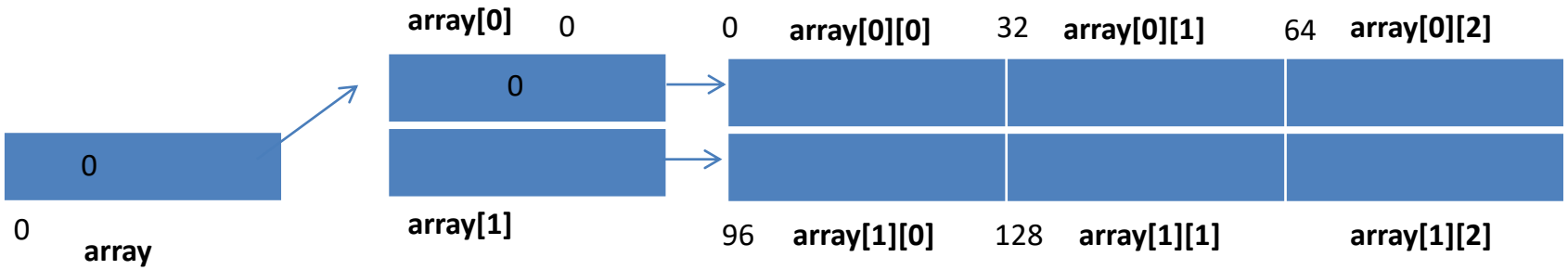
CS 161
Intro to CS I

2d arrays and Command-Line Arguments

Odds and Ends…

- Last week to demo Assignment 4
- Test next Wed., 11/22

Static 2-D arrays…



Passing a 2-D Array (Static)

```
int main() {
  int array[5][5];
  …
  pass_2darray(array);
  …
}
void pass_2darray(int a[5][5]) {
  cout << “Array at zero: ” << a[0][0] << endl;
}
```

OR

```
void pass_2darray(int a[][5]) {
  cout << “Array at zero: ” << a[0][0] << endl;
}
```

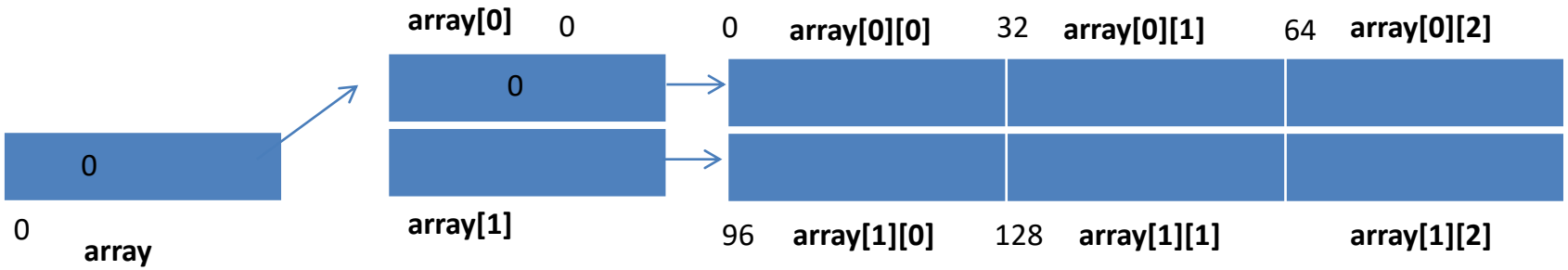
Demo…

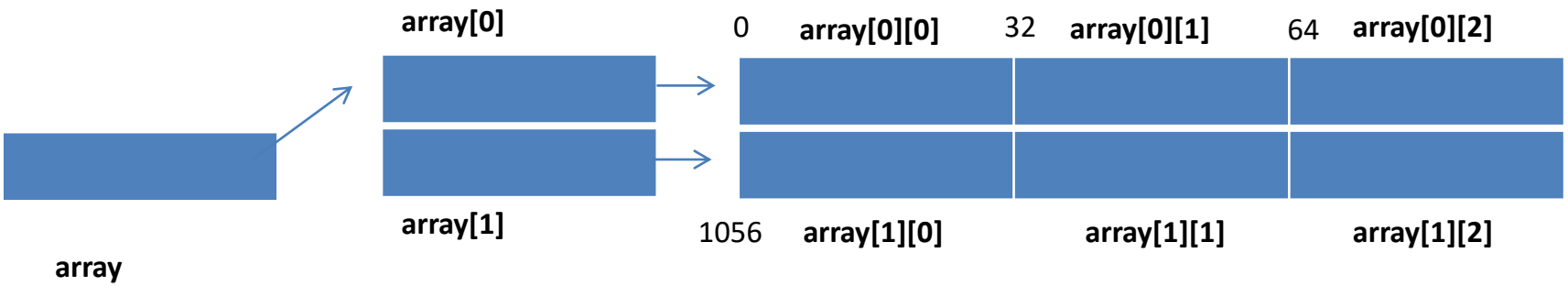
4. ENGR

Re-attach Fullscreen Stay on top Duplicate Close

```
flip2 ~/cs161/private 155% g++ 2d.cpp
flip2 ~/cs161/private 156% a.out
1
a.out
a
error
flip2 ~/cs161/private 157% a.out 2
2
a.out
a
2
flip2 ~/cs161/private 158% a.out 1 2
3
a.out
a
error
flip2 ~/cs161/private 159%
```

Static vs. Dynamic 2-D arrays…





Passing a 2-D Array (Dynamic)

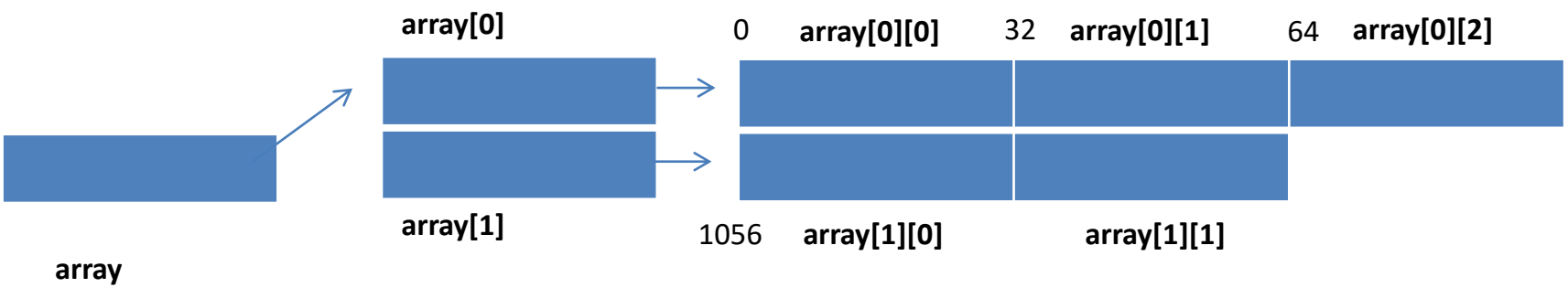
```
int main() {
  int **array;
  …
  pass_2darray(array);
  …
}
void pass_2darray(int *a[]) {
   cout << “Array at zero: ” << a[0][0] << endl;
}
```

OR

```
void pass_2darray(int **a) {
  cout << “Array at zero: ” << a[0][0] << endl;
}
```

Jagged Arrays

```
int *array[2];
array[0] = new int[3];
array[1] = new int[2];
```



Passing a 2-D Array (Dynamic)

```
int main() {
  int *array[2];
  …
  pass_2darray(array);
  …
}
void pass_2darray(int *a[]) {
   cout << “Array at zero: ” << a[0][0] << endl;
}
```

OR

```
void pass_2darray(int **a) {
  cout << “Array at zero: ” << a[0][0] << endl;
}
```

Create 2-D Array in Functions

```
int main() {
  int **array;
  …
  array = create_2darray(rows, cols);
  …
}
int **create_2darray(int r, int c) {
   int **a;
   a = new int*[r];
   for(int i=0; i<r; i++)
      a[i] = new int[c];
   return a;

}
```

Create 2-D Array in Functions

```
int main() {
  int **array;
  …
create_2darray(&array, rows, cols);
  …
}
void create_2darray(int ***a, int r, int c) {
   *a = new int*[r];
   for(int i=0; i<r; i++)
      (*a)[i] = new int[c];
}
```

Create 2-D Array in Functions

```
int main() {
  int **array;
  …
create_2darray(array, rows, cols);
  …
}
void create_2darray(int **&a, int r, int c) {
   a = new int*[r];
   for(int i=0; i<r; i++)
      a[i] = new int[c];
}
```

How does freeing memory work?

```
int *r[5], **s;

for(int i=0; i < 5; i++)
  r[i]=new int;
for(int i=0; i < 5; i++)
  delete r[i];

for(int i=0; i < 5; i++)
  r[i]=new int[5];
for(int i=0; i < 5; i++)
  delete [] r[i];


s=new int*[5];
for(int i=0; i < 5; i++)
  s[i]=new int[5];
for(int i=0; i < 5; i++)
  delete [] s[i];
delete [] s;
```