# APS105
# Winter 2012

Jonathan Deber
jdeber -at- cs -dot- toronto -dot- edu

Lecture 22
March 12, 2012

# Today

- Arrays of Strings
- Dynamic Memory Allocation

# Arrays of Strings

# Arrays of Strings

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 'A' | 's' | 's' | 'i' | 'g' | 'n' | 'm' | 'e' | 'n' | 't' | ' ' | '1' | '\0' |
| "Assignment 1" | 'A' | 's' | 's' | 'i' | 'g' | 'n' | 'm' | 'e' | 'n' | 't' | ' ' | '2' | '\0' |
| "Assignment 2" | 'A' | 's' | 's' | 'i' | 'g' | 'n' | 'm' | 'e' | 'n' | 't' | ' ' | '3' | '\0' |
| "Assignment 3" | 'A' | 's' | 's' | 'i' | 'g' | 'n' | 'm' | 'e' | 'n' | 't' | ' ' | '4' | '\0' |
| "Assignment 4" | 'A' | 's' | 's' | 'i' | 'g' | 'n' | 'm' | 'e' | 'n' | 't' | ' ' | '5' | '\0' |
| "Assignment 5" | 'C' | 'o' | 'd' | 'e' | 'L' | 'a' | 'b' | 's' | '\0' | '\0' | '\0' | '\0' | '\0' |
| "CodeLabs" | 'M' | 'i' | 'd' | 't' | 'e' | 'r' | 'm' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' |
| "Midterm" | 'F' | 'i' | 'n' | 'a' | 'l' | ' ' | 'E' | 'x' | 'a' | 'm' | '\0' | '\0' | '\0' |
| "Final Exam" | | | | | | | | | | | | | |

```
char marks[][] = {"Assignment 1", "Assignment 2",
                  "Assignment 3", "Assignment 4",
                  "Assignment 5", "CodeLabs",
                  "Midterm", "Final Exam"};
```

Error

```
error: array type has incomplete element type
```

```
int sum2DArray(int a[][], int n);
```

Error

```
int sum2DArray(int a[][COLS], int n);
```

```
char marks[][12+1] = {"Assignment 1", "Assignment 2",
                      "Assignment 3", "Assignment 4",
                      "Assignment 5", "CodeLabs",
                      "Midterm", "Final Exam"};
```

# Arrays of Strings

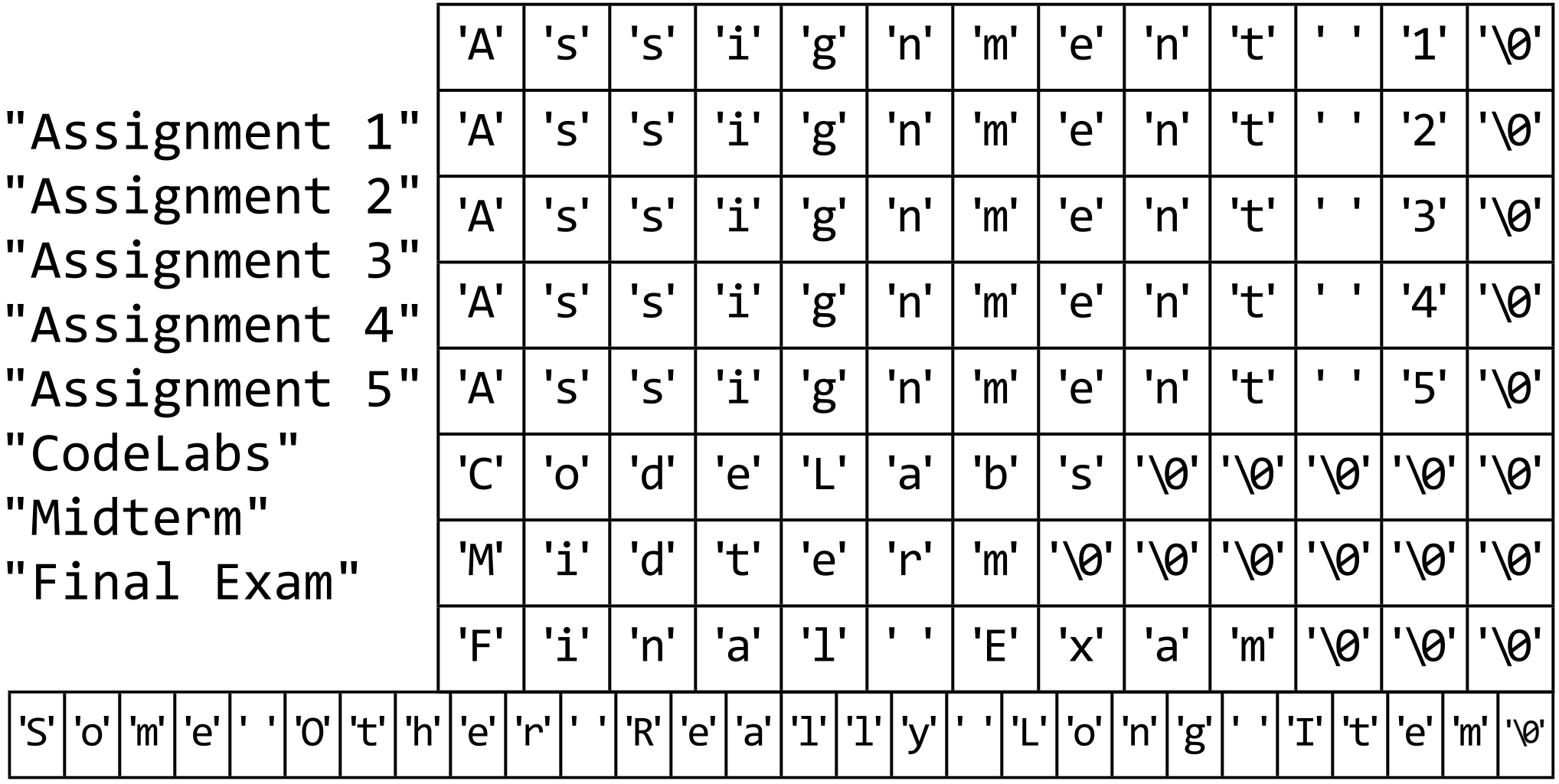
| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 'A' | 's' | 's' | 'i' | 'g' | 'n' | 'm' | 'e' | 'n' | 't' | ' ' | '1' | '\0' |
| "Assignment 1" | 'A' | 's' | 's' | 'i' | 'g' | 'n' | 'm' | 'e' | 'n' | 't' | ' ' | '2' | '\0' |
| "Assignment 2" | 'A' | 's' | 's' | 'i' | 'g' | 'n' | 'm' | 'e' | 'n' | 't' | ' ' | '3' | '\0' |
| "Assignment 3" | 'A' | 's' | 's' | 'i' | 'g' | 'n' | 'm' | 'e' | 'n' | 't' | ' ' | '4' | '\0' |
| "Assignment 4" | 'A' | 's' | 's' | 'i' | 'g' | 'n' | 'm' | 'e' | 'n' | 't' | ' ' | '5' | '\0' |
| "Assignment 5" | 'C' | 'o' | 'd' | 'e' | 'L' | 'a' | 'b' | 's' | '\0' | '\0' | '\0' | '\0' | '\0' |
| "CodeLabs" | 'M' | 'i' | 'd' | 't' | 'e' | 'r' | 'm' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' |
| "Midterm" | 'F' | 'i' | 'n' | 'a' | 'l' | ' ' | 'E' | 'x' | 'a' | 'm' | '\0' | '\0' | '\0' |
| "Final Exam" | | | | | | | | | | | | | |

# Arrays of Strings

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 'A' | 's' | 's' | 'i' | 'g' | 'n' | 'm' | 'e' | 'n' | 't' | ' ' | '1' | '\0' |
| "Assignment 1" | 'A' | 's' | 's' | 'i' | 'g' | 'n' | 'm' | 'e' | 'n' | 't' | ' ' | '2' | '\0' |
| "Assignment 2" | 'A' | 's' | 's' | 'i' | 'g' | 'n' | 'm' | 'e' | 'n' | 't' | ' ' | '3' | '\0' |
| "Assignment 3" | 'A' | 's' | 's' | 'i' | 'g' | 'n' | 'm' | 'e' | 'n' | 't' | ' ' | '4' | '\0' |
| "Assignment 4" | 'A' | 's' | 's' | 'i' | 'g' | 'n' | 'm' | 'e' | 'n' | 't' | ' ' | '5' | '\0' |
| "Assignment 5" | 'C' | 'o' | 'd' | 'e' | 'L' | 'a' | 'b' | 's' | '\0' | '\0' | '\0' | '\0' | '\0' |
| "CodeLabs" | 'M' | 'i' | 'd' | 't' | 'e' | 'r' | 'm' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' |
| "Midterm" | 'F' | 'i' | 'n' | 'a' | 'l' | ' ' | 'E' | 'x' | 'a' | 'm' | '\0' | '\0' | '\0' |
| "Final Exam" | | | | | | | | | | | | | |

"Some Other Really Long Item"

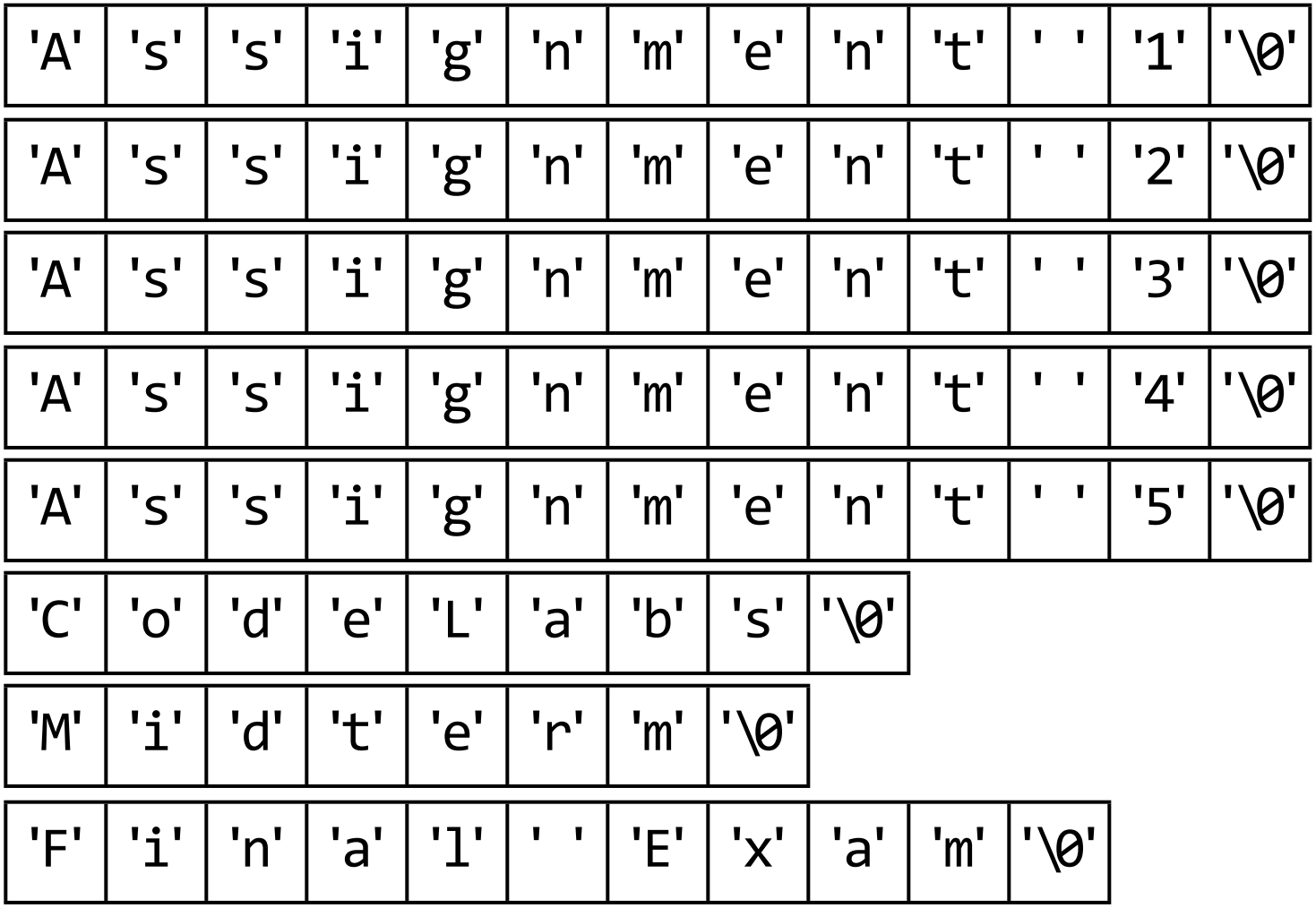
# Arrays of Strings

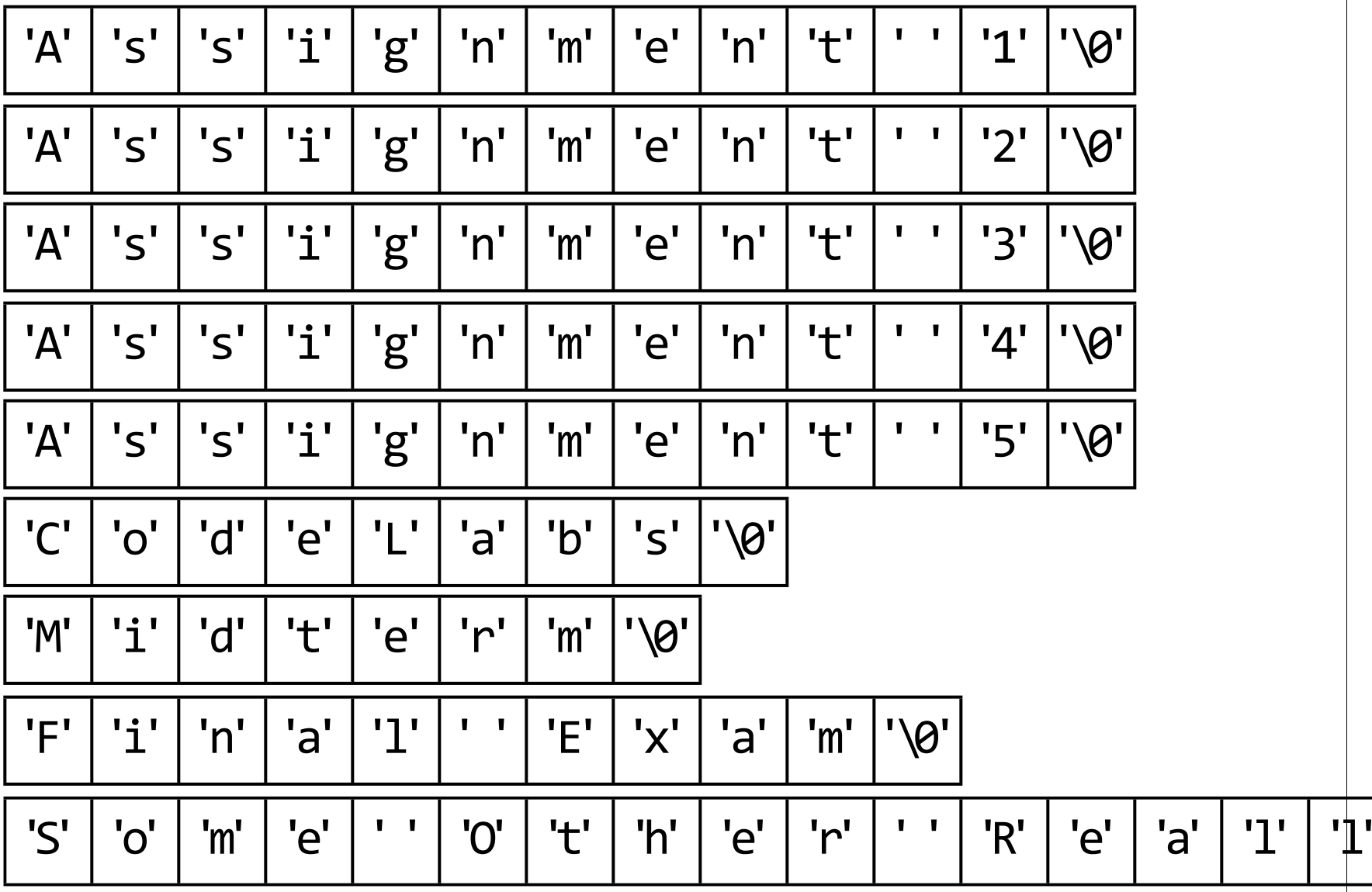


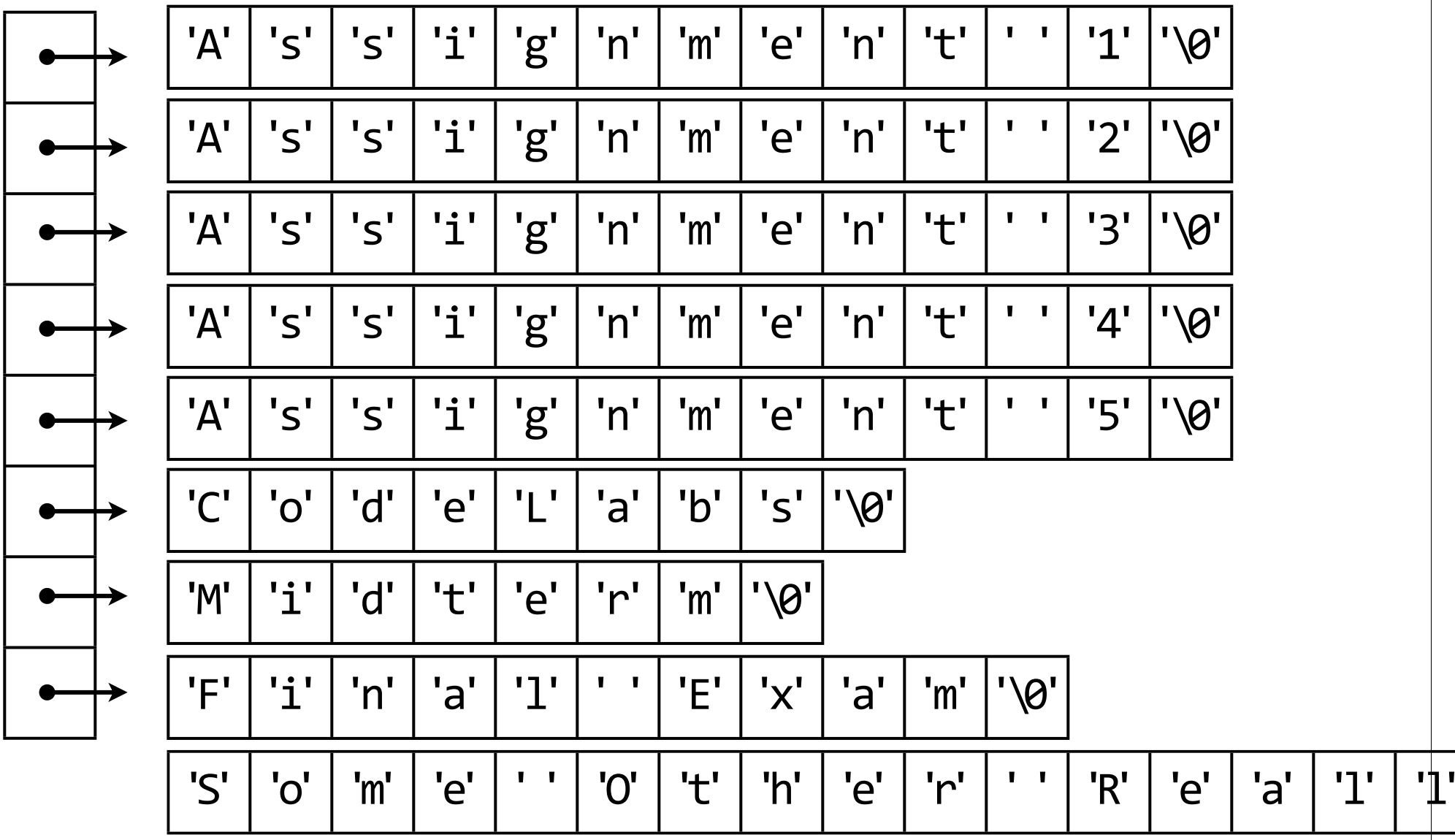

"Some Other Really Long Item"

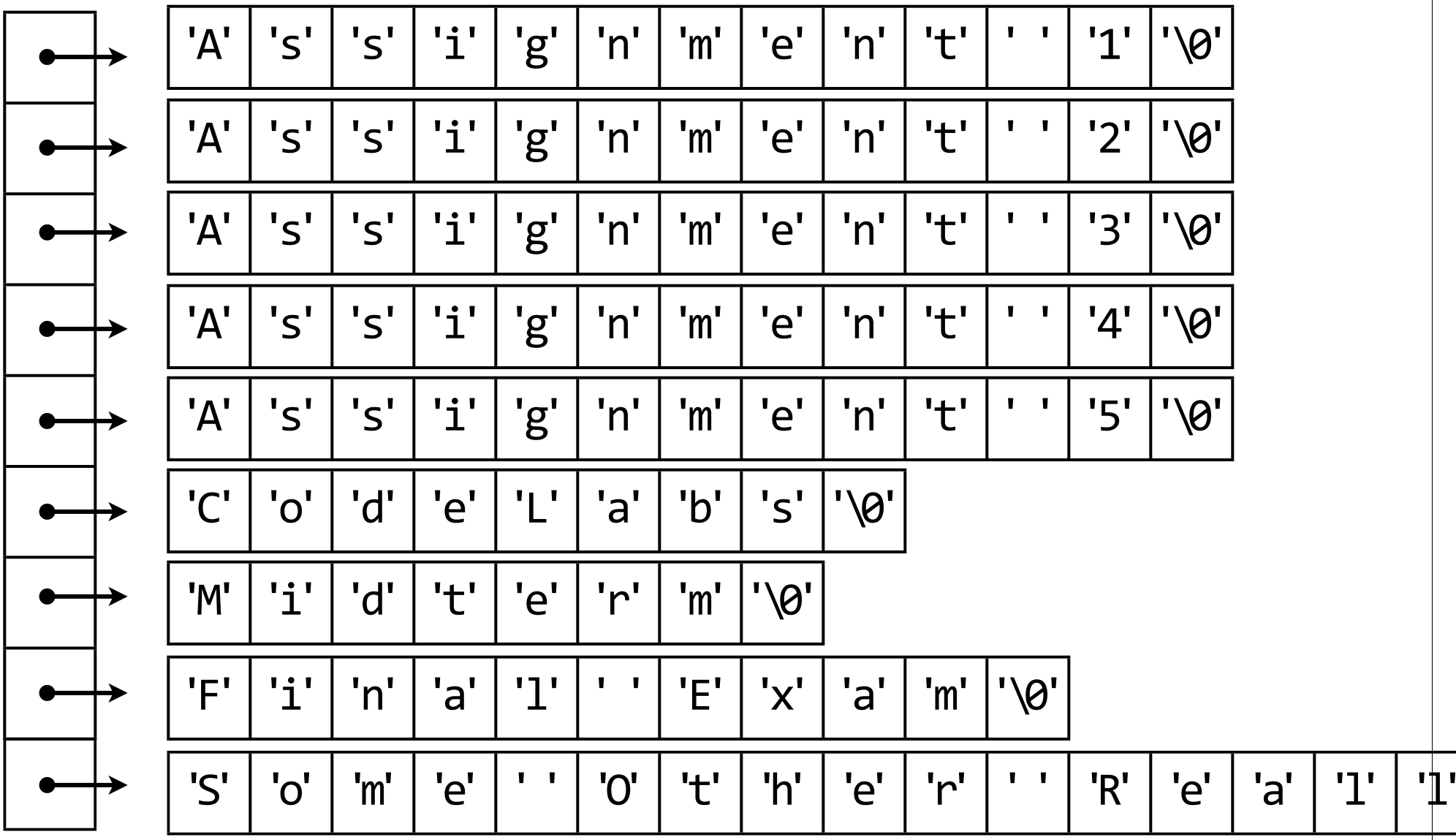
# Arrays of Strings

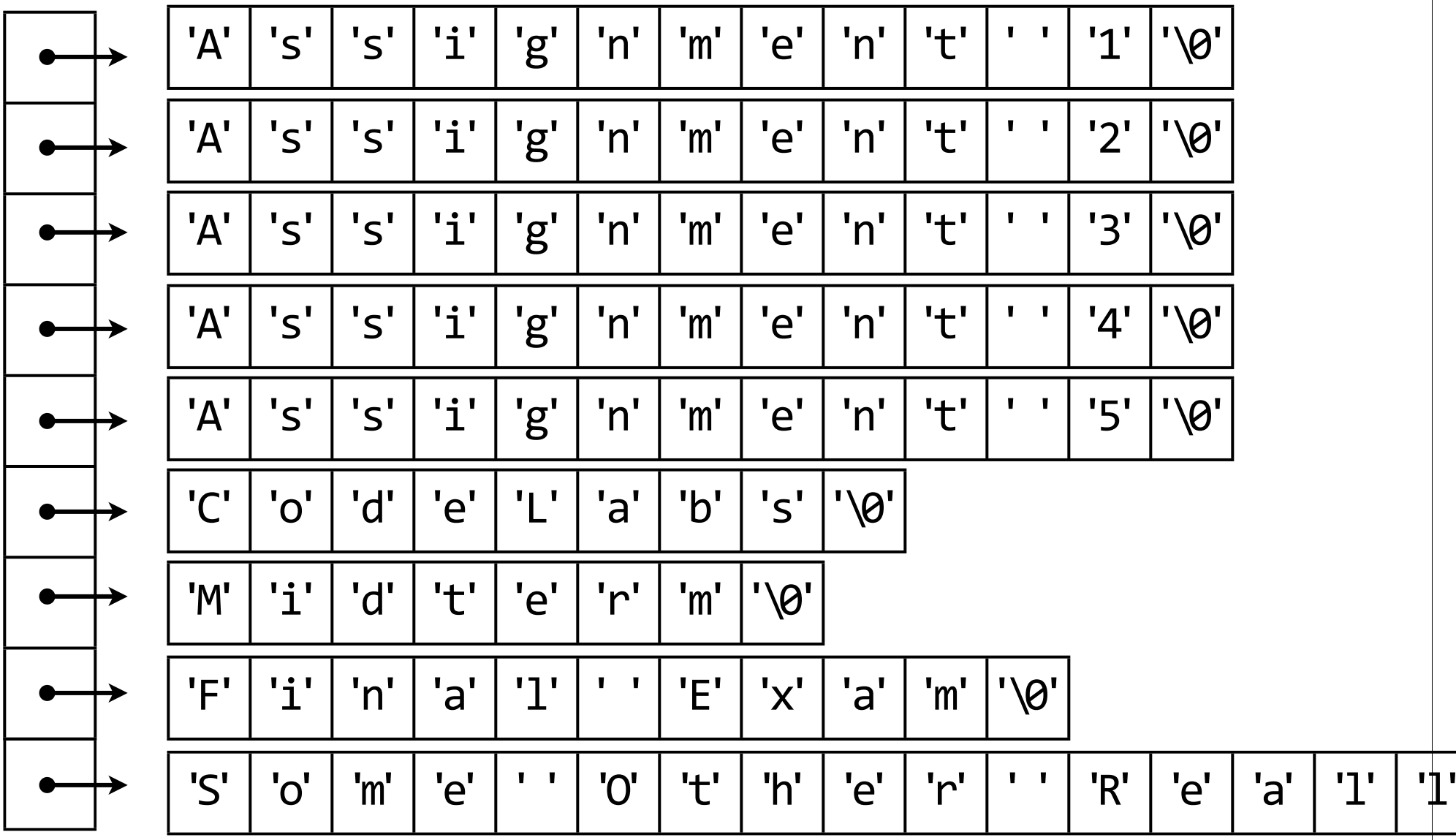
| 'A' | 's' | 's' | 'i' | 'g' | 'n' | 'm' | 'e' | 'n' | 't' | ' ' | '1' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 'A' | 's' | 's' | 'i' | 'g' | 'n' | 'm' | 'e' | 'n' | 't' | ' ' | '2' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' |
| 'A' | 's' | 's' | 'i' | 'g' | 'n' | 'm' | 'e' | 'n' | 't' | ' ' | '3' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' |
| 'A' | 's' | 's' | 'i' | 'g' | 'n' | 'm' | 'e' | 'n' | 't' | ' ' | '4' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' |
| 'A' | 's' | 's' | 'i' | 'g' | 'n' | 'm' | 'e' | 'n' | 't' | ' ' | '5' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' |
| 'C' | 'o' | 'd' | 'e' | 'L' | 'a' | 'b' | 's' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' |
| 'M' | 'i' | 'd' | 't' | 'e' | 'r' | 'm' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' |
| 'F' | 'i' | 'n' | 'a' | 'l' | ' ' | 'E' | 'x' | 'a' | 'm' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' | '\0' |
| 'S' | 'o' | 'm' | 'e' | ' ' | 'O' | 't' | 'h' | 'e' | 'r' | ' ' | 'R' | 'e' | 'a' | 'l' | 'l' | 'y' | ' ' | 'L' | 'o' | 'n' | 'g' | ' ' | 'I' | 't' | 'e' | 'm' | '\0' |


'A' 's' 's' 'i' 'g' 'n' 'm' 'e' 'n' 't' ' ' '1' '\0'
'A' 's' 's' 'i' 'g' 'n' 'm' 'e' 'n' 't' ' ' '2' '\0'
'A' 's' 's' 'i' 'g' 'n' 'm' 'e' 'n' 't' ' ' '3' '\0'
'A' 's' 's' 'i' 'g' 'n' 'm' 'e' 'n' 't' ' ' '4' '\0'
'A' 's' 's' 'i' 'g' 'n' 'm' 'e' 'n' 't' ' ' '5' '\0'
'C' 'o' 'd' 'e' 'L' 'a' 'b' 's' '\0'
'M' 'i' 'd' 't' 'e' 'r' 'm' '\0'
'F' 'i' 'n' 'a' 'l' ' ' 'E' 'x' 'a' 'm' '\0'


'A' 's' 's' 'i' 'g' 'n' 'm' 'e' 'n' 't' ' ' '1' '\0'
'A' 's' 's' 'i' 'g' 'n' 'm' 'e' 'n' 't' ' ' '2' '\0'
'A' 's' 's' 'i' 'g' 'n' 'm' 'e' 'n' 't' ' ' '3' '\0'
'A' 's' 's' 'i' 'g' 'n' 'm' 'e' 'n' 't' ' ' '4' '\0'
'A' 's' 's' 'i' 'g' 'n' 'm' 'e' 'n' 't' ' ' '5' '\0'
'C' 'o' 'd' 'e' 'L' 'a' 'b' 's' '\0'
'M' 'i' 'd' 't' 'e' 'r' 'm' '\0'
'F' 'i' 'n' 'a' 'l' ' ' 'E' 'x' 'a' 'm' '\0'
'S' 'o' 'm' 'e' ' ' 'O' 't' 'h' 'e' 'r' ' ' 'R' 'e' 'a' 'l' 'l'


'A' 's' 's' 'i' 'g' 'n' 'm' 'e' 'n' 't' ' ' '1' '\0'
'A' 's' 's' 'i' 'g' 'n' 'm' 'e' 'n' 't' ' ' '2' '\0'
'A' 's' 's' 'i' 'g' 'n' 'm' 'e' 'n' 't' ' ' '3' '\0'
'A' 's' 's' 'i' 'g' 'n' 'm' 'e' 'n' 't' ' ' '4' '\0'
'A' 's' 's' 'i' 'g' 'n' 'm' 'e' 'n' 't' ' ' '5' '\0'
'C' 'o' 'd' 'e' 'L' 'a' 'b' 's' '\0'
'M' 'i' 'd' 't' 'e' 'r' 'm' '\0'
'F' 'i' 'n' 'a' 'l' ' ' 'E' 'x' 'a' 'm' '\0'
'S' 'o' 'm' 'e' ' ' 'O' 't' 'h' 'e' 'r' ' ' 'R' 'e' 'a' 'l' 'l'

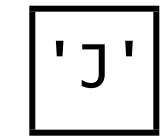

'A' 's' 's' 'i' 'g' 'n' 'm' 'e' 'n' 't' ' ' '1' '\0'
'A' 's' 's' 'i' 'g' 'n' 'm' 'e' 'n' 't' ' ' '2' '\0'
'A' 's' 's' 'i' 'g' 'n' 'm' 'e' 'n' 't' ' ' '3' '\0'
'A' 's' 's' 'i' 'g' 'n' 'm' 'e' 'n' 't' ' ' '4' '\0'
'A' 's' 's' 'i' 'g' 'n' 'm' 'e' 'n' 't' ' ' '5' '\0'
'C' 'o' 'd' 'e' 'L' 'a' 'b' 's' '\0'
'M' 'i' 'd' 't' 'e' 'r' 'm' '\0'
'F' 'i' 'n' 'a' 'l' ' ' 'E' 'x' 'a' 'm' '\0'
'S' 'o' 'm' 'e' ' ' 'O' 't' 'h' 'e' 'r' ' ' 'R' 'e' 'a' 'l' 'l'



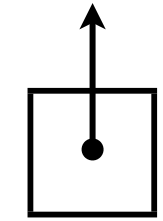

array of pointers to `char`
i.e., an array of `char *`

# Arrays of Pointers

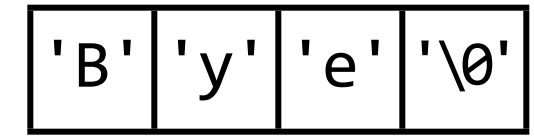
```
char c;
```




```
char *p;
```



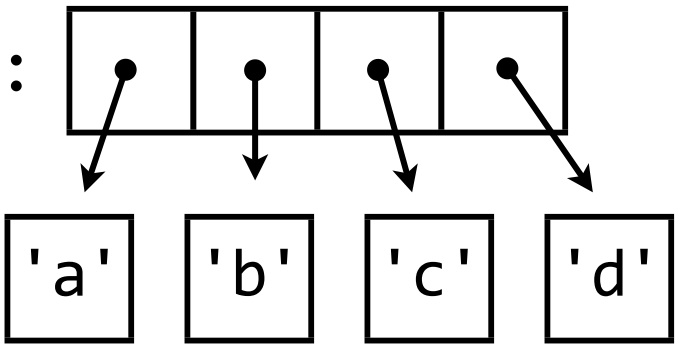

```
char a[4];
```



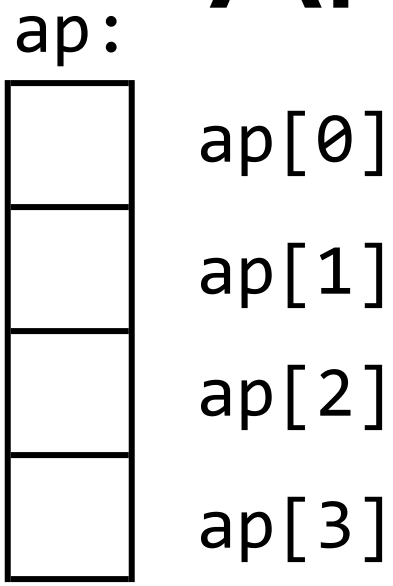

array of something

```
char *ap[4];
```

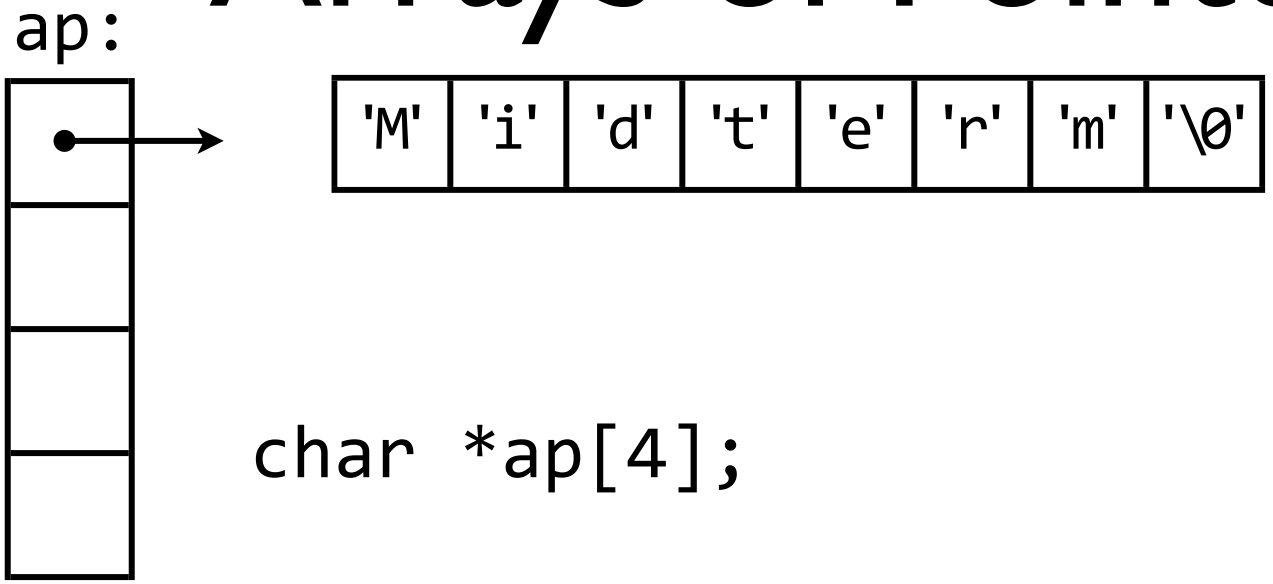
what "something" is



# Arrays of Pointers

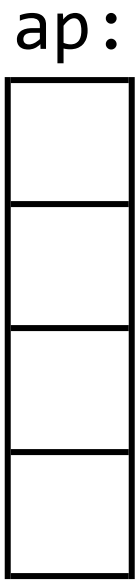


# Arrays of Pointers




```
char *ap[4];
```

# Arrays of Pointers

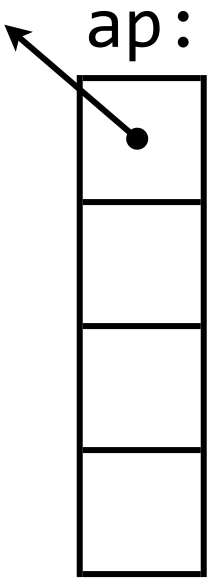



```
char *ap[4];
strncpy(ap[0], "Midterm", ? );
```

Wrong

ap[0] doesn't point at an array!

# Arrays of Pointers



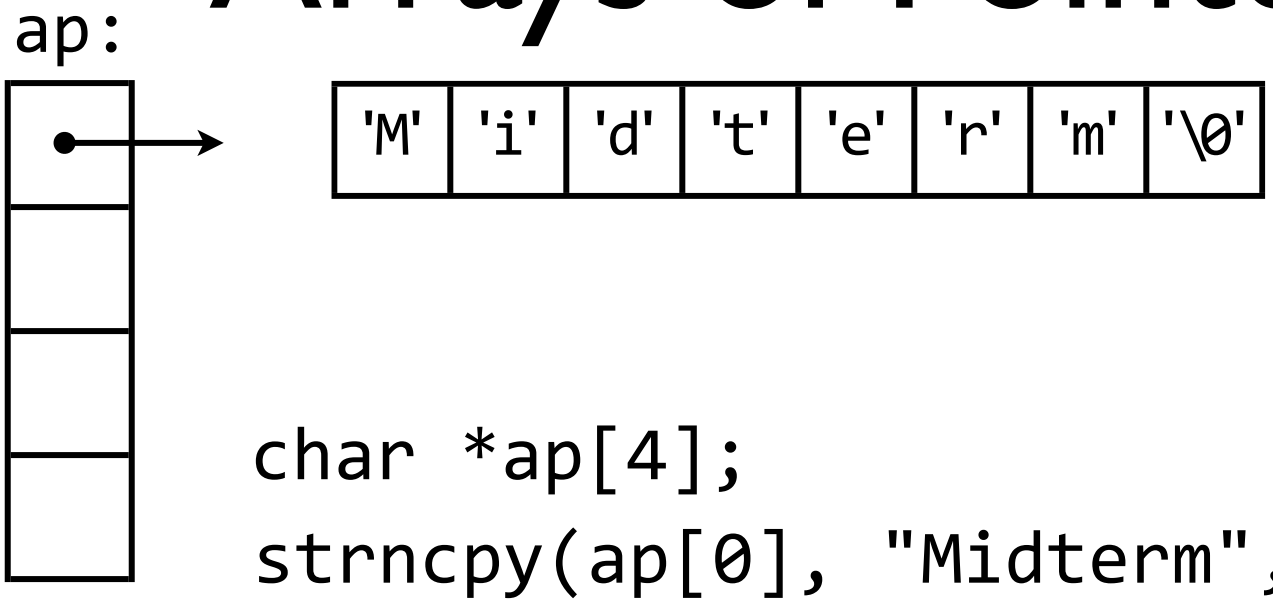

```
int *p;
*p = 12;
```

```
char *ap[4];
strncpy(ap[0], "Midterm", ? );
```

Wrong

ap[0] doesn't point at an array!

# Arrays of Pointers



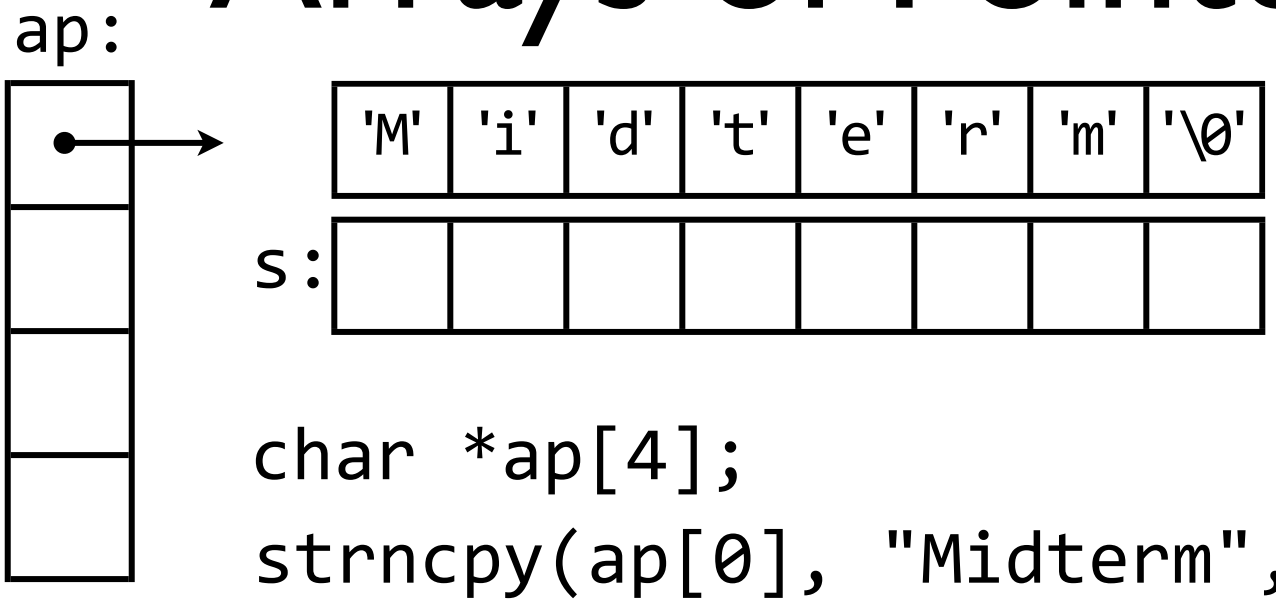

```
char *ap[4];
strncpy(ap[0], "Midterm", ? );
```

Wrong

ap[0] doesn't point at an array!

```
ap[0] = "Midterm";
```

# Arrays of Pointers




```
char *ap[4];
strncpy(ap[0], "Midterm", ? );
```

Wrong

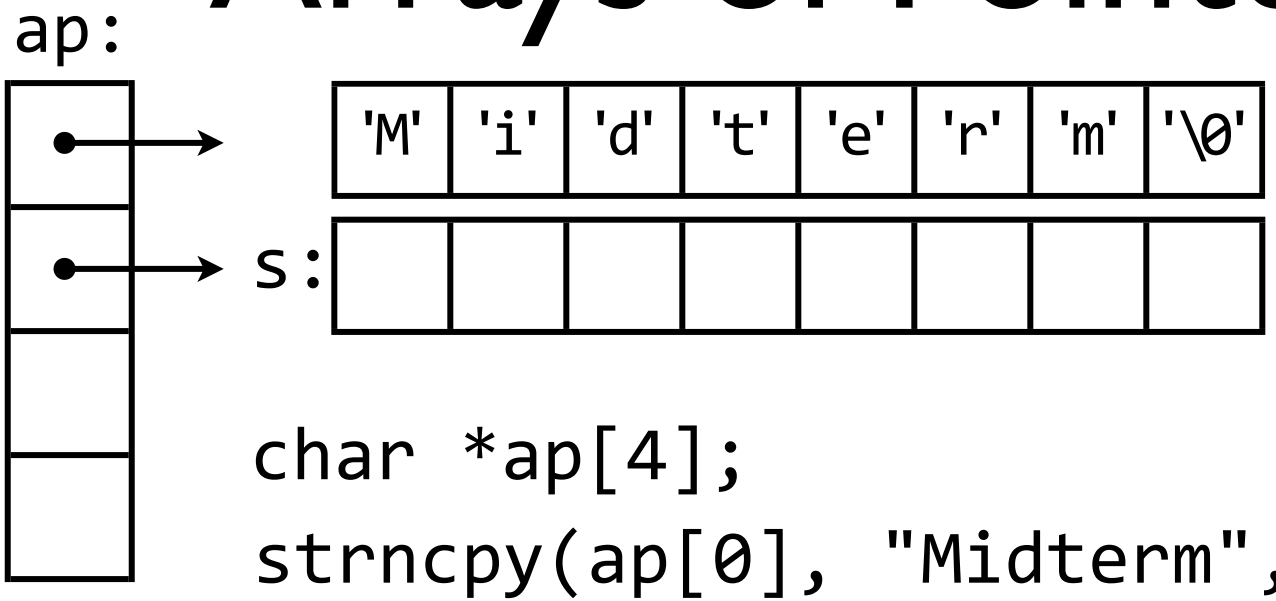
ap[0] doesn't point at an array!

```
ap[0] = "Midterm";
```

Error

```
char s[7 + 1];
s = "Midterm";      &s[0] = "Midterm";
```

# Arrays of Pointers




```
char *ap[4];
strncpy(ap[0], "Midterm", ? );
```

Wrong

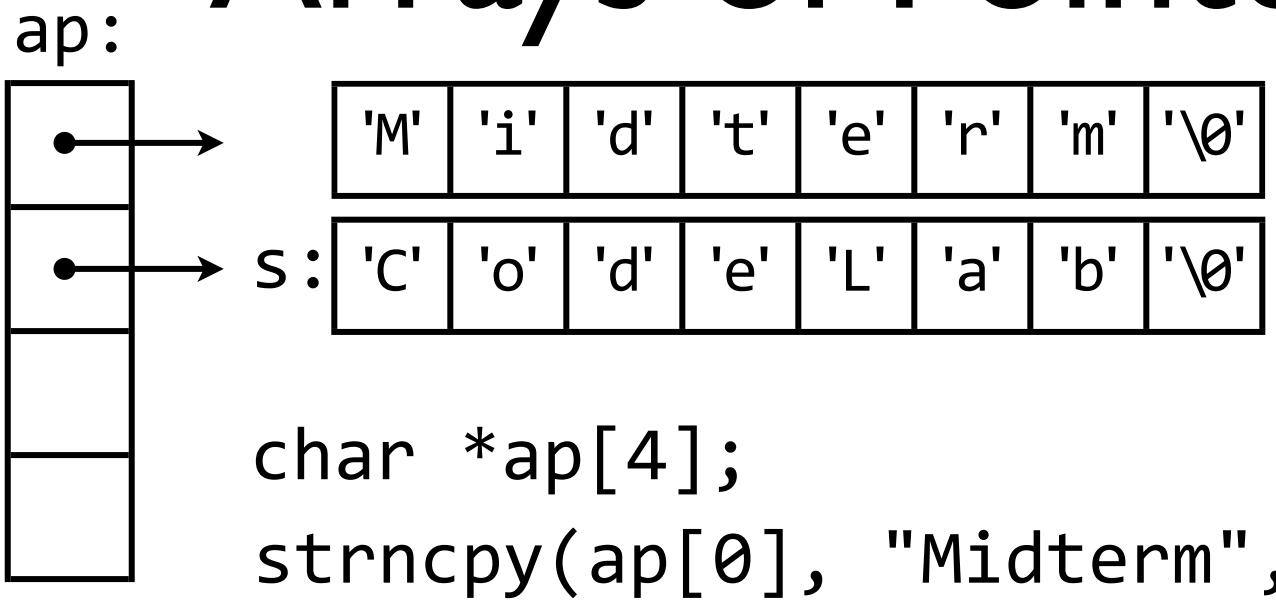
ap[0] doesn't point at an array!

```
ap[0] = "Midterm";
```

Error

```
char s[7 + 1];
s = "Midterm";        &s[0] = "Midterm";
ap[1] = s;
```

# Arrays of Pointers




```
char *ap[4];
strncpy(ap[0], "Midterm", ? );
```

Wrong

ap[0] doesn't point at an array!

```
ap[0] = "Midterm";
```

Error

```
char s[7 + 1];
s = "Midterm";        &s[0] = "Midterm";
ap[1] = s;
strncpy(ap[1], "CodeLab", 7 + 1);
```

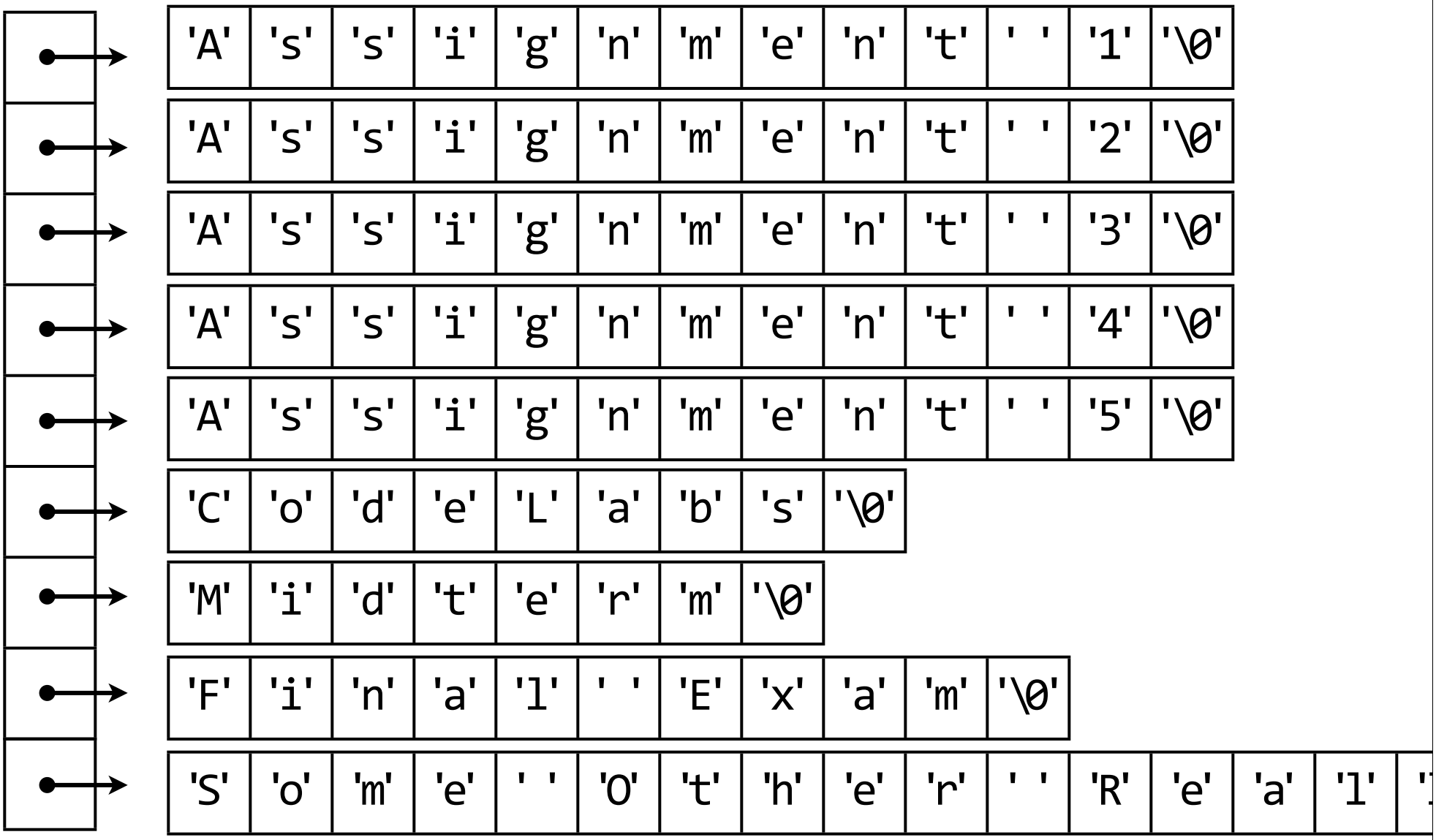
# Arrays of Pointers

```
char marks1[][12+1] = {"Assignment 1", "Assignment 2",
                       "Assignment 3", "Assignment 4",
                       "Assignment 5", "CodeLabs",
                       "Midterm", "Final Exam"};
```

marks1 is an array of arrays of char

```
char *marks2[] = {"Assignment 1", "Assignment 2",
                  "Assignment 3", "Assignment 4",
                  "Assignment 5", "CodeLabs",
                  "Midterm", "Final Exam"};
```

marks2 is an array of pointers to char




`marks2[0]` is the first pointer

```
char marks1[][12+1] = {"Assignment 1", "Assignment 2",
                       "Assignment 3", "Assignment 4",
                       "Assignment 5", "CodeLabs",
                       "Midterm", "Final Exam"};
```

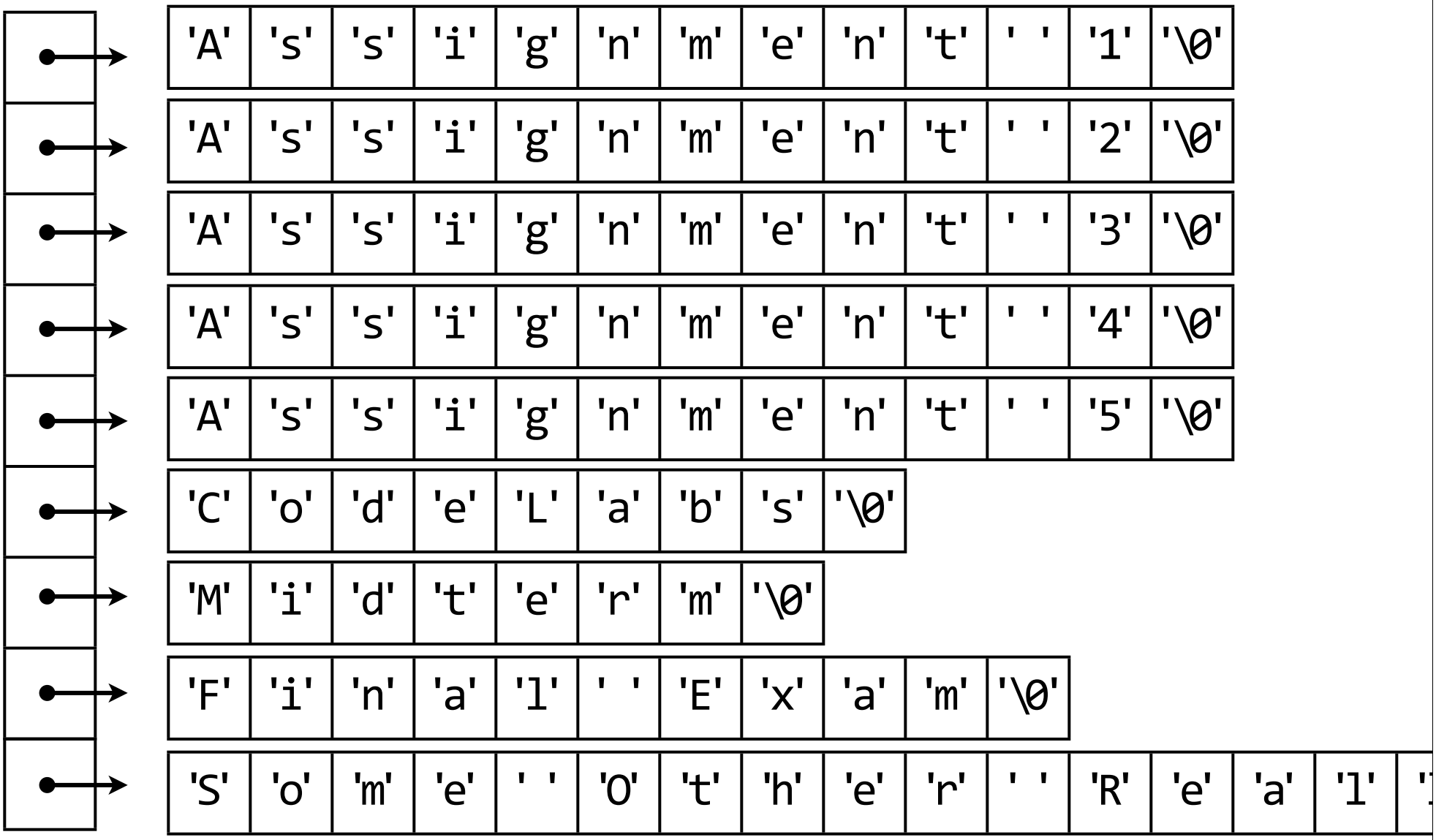
Want first letter of first string: `marks1[0][0]`

Want second letter of first string: `marks1[0][1]`

```
char *marks2[] = {"Assignment 1", "Assignment 2",
                  "Assignment 3", "Assignment 4",
                  "Assignment 5", "CodeLabs",
                  "Midterm", "Final Exam"};
```

Want first letter of first string: `*(marks2[0])`

Want second letter of first string: `*(marks2[0] + 1)`




`marks2[0]` is the first pointer

`*(marks2[0])` is the char in that string

```
char marks1[][12+1] = {"Assignment 1", "Assignment 2",
                       "Assignment 3", "Assignment 4",
                       "Assignment 5", "CodeLabs",
                       "Midterm", "Final Exam"};
```

Want first letter of first string: `marks1[0][0]`

Want second letter of first string: `marks1[0][1]`

```
char *marks2[] = {"Assignment 1", "Assignment 2",
                  "Assignment 3", "Assignment 4",
                  "Assignment 5", "CodeLabs",
                  "Midterm", "Final Exam"};
```

Want first letter of first string: `*(marks2[0])`

Want second letter of first string: `*(marks2[0] + 1)`

```
char marks1[][12+1] = {"Assignment 1", "Assignment 2",
                       "Assignment 3", "Assignment 4",
                       "Assignment 5", "CodeLabs",
                       "Midterm", "Final Exam"};
```

Want first letter of first string: `marks1[0][0]`

Want second letter of first string: `marks1[0][1]`

```
char *marks2[] = {"Assignment 1", "Assignment 2",
                  "Assignment 3", "Assignment 4",
                  "Assignment 5", "CodeLabs",
                  "Midterm", "Final Exam"};
```

Want first letter of first string: `marks2[0][0]`

Want second letter of first string: `marks2[0][1]`

# A Pointer Oddity

```
int a[] = {1, 2, 3};
int *p = a;
```

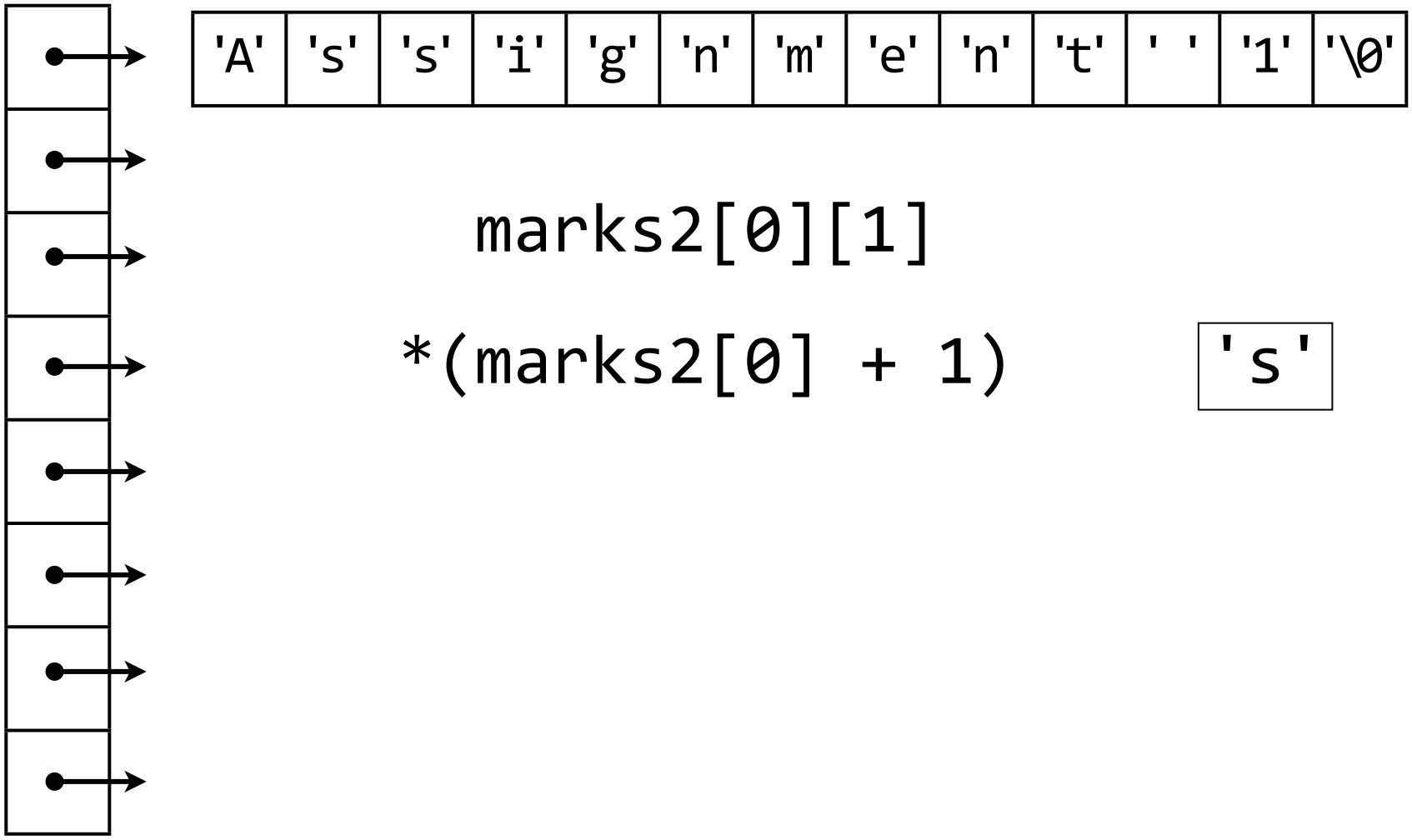
| | |
|---|---|
| `*(p)` `*(&a[0])` | `a[0]` |
| `*(p + 1)` | `a[1]` |
| `*(p + i)` | `a[i]` |
| `*(a + i)` | `a[i]` |
| `*(&a[0] + i)` | `a[i]` |
| `p[i]` | `*(p + i)` |
| | `a[i]` |

**Crazy C Aside**

```
a[i]
*(a + i)
*(i + a)
i[a]
1[a]
```

2

Never actually do this!




marks2[0][1]

*(marks2[0] + 1)  's'

# Arrays of Pointers

- This is not limited to strings
- Sometimes called "ragged arrays" or "sparse arrays"
- Pro:
  - Doesn't waste space
- Con:
  - Some tasks are more complicated (e.g., accessing each element)

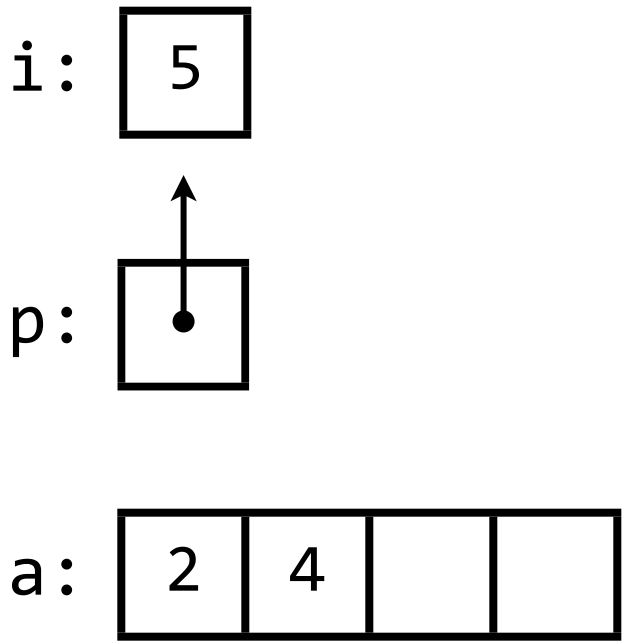
# Dynamic Memory Allocation

`malloc()` and friends

# Automatic Allocation

- We've been doing this since the first week
- Variables have a name and an address

```
int i;
i = 5;

int *p = &i;

int a[4];
a[0] = 2;
a[1] = 4;
```

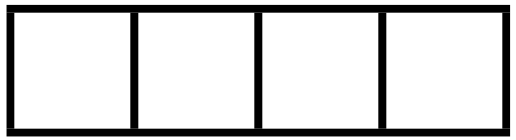


# Fixed Sizes

- Automatically allocated variables can't change size
- Until C99, array sizes needed to be fixed at compile time

Pre-C99

```
int a[4];
```




C99

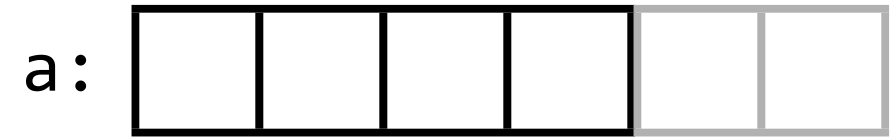
```
int n;
scanf("%d", &n);
int a[n];
```

# Fixed Sizes

- Automatically allocated variables can't change size
- Until C99, array sizes needed to be fixed at compile time

Pre-C99

```
int a[4];
```

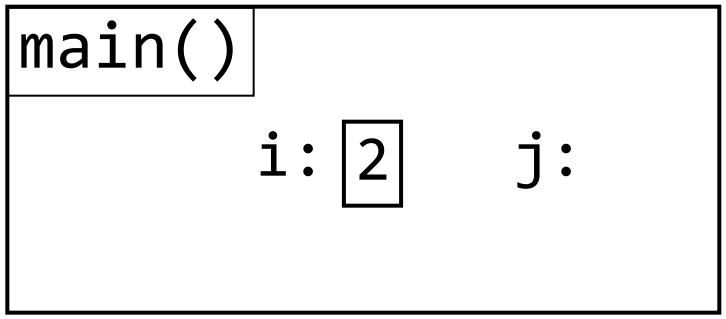



C99

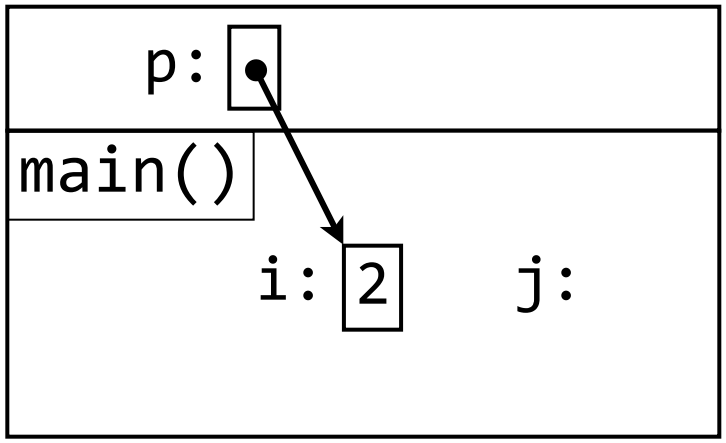
```
int n;
scanf("%d", &n);
int a[n];
```

```
int main(void)
{
    char *lines[N];

    for (int i = 0; i < N; i++)
    {
        printf("$ ");
        lines[i] = getLine();
    }

    for (int i = 0; i < N; i++)
    {
        printf("<<%s>>\n", lines[i]);
    }
    return 0;
}
char *getLine(void)
{
    char input[MAX_LENGTH + 1];
    fgets(input, MAX_LENGTH + 1, stdin);

    return input;
}
```

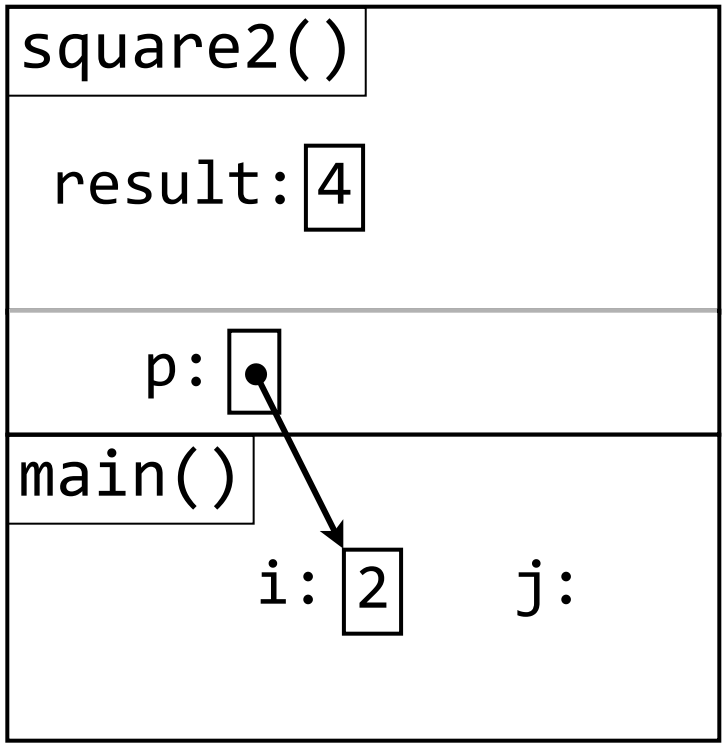
```
int *square2(int *p)
{
  int result = *p * *p;
  return &result;
}

int main (void)
{
  int i = 2;
  int *j = square2(&i);
  printf("%d", *j);

  return 0;
}
```



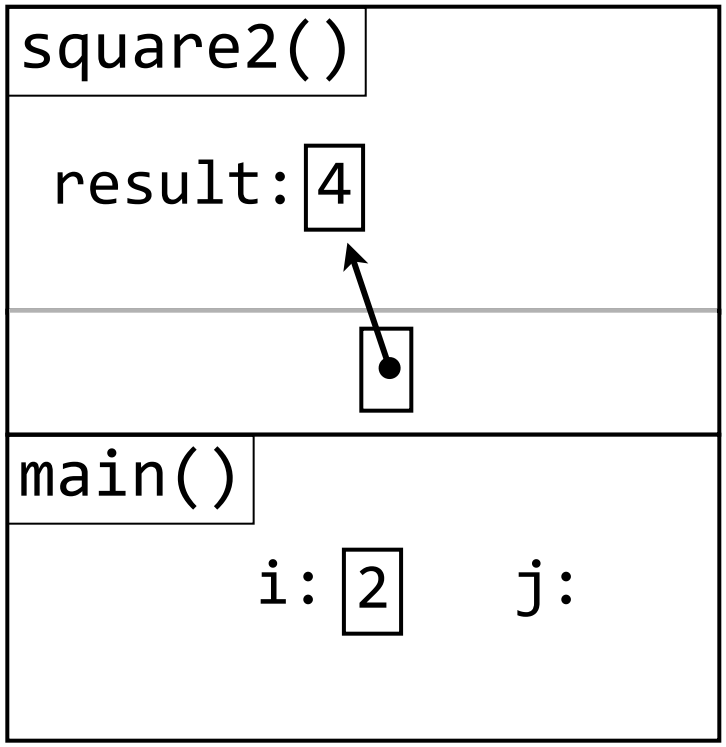
```
int *square2(int *p)
{
  int result = *p * *p;
  return &result;
}

int main (void)
{
  int i = 2;
  int *j = square2(&i);
  printf("%d", *j);

  return 0;
}
```



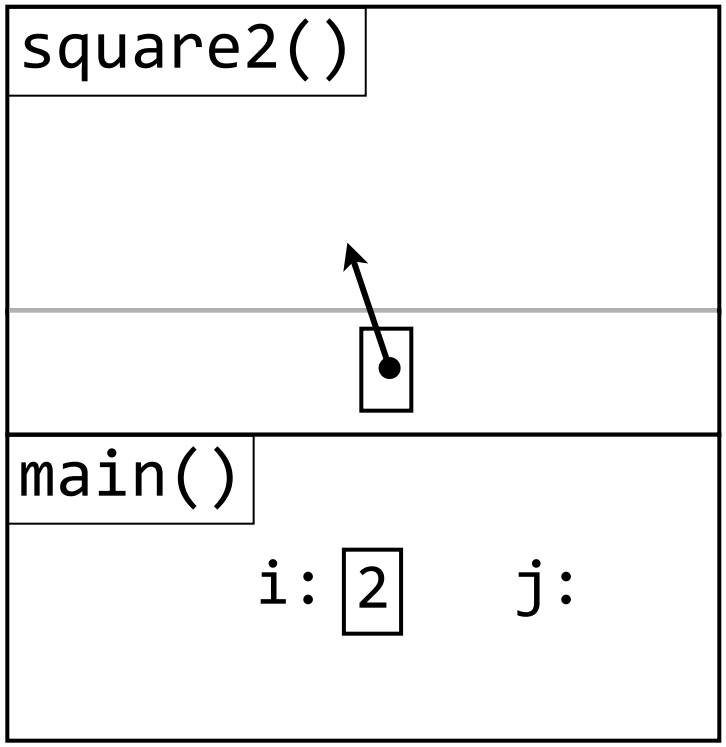
```
int *square2(int *p)
{
  int result = *p * *p;
  return &result;
}

int main (void)
{
  int i = 2;
  int *j = square2(&i);
  printf("%d", *j);

  return 0;
}
```



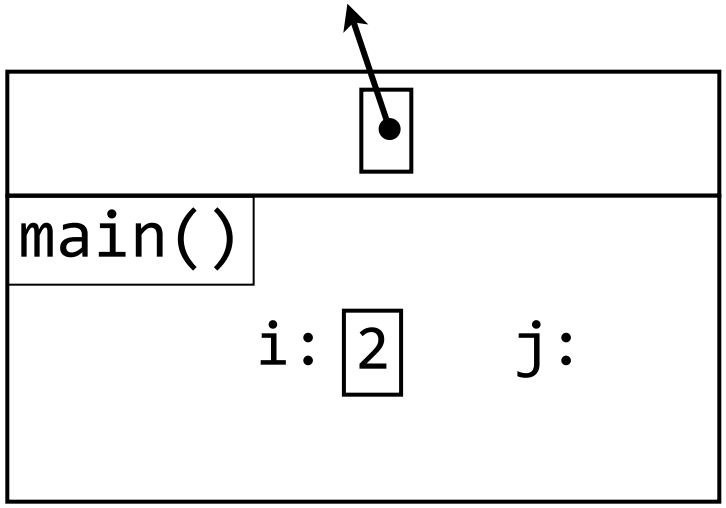
```
int *square2(int *p)
{
  int result = *p * *p;
  return &result;
}

int main (void)
{
  int i = 2;
  int *j = square2(&i);
  printf("%d", *j);

  return 0;
}
```



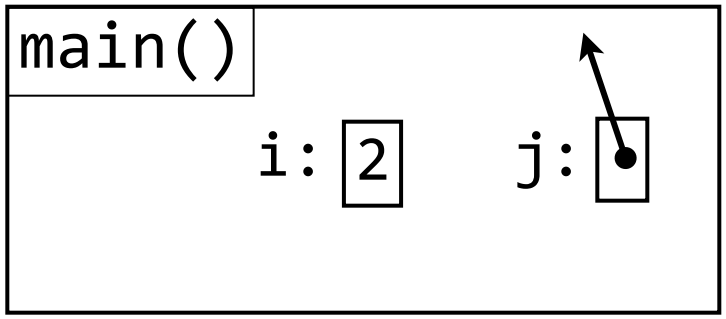
```
int *square2(int *p)
{
  int result = *p * *p;
  return &result;
}

int main (void)
{
  int i = 2;
  int *j = square2(&i);
  printf("%d", *j);

  return 0;
}
```



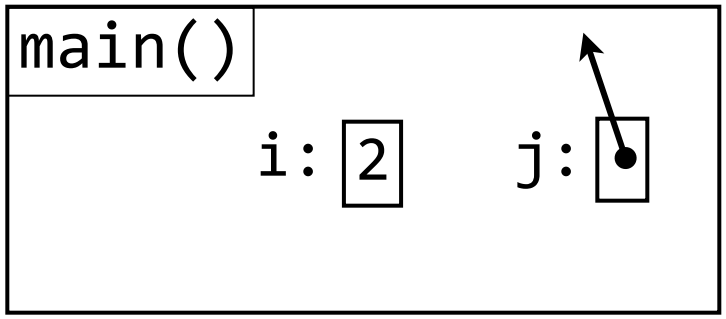
```
int *square2(int *p)
{
  int result = *p * *p;
  return &result;
}

int main (void)
{
  int i = 2;
  int *j = square2(&i);
  printf("%d", *j);

  return 0;
}
```



```
int *square2(int *p)
{
  int result = *p * *p;
  return &result;
}

int main (void)
{
  int i = 2;
  int *j = square2(&i);
  printf("%d", *j);

  return 0;
}
```



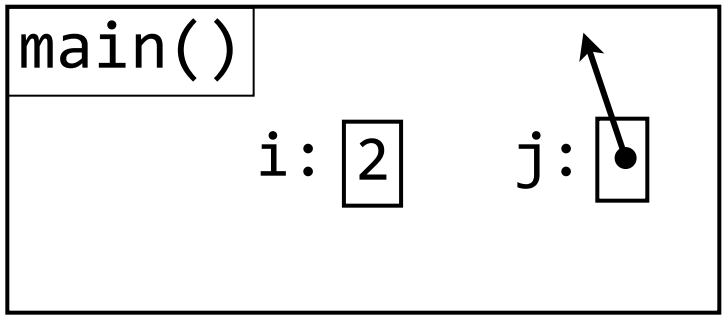
```
int *square2(int *p)
{
  int result = *p * *p;
  return &result;
}

int main (void)
{
  int i = 2;
  int *j = square2(&i);
  printf("%d", *j);
                  ????
  return 0;
}
```



```
int *square2(int *p)
{
  int result = *p * *p;
  return &result;
}

int main (void)
{
  int i = 2;
  int *j = square2(&i);
  printf("%d", *j);

  return 0;
}
```

Wrong

????



```
int main(void)
{
    char *lines[N];

    for (int i = 0; i < N; i++)
    {
        printf("$ ");
        lines[i] = getLine();
    }

    for (int i = 0; i < N; i++)
    {
        printf("<<%s>>\n", lines[i]);
    }
    return 0;
}
char *getLine(void)
{
    char input[MAX_LENGTH + 1];
    fgets(input, MAX_LENGTH + 1, stdin);

    return input;
}
```

Wrong

# Dynamic Memory Allocation

- Alternate way of allocating memory
- Obtained from a different pool of memory

# malloc()

```
void *malloc(size_t size);
```

for now think of it as int

- "Memory Allocator"
- `#include <stdlib.h>`
- You ask `malloc()` for some memory, it finds some, and then gives it to you
- `void *` pointer ("generic" pointer)