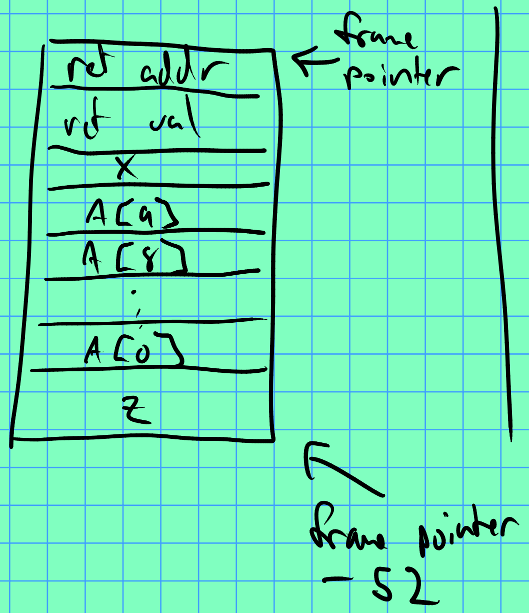
Dynamic memory & classes.

Dynamic memory: allocated as the program runs.

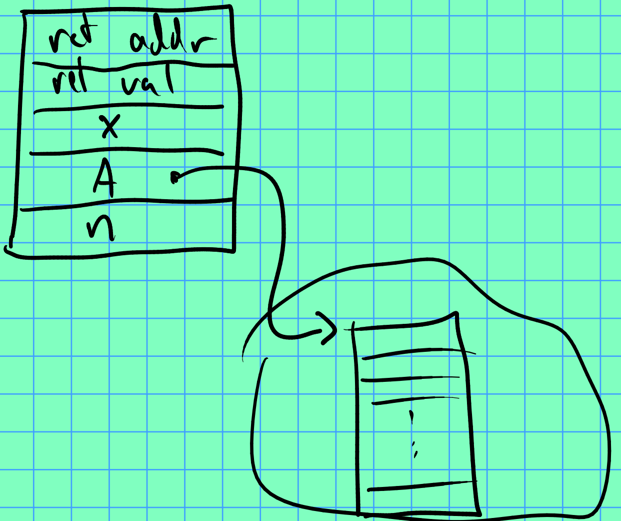
This won't be allocated on the call stack. Why?



```
int f(int x) {
  int A[10];
  int z;
    :
}
```

If size of A unknown @ compile time, Compiler couldn't compute the offset (52) for z. So we store dynamically allocated memory somewhere else ("free store", "heap").

Here's a picture:



```
int f(int x) {
  int* A;
  int n;
  cin >> n;
  A = new int[n];
    :
}
```

what does "new" do?

1. Finds a contiguous block of memory of the size you requested.
2. returns address of beginning of the block.

Important note: dynamically allocated memory is not deallocated automatically! (unless the program ends)

~~2~~ Thus, it is your responsibility to clean up unused memory allocations!

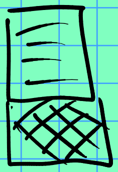
Here's how:

```
delete [] A;
```

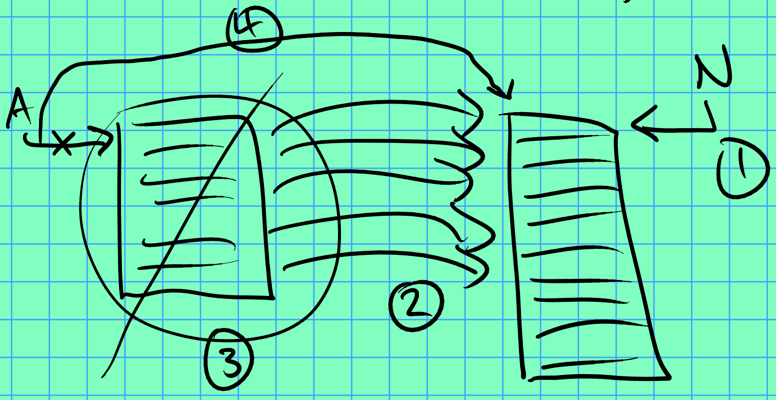
(For non arrays, don't need the brackets:

```
int * p = new int;
  :
delete p;
```
)

Example: how to resize (make bigger) an array?



```
void resize(int*& A, int n, int newsize);
```



1. make new array N
2. copy values
3. delete A
4. make A point to new one (N)

```
void resize(int*& A, int n, int newsize) {
    int* N = new int[newsize]; // ①
    for(int i=0; i<n; i++)
            N[i] = A[i];              // ②
    delete[] A;     // ③
     A = N;  //④
}
```