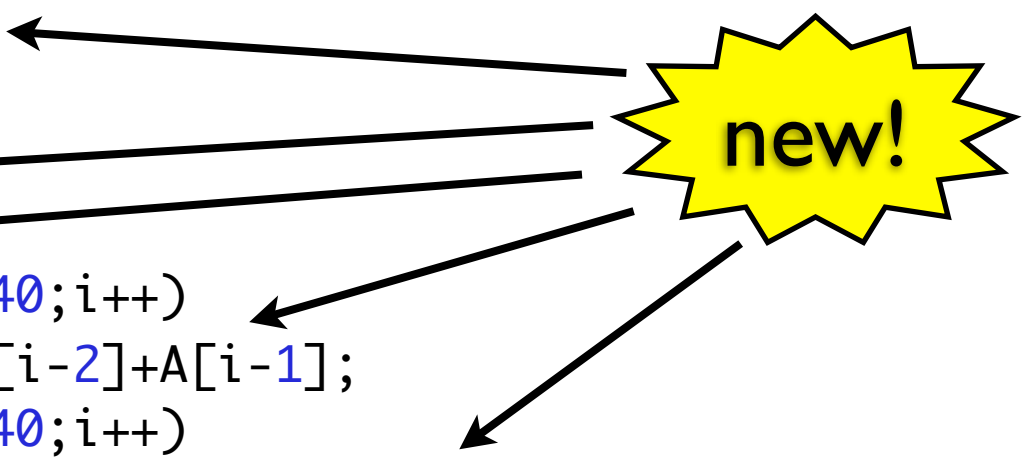

Source code file - hello.c, hello.cpp
C preprocessor
Preprocessed code file - hello.i
C Compiler
Assembly code file - hello.s
Assembler
Object code file - hello.o
Relocation object code information
Other objects file/modules
Linker/link editor
Library files
Executable code - hello, hello.exe
Stored in secondary storage such as hard disk (hdd) as an executable image
when running/execute the program (a process)
Loader
Run time objects / modules / libraries (defered linking)
Process Address Space

variables are nice, but you might run out of them fast

- imagine a list of prices, which is the cheapest?
- a database, phone book, friend list, visited sites cache
- 2D maps, images, stacks, streams, buffers...

store the fibonacci numbers

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(int argc, const char * argv[]){
    int A[40];
    int i;
    A[0]=0;
    A[1]=1;
    for(i=2;i<40;i++)
        A[i]=A[i-2]+A[i-1];
    for(i=0;i<40;i++)
        printf("%i %i\n",i,A[i]);
}
```

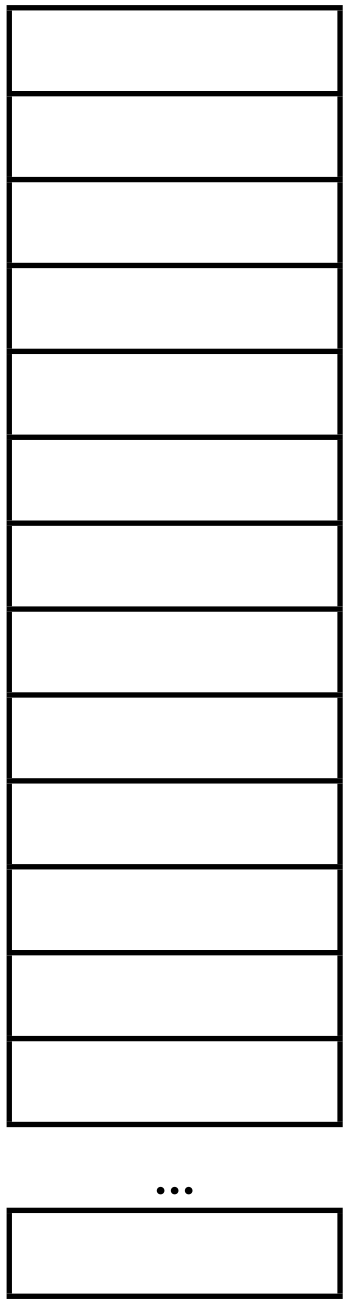


```
int A[40];
```

int A[40];

you create
a variable
A with 40
indexes
[0 ..39]

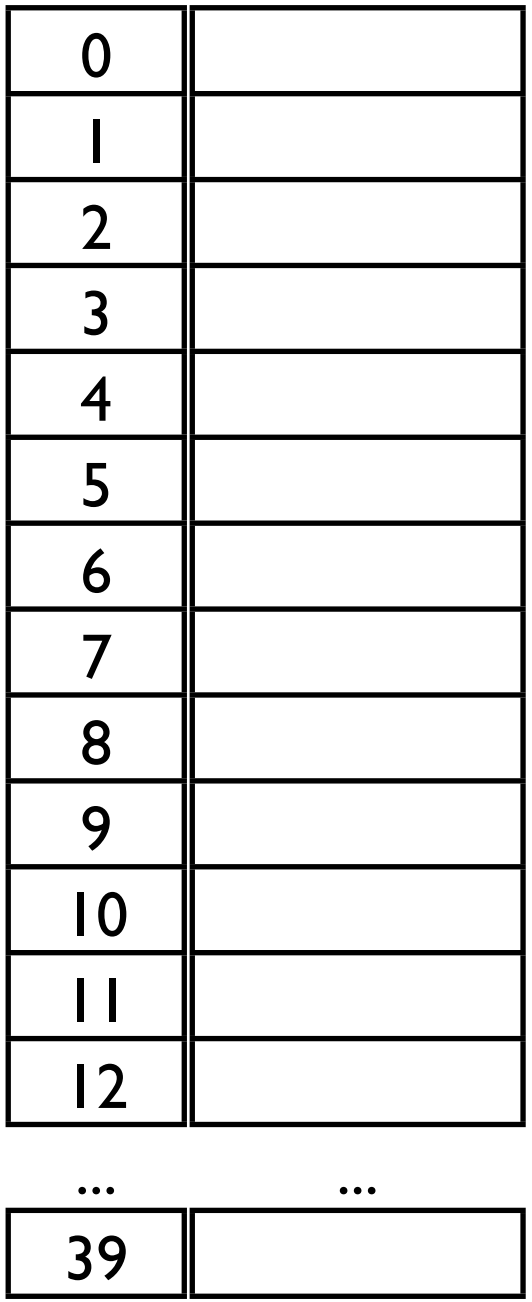
int A[40];



you create
a variable
A with 40
indexes
[0 ..39]

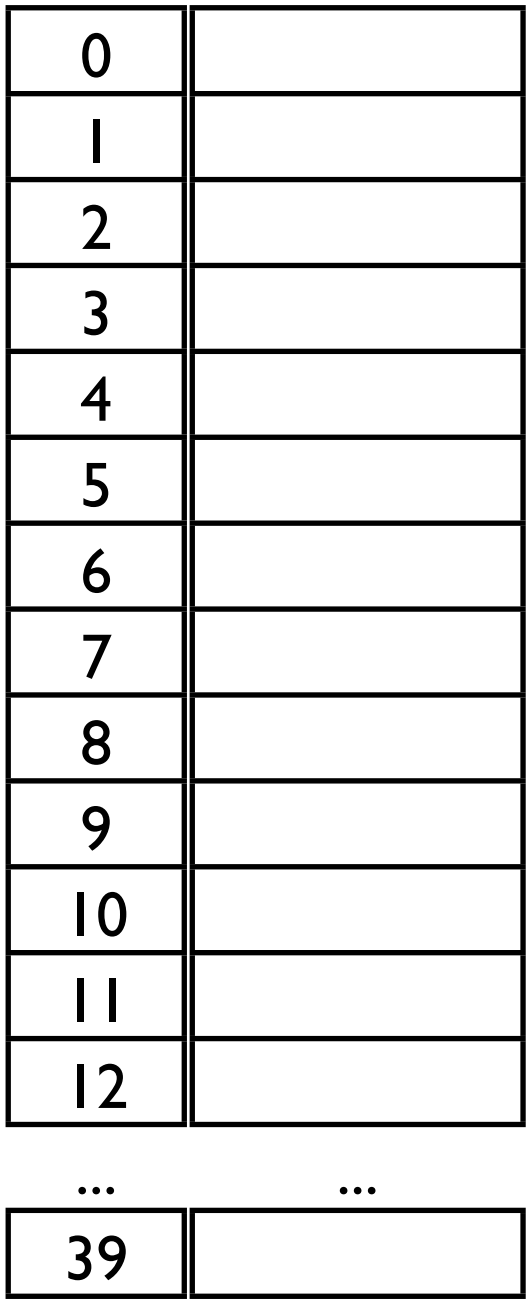
int A[40];

you create a variable A with 40 indexes [0 ..39]



int A[40];

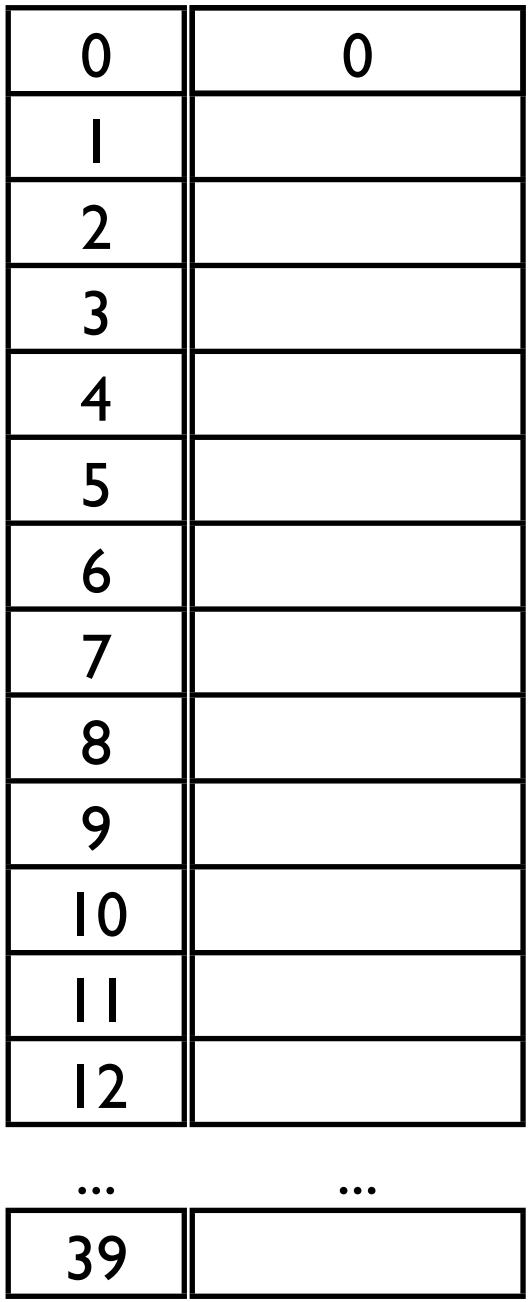
you create a variable A with 40 indexes [0 ..39]




```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(){
    int A[40];
    int i;
    A[0]=0;
    A[1]=1;
    for(i=2;i<40;i++)
        A[i]=A[i-2]+A[i-1];
}
```

int A[40];

you create a variable A with 40 indexes [0 ..39]




```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(){
    int A[40];
    int i;
    A[0]=0;
    A[1]=1;
    for(i=2;i<40;i++)
        A[i]=A[i-2]+A[i-1];
}
```

int A[40];

you create a variable A with 40 indexes [0 ..39]

0	0
1	1
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
...	...
39	

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(){
    int A[40];
    int i;
    A[0]=0;
    A[1]=1;
    for(i=2;i<40;i++)
        A[i]=A[i-2]+A[i-1];
}
```

int A[40];

you create a variable A with 40 indexes [0 ..39]

0	0
1	1
2	1
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
...	...
39	

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(){
     int A[40];
     int i;
     A[0]=0;
     A[1]=1;
     for(i=2;i<40;i++)
          A[i]=A[i-2]+A[i-1];
}
```

int A[40];

you create a variable A with 40 indexes [0 ..39]

0	0
1	1
2	1
3	2
4	
5	
6	
7	
8	
9	
10	
11	
12	
...	...
39	

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(){
     int A[40];
     int i;
     A[0]=0;
     A[1]=1;
     for(i=2;i<40;i++)
          A[i]=A[i-2]+A[i-1];
}
```

int A[40];

you create a variable A with 40 indexes [0 ..39]

0	0
1	1
2	1
3	2
4	3
5	
6	
7	
8	
9	
10	
11	
12	
...	...
39	

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(){
    int A[40];
    int i;
    A[0]=0;
    A[1]=1;
    for(i=2;i<40;i++)
        A[i]=A[i-2]+A[i-1];
}
```

int A[40];

you create a variable A with 40 indexes [0 ..39]

0	0
1	1
2	1
3	2
4	3
5	5
6	8
7	13
8	21
9	34
10	55
11	89
12	144
...	...
39	63245986

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(){
     int A[40];
     int i;
     A[0]=0;
     A[1]=1;
     for(i=2;i<40;i++)
          A[i]=A[i-2]+A[i-1];
}
```

things you can't do:

- Array from -10 to 10 ... Arrays start with index 0 ALWAYS!
- Array from 30 to 50 ... Arrays start with index 0 ALWAYS!!
- int A[]; ... Arrays need a size, seriously!
- the index of an array has to be an integer type! ... what do you mean by A[0.45]?

things you shouldn't do

- Assume the contents of an array are 0 ... they are most likely not!
- write to a negative index A[-1] ... Arrays start with 0
- write after the end of the array: int A[100]; A[200]=5;
- read from indexes behind the end of the array: int A[100]; b=A[200];

things you can do:

- arrays can have arbitrary types: int A[100]; float F[200]; char C[2]; you name it
- make them multidimensional: int A[10][10][10][10];
- pass them to a function

todo:

- make an int array size 20, filled with random numbers [0..99]
- print it
- find the highest number, print it

- special: sort the same array low to high and then high to low
- special: make a single function with two parameters: 1) the array 2) an int +1 or -1 indicating high or low

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define N 20

int main(int argc, const char * argv[]){
    int A[N];
    int i,highest;
    srand(time(NULL));
    for(i=0;i<N;i++)
        A[i]=rand()%100;
    for(i=0;i<N;i++)
        printf("%i\n",A[i]);
    highest=-1;
    for(i=0;i<N;i++)
        if(A[i]>highest)
            highest=A[i];
    printf("highest nr: %i\n",highest);
}
```

pre compiler command -> define a Macro and replaces all N with 20 (in this case)

now make this:

```
1   2  3   4   5   6
2   4  6   8   10  12
3   6  9   12  15  18
4   8  12  16  20  24
5   10 15  20  25  30
6   12 18  24  30  36
```

```c
#include <stdio.h>

int main(int argc, const char * argv[]){
    int M[6][6];
    int i,j;
    for(i=0;i<6;i++){
        for(j=0;j<6;j++){
            M[i][j]=(1+i)*(1+j);
        }
    }
    for(i=0;i<6;i++){
        for(j=0;j<6;j++)
            printf("%i\t",M[i][j]);
        printf("%i");
    }
    return 0;
}
```

now this

```
5    10 15  20  25  30
6    12 18  24  30  36
7    14 21  28  35  42
8    16 24  32  40  48
9    18 27  36  45  54
10   20 30  40  50  60
```

```c
#include <stdio.h>

int main(int argc, const char * argv[]){
    int M[6][6];
    int i,j;
    for(i=0;i<6;i++){
        for(j=0;j<6;j++){
            M[i][j]=(5+i)*(1+j);
        }
    }
    for(i=0;i<6;i++){
        for(j=0;j<6;j++)
            printf("%i\t",M[i][j]);
        printf("%i");
    }
    return 0;
}
```

finally

sum of each row

5	10	15	20	25	30	105
6	12	18	24	30	36	126
7	14	21	28	35	42	147
8	16	24	32	40	48	168
9	18	27	36	45	54	189
10	20	30	40	50	60	210
45	90	135	180	225	270	

sum of each column

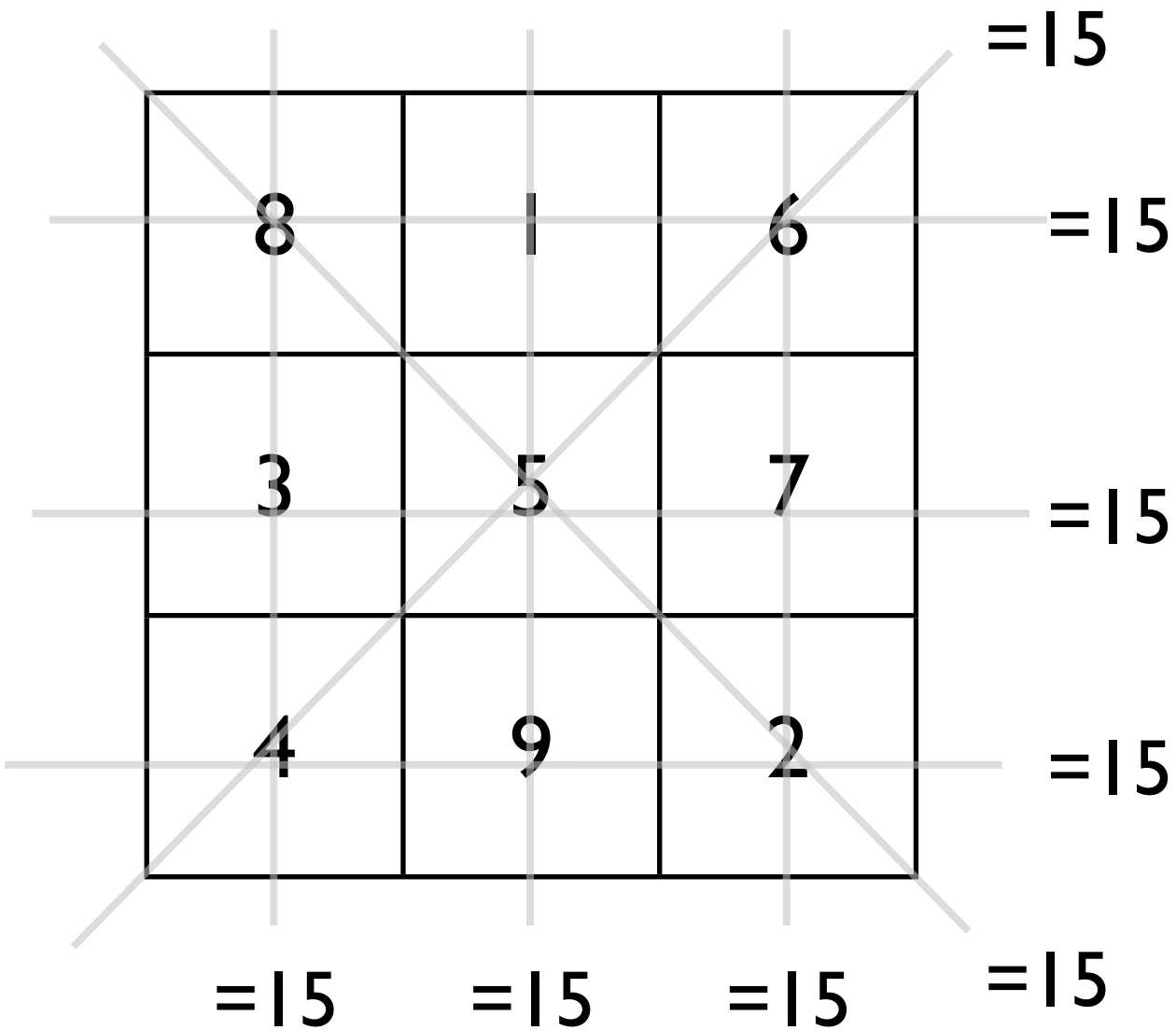
```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, const char * argv[]){
    int M[6][6];
    int R[6],C[6];
    int i,j;
    for(i=0;i<6;i++){
        R[i]=0;
        C[i]=0;
    }
    for(i=0;i<6;i++){
        for(j=0;j<6;j++){
            M[i][j]=(5+i)*(1+j);
            R[i]+=M[i][j];
            C[j]+=M[i][j];
        }
    }
    for(i=0;i<6;i++){
        for(j=0;j<6;j++)
            printf("%i\t\t",M[i][j]);
        printf("%i\n",R[i]);
    }
    for(j=0;j<6;j++)
        printf("%i\t\t",C[j]);
    printf("\n");
    return 0;
}
```

magic squares

8	1	6
3	5	7
4	9	2

I will ask for 4x4, 5x5, and maybe 6x6
(4x4 sum is 34, 5x5 sum is 65, 6x6 is 111)

magic squares



I will ask for 4x4, 5x5, and maybe 6x6
(4x4 sum is 34, 5x5 sum is 65, 6x6 is 111)

things we know

- we will need a 3x3 array
- we should be able to test if a matrix is magic
- keep making random ones until we find one
- we know one that works, maybe we use it for testing

in case that is too simple for you:
http://www.math.hmc.edu/funfacts/ffiles/10001.4-8.shtml#

```
#define Dim 3
#define Sum 15

bool isMagic(int M[Dim][Dim]){
    int i,j,k;
    //check rows
    for(i=0;i<Dim;i++){
        k=0;
        for(j=0;j<Dim;j++)
            k+=M[i][j];
        if(k!=Sum)
            return false;
    }
    //check columns
    for(i=0;i<Dim;i++){
        k=0;
        for(j=0;j<Dim;j++)
            k+=M[j][i];
        if(k!=Sum)
            return false;
    }
    //check diagonal 1
    k=0;
    for(i=0;i<Dim;i++)
        k+=M[i][i];
    if(k!=Sum)
        return false;
    //check diagonal 2
    k=0;
    for(i=0;i<Dim;i++)
        k+=M[i][Dim-1-i];
    if(k!=Sum)
        return false;
    return true;
}
```

how to test this?

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <stdbool.h>

#define Dim 3
#define Sum 15

bool isMagic(int M[Dim][Dim]);

int main(int argc, const char * argv[]){
    int Solution[3][3]={{8,1,6},{3,5,7},{4,9,2}};
    if(isMagic(Solution))
        printf("magic!\n");
    else
        printf("no magic\n");
...
```

next: making a magic square

```
for(i=0;i<Dim;i++)
    for(j=0;j<Dim;j++)
        M[i][j]=1+(rand()%9);

for(i=0;i<Dim;i++){
    for(j=0;j<Dim;j++)
        printf("%i\t",M[i][j]);
    printf("\n");
}
```

```
do{
    for(i=0;i<Dim;i++)
        for(j=0;j<Dim;j++)
            M[i][j]=1+(rand()%9);
} while(!isMagic(M));


for(i=0;i<Dim;i++){
    for(j=0;j<Dim;j++)
        printf("%i\t",M[i][j]);
    printf("\n");
}
```

yes, instead of:
while(){
}
you can say:
do{
}while();

```
do{
    for(i=0;i<Dim;i++)
        for(j=0;j<Dim;j++)
            M[i][j]=1+(rand()%9);
} while(!isMagic(M));
```

yes, instead of:

```
while(){
}
```

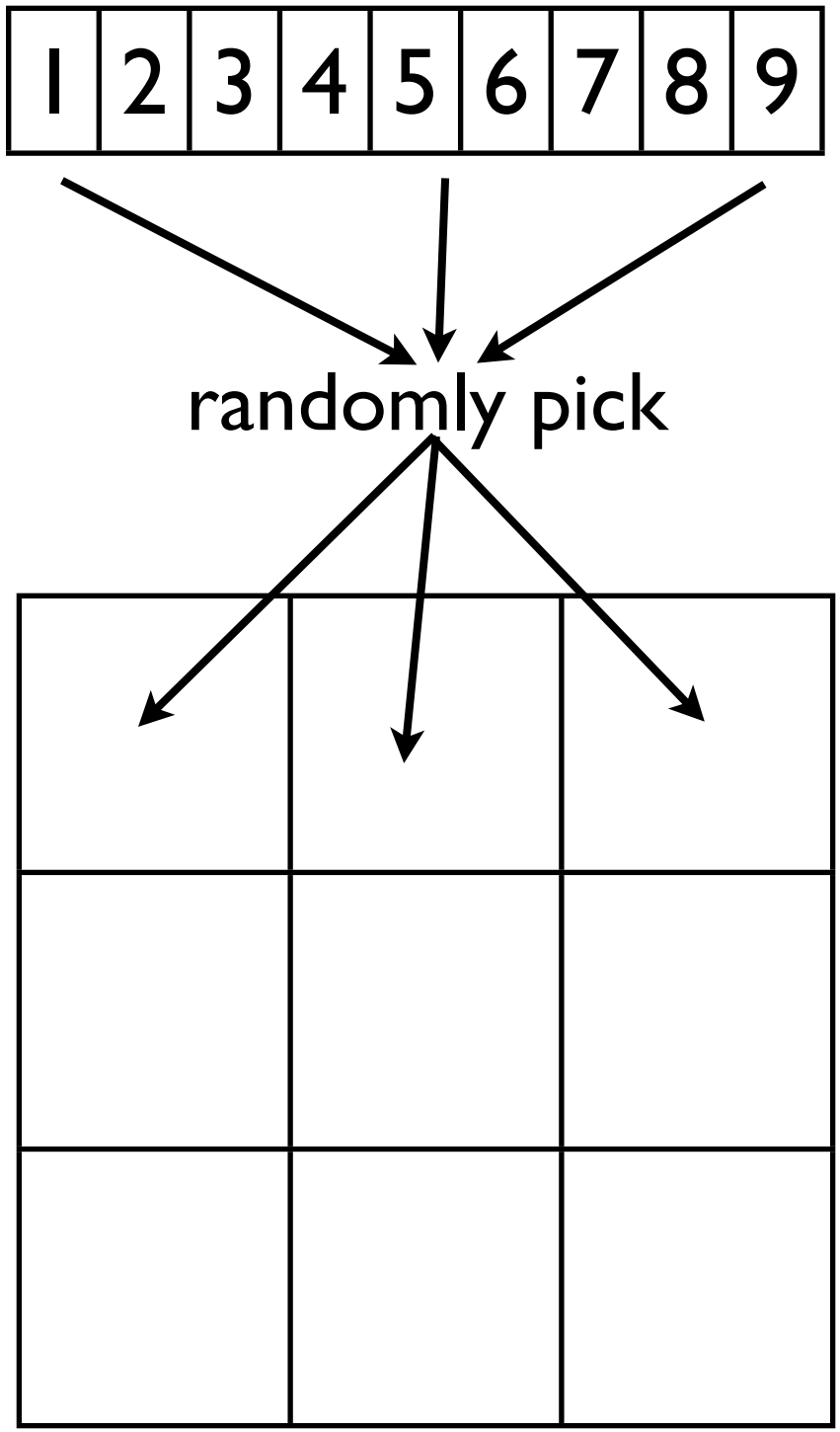
you can say:

```
do{
}while();
```

```
for(i=0;i<Dim;i++){
    for(j=0;j<Dim;j++)
        printf("%i\t",M[i][j]);
    printf("\n");
}
```

5 4 6
6 5 4
4 6 5

Wait, what is wrong?



1	2	3	4	5	6	7	8	9

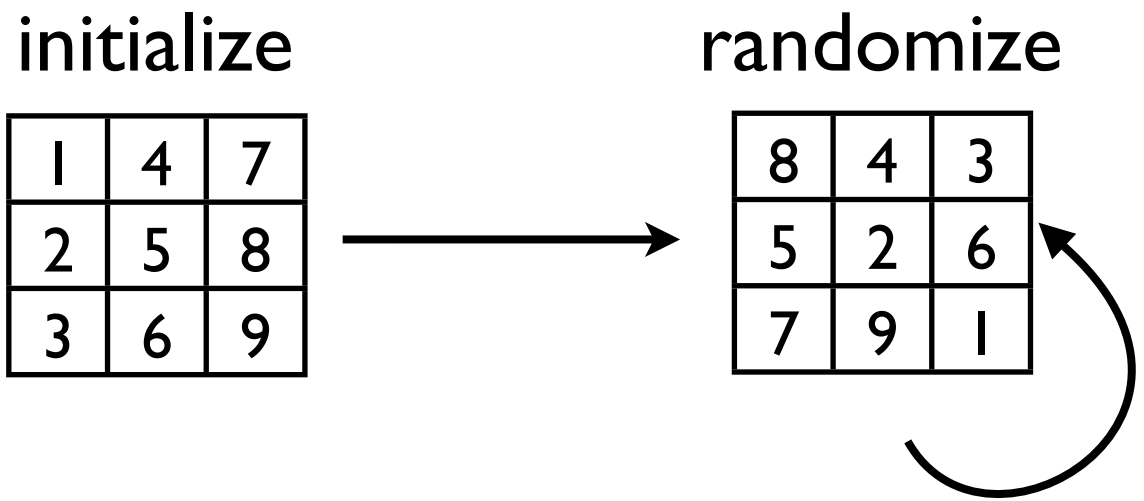
1) randomize list

8	4	3	5	2	6	7	9	1

2) pick in order

8	4	3
5	2	6
7	9	1

or:



```
k=1;
for(i=0;i<Dim;i++)
    for(j=0;j<Dim;j++){
        M[i][j]=k;
        k++;
    }
do{
    for(i=0;i<Dim*Dim;i++){
        x1=rand()%Dim;
        y1=rand()%Dim;
        x2=rand()%Dim;
        y2=rand()%Dim;
        j=M[x1][y1];
        M[x1][y1]=M[x2][y2];
        M[x2][y2]=j;
    }
} while(!isMagic(M));
```

```
6 1 8
7 5 3
2 9 4
```

How to prepare for the exam next time:

- read all the test questions on the blog, be able to answer all of them
- revisit all in class examples, be able to solve all of them
- form small groups, give each other programming problems
- be honest with yourself: Know what you know! Know what you don't know!