2D Array, Struct

Zhaoguo Wang
Multi-dimensional arrays

Declare a k dimensional array

```c
int arr[n_1][n_2][n_3]...[n_{k-1}][n_k]
```

$n_i$ is the length of the $i$th dimension
Multi-dimensional arrays

Declare a k dimensional array

\[
\text{int arr}[n_1][n_2][n_3]...[n_{k-1}][n_k]
\]

\(n_i\) is the length of the \(i^{th}\) dimension

Example: 2D array

\[
\text{int matrix}[2][3] = \{\{1, 2, 3\}, \{4, 5, 6\}\};
\]
Multi-dimensional arrays

Declare a \( k \) dimensional array

\[
\text{int } \text{arr}[n_1][n_2][n_3] \ldots [n_{k-1}][n_k]
\]

\( n_i \) is the length of the \( i \)th dimension

Example: 2D array

\[
\text{int } \text{matrix}[2][3] = \{\{1, 2, 3\}, \{4, 5, 6\}\};
\]
Multi-dimensional arrays

Declare a $k$ dimensional array

$$\text{int arr}[n_1][n_2][n_3]...[n_{k-1}][n_k]$$

$n_i$ is the length of the $i$th dimension

Example: 2D array

$$\text{int matrix}[2][3] = \{\{1, 2, 3\}, \{4, 5, 6\}\};$$

Access an element at second row and third column

$$\text{matrix}[1][2] = 100$$
Memory layout

```c
char name[2][3] = {{'t', 'i', 'g'}, {'e', 'r', '!'}};

for(int i = 0; i < 2; i++) {
    for(int j = 0; j < 3; j++) {
        printf("%lx\n", (unsigned long)&c[i][j]);
    }
}
```
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>matrix[0][0]</td>
<td>0x100</td>
<td></td>
</tr>
<tr>
<td>matrix[0][1]</td>
<td>0x104</td>
<td></td>
</tr>
<tr>
<td>matrix[0][2]</td>
<td>0x108</td>
<td></td>
</tr>
<tr>
<td>matrix[1][0]</td>
<td>0x10c</td>
<td></td>
</tr>
<tr>
<td>matrix[1][1]</td>
<td>0x110</td>
<td></td>
</tr>
<tr>
<td>matrix[1][2]</td>
<td>0x114</td>
<td></td>
</tr>
</tbody>
</table>

0x400
# Memory Layout

<table>
<thead>
<tr>
<th>Memory Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01000</td>
<td>0x100</td>
</tr>
<tr>
<td>0x01004</td>
<td>0x104</td>
</tr>
<tr>
<td>0x01008</td>
<td>0x108</td>
</tr>
<tr>
<td>0x01010</td>
<td>0x110</td>
</tr>
<tr>
<td>0x01014</td>
<td>0x114</td>
</tr>
<tr>
<td>0x01018</td>
<td>0x10c</td>
</tr>
<tr>
<td>0x01020</td>
<td>0x400</td>
</tr>
<tr>
<td>0x01024</td>
<td>0x108</td>
</tr>
<tr>
<td>0x01028</td>
<td>0x104</td>
</tr>
<tr>
<td>0x01030</td>
<td>0x100</td>
</tr>
</tbody>
</table>

The first row contains the elements of a matrix:

- `matrix[0][0]`: 1
- `matrix[0][1]`: 2
- `matrix[0][2]`: 3

The second row contains:

- `matrix[1][0]`: 4
- `matrix[1][1]`: 5
- `matrix[1][2]`: 6
## Memory Layout

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>matrix[1][2]</td>
<td>6</td>
<td>0x10c</td>
<td>0x114</td>
</tr>
<tr>
<td>matrix[1][1]</td>
<td>5</td>
<td>0x10c</td>
<td>0x110</td>
</tr>
<tr>
<td>matrix[1][0]</td>
<td>4</td>
<td>0x10c</td>
<td>0x108</td>
</tr>
<tr>
<td>matrix[0][2]</td>
<td>3</td>
<td>0x10c</td>
<td>0x104</td>
</tr>
<tr>
<td>matrix[0][1]</td>
<td>2</td>
<td>0x10c</td>
<td></td>
</tr>
<tr>
<td>matrix[0][0]</td>
<td>1</td>
<td>0x10c</td>
<td></td>
</tr>
</tbody>
</table>

**1st row:**
- matrix[0][0]: 1
- matrix[0][1]: 2
- matrix[0][2]: 3

**2nd row:**
- matrix[1][0]: 4
- matrix[1][1]: 5
- matrix[1][2]: 6
What are the values of matrix[0] and matrix[1]?

<table>
<thead>
<tr>
<th></th>
<th>matrix[0][0]</th>
<th>matrix[0][1]</th>
<th>matrix[0][2]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st row</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2nd row</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

0x100 - 0x10c 0x104 - 0x108 0x110 - 0x114 0x110 - 0x114 0x400
## Pointers

<table>
<thead>
<tr>
<th>1st row</th>
<th>2nd row</th>
</tr>
</thead>
<tbody>
<tr>
<td>matrix[0][0]</td>
<td>1 0x100</td>
</tr>
<tr>
<td>matrix[0][1]</td>
<td>2 0x104</td>
</tr>
<tr>
<td>matrix[0][2]</td>
<td>3 0x108</td>
</tr>
<tr>
<td>matrix[1][0]</td>
<td>4 0x10c</td>
</tr>
<tr>
<td>matrix[1][1]</td>
<td>5 0x110</td>
</tr>
<tr>
<td>matrix[1][2]</td>
<td>6 0x114</td>
</tr>
</tbody>
</table>

- matrix[0]: 0x100
- matrix[1]: 0x10c
What are the values of &matrix[0] and &matrix[1]?

<table>
<thead>
<tr>
<th>1st row</th>
<th></th>
<th>2nd row</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>matrix[0][0]</td>
<td>1</td>
<td>matrix[1][0]</td>
<td>4</td>
</tr>
<tr>
<td>matrix[0][1]</td>
<td>2</td>
<td>matrix[1][1]</td>
<td>5</td>
</tr>
<tr>
<td>matrix[0][2]</td>
<td>3</td>
<td>matrix[1][2]</td>
<td>6</td>
</tr>
</tbody>
</table>

Values:
- &matrix[0] = 0x100
- &matrix[1] = 0x10C

Note: The values 0x100, 0x104, 0x108, 0x110, 0x114, and 0x400 correspond to the addresses indicated in the table.
### Pointers

- &matrix[0]: 0x100
- &matrix[1]: 0x10c

<table>
<thead>
<tr>
<th>1st row</th>
<th>2nd row</th>
</tr>
</thead>
<tbody>
<tr>
<td>matrix[0][0]</td>
<td>1</td>
</tr>
<tr>
<td>matrix[0][1]</td>
<td>2</td>
</tr>
<tr>
<td>matrix[0][2]</td>
<td>3</td>
</tr>
<tr>
<td>matrix[1][0]</td>
<td>4</td>
</tr>
<tr>
<td>matrix[1][1]</td>
<td>5</td>
</tr>
<tr>
<td>matrix[1][2]</td>
<td>6</td>
</tr>
</tbody>
</table>

...
Pointers

Compiler convert the two dimensional array into one dimension.

<table>
<thead>
<tr>
<th>1st row</th>
<th>2nd row</th>
</tr>
</thead>
<tbody>
<tr>
<td>matrix[0][0]</td>
<td>1</td>
</tr>
<tr>
<td>matrix[0][1]</td>
<td>2</td>
</tr>
<tr>
<td>matrix[0][2]</td>
<td>3</td>
</tr>
<tr>
<td>matrix[1][0]</td>
<td>4</td>
</tr>
<tr>
<td>matrix[1][1]</td>
<td>5</td>
</tr>
<tr>
<td>matrix[1][2]</td>
<td>6</td>
</tr>
</tbody>
</table>

... | ... | 0x400 |

...
Compiler convert the two dimensional array into one dimension.

matrix[0] and matrix[1] are not physically exist.
What is the value of `matrix`?
Pointers

matrix is 0x100

1st row

matrix[0][0] 1 0x100
matrix[0][1] 2 0x104
matrix[0][2] 3 0x108

2nd row

matrix[1][0] 4 0x10c
matrix[1][1] 5 0x110
matrix[1][2] 6 0x114

...
### Pointers

#### 1st row
- matrix[0][0]: 1 (0x100)
- matrix[0][1]: 2 (0x104)
- matrix[0][2]: 3 (0x108)

#### 2nd row
- matrix[1][0]: 4 (0x10c)
- matrix[1][1]: 5 (0x110)
- matrix[1][2]: 6 (0x114)

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>matrix</td>
<td>matrix</td>
<td>matrix</td>
<td>matrix</td>
<td>matrix</td>
</tr>
<tr>
<td>[1]</td>
<td>[1]</td>
<td>[0]</td>
<td>[1]</td>
<td>[0]</td>
</tr>
<tr>
<td>[2]</td>
<td>[1]</td>
<td>[0]</td>
<td>[2]</td>
<td>[0]</td>
</tr>
</tbody>
</table>

**Matrix Dimensions:**
- Width: 3
- Height: 2

**Matrix Representation:**
- `0x100` to `0x114` for the 1st row.
- `0x10c` to `0x110` for the 2nd row.

- `0x400` to `0x400` for the matrix's data range.
### Pointers

How many ways to define a pointer which points to the head of the array?

<table>
<thead>
<tr>
<th>row</th>
<th>matrix[0][0]</th>
<th>matrix[0][1]</th>
<th>matrix[0][2]</th>
<th>matrix[1][0]</th>
<th>matrix[1][1]</th>
<th>matrix[1][2]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st row</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>2nd row</td>
<td>0x100</td>
<td>0x104</td>
<td>0x108</td>
<td>0x10c</td>
<td>0x110</td>
<td>0x114</td>
</tr>
</tbody>
</table>

Matrix:

```
    0x100  0x104  0x108  0x10c  0x110  0x114
  0x400
```
**Pointers**

<table>
<thead>
<tr>
<th>1st row</th>
<th>2nd row</th>
</tr>
</thead>
<tbody>
<tr>
<td>matrix[0][0]</td>
<td>1</td>
</tr>
<tr>
<td>matrix[0][1]</td>
<td>2</td>
</tr>
<tr>
<td>matrix[0][2]</td>
<td>3</td>
</tr>
<tr>
<td>matrix[1][0]</td>
<td>4</td>
</tr>
<tr>
<td>matrix[1][1]</td>
<td>5</td>
</tr>
<tr>
<td>matrix[1][2]</td>
<td>6</td>
</tr>
</tbody>
</table>

```
int *p = &matrix[0][0]
int *p = matrix[0]
int *p = *matrix
```

No compatible error/warnings
How to access matrix[1][0] with p?

int *p = &matrix[0][0]
int *p = matrix[0]
int *p = *matrix
Pointers

```
int *p = &matrix[0][0]
int *p = matrix[0]
int *p = *matrix
```

matrix[0][0]: *(p + 3)
int main() {

    char name[2][3] = {{'t', 'i', 'g'}, {'e', 'r', '!'}};

    for(int i = 0; i < 6; i++) {
        printf("%c\n", *name[i]);
    }
    return 0;
}
int main() {
    char name[2][3] = {{'t', 'i', 'g'}, {'e', 'r', '!'}};
    char *p = name[0];
    for(int i = 0; i < 6; i++) {
        printf("%c\n", p[i]);
    }
    return 0;
}
A general question

Given a 2D array matrix[m][n] and a pointer p which points to matrix[0][0], how to use p to access matrix[i][j]?
A general question

Given a 2D array matrix[m][n] and a pointer p which points to matrix[0][0], how to use p to access matrix[i][j]?

matrix[i][j]: p + i * n + j
Array – a block of n consecutive objects, the objects have the same type.

How to define a block of objects, but each may or may not have a different type?
struct student {
    int id;
    char name[100];
};
struct student {
    int id;
    char name[100];
};

Field 1: a integer
struct student {
    int id;
    char name[100];
};

Field 2: an array
struct student {
    int id;
    char name[100];
};

struct student t;  // define an object with type student
Structure

```c
struct student {
    int id;
    char name[100];
};

struct student t;

t.id = 1024
    t.name[0] = 'z'
    t.name[1] = 'h'
...
```

Access the fields of this object
typedef struct {
    int id;
    char name[100];
} student;

struct student t;

t.id = 1024

t.name[0] = 'z'
t.name[1] = 'h'
...
typedef struct {
    int id;
    char name[100];
} student;
Structure’s size

1st question:
What is the size of structure student?

typedef struct {
    int id;
    char name[100];
} student;
Structure’s size

What is the size of structure A?

typedef struct {
    int id;
} A;
What is the size of structure A?

typedef struct {
    int id;
} A;

Answer: 4
What is the size of structure B?

typedef struct {
    char name[100];
} B;
Structure’s size

What is the size of structure B?

typedef struct {
    char name[100];
} B;

Answer: 100
Structure’s size

1\textsuperscript{st} question:  
What is the size of structure student?

typedef struct {
    int id;
    char name[100];
} student;
Structure’s size

1st question:
What is the size of structure student?

typedef struct {
    int id;
    char name[100];
} student;

Answer: 104
2nd question:
What is the size of structure student?

typedef struct {
    int id;
    char gender;
} student;
Structure’s size

2nd question:
What is the size of structure student?

typedef struct {
    int id;
    char gender;
} student;

Answer: 5 ?
2\textsuperscript{st} question:
What is the size of structure student?

typedef struct {
   int id;
   char gender;
} student;

Answer: 5
2\textsuperscript{nd} question:
What is the size of structure student?

typedef struct {
    int id;
    char gender;
} student;

Answer: 8
Structure’s size

typedef struct {
    int id;
    char gender;
} student;
typedef struct {
   int id;
   char gender;
} student;
typedef struct {  
  int id;  
  char gender;  
} student;
Data structure alignment

Put the data at a memory address equal to some multiple of the word size through the data padding
Data structure alignment

Put the data at a memory address equal to some multiple of the word size through the data padding

CPU reads/writes data from/into memory in word sized chunks. (e.g., 4 bytes chunks on a 32-bit system)

Ensure read each primary type with a single read.
Put the data at a memory address equal to some multiple of the word size through the data padding.

```
readInfo(student s) {
    int id = s.id
    char gender = s.gender
}
```
Data structure alignment

Put the data at a memory address equal to some multiple of the word size through the data padding.

```
readInfo(student s) {
    int id = s.id
    char gender = s.gender
}
```
Problem without data alignment

student s[2];

for(int i = 0; i < 2; i++) {
    readInfo(s[i])
}
Problem without data alignment

```
student s[2];
for(int i = 0; i < 2; i++) {
    readInfo(s[i])
}
readInfo(student s) {
    int id = s.id
    char gender = s.gender
}
```

Memory Layout

![Diagram showing memory layout and data transfer]

1st transfer
Problem without data alignment

```c
student s[2];
for(int i = 0; i < 2; i++) {
    readInfo(s[i])
}
readInfo(student s) {
    s[1] → int id = s.id
    2^{nd} transfer
    char gender = s.gender
    }
```
Problem without data alignment

```c
student s[2];
for(int i = 0; i < 2; i++) {
    readInfo(s[i])
}

readInfo(student s) {
    int id = s.id
    char gender = s.gender
}
```

Performance and correctness issues
Data structure alignment

```
student s[2];
for(int i = 0; i < 2; i++) {
    readInfo(s[i])
}
readInfo(student s) {
    int id = s.id
    char gender = s.gender
}
```
Questions

What is the memory size and memory layout of following structures? (64 bits architecture)

typedef struct {
    int a;
    char b;
    int c;
    char d;
} S_A;

typedef struct {
    int a;
    int b;
    char c;
    char d;
} S_B;

Ensure transfer each primary type with a single read.
typedef struct {
    int a;
    char b;
    int c;
    char d;
} S_A;

1 word
typedef struct {
    int a;
    char b;
    int c;
    char d;
} S_A;

1 word
typedef struct {
    int a;
    char b;
    int c;
    char d;
} S_A;
typedef struct {
    int a;
    char b;
    int c;
    char d;
} S_A;

1 word
typedef struct {
  int a;
  char b;
  int c;
  char d;
} S_A;

1 word
typedef struct {
    int a;
    char b;
    int c;
    char d;
} S_A;
typedef struct {
    int a;
    char b;
    int c;
    char d;
} S_A;

1 word
typedef struct {
    int a;
    int b;
    char c;
    char d;
} S_A;

1 word
typedef struct {
    int a;
    int b;
    char c;
    char d;
} S_A;

1 word
typedef struct {
    int a;
    int b;
    char c;
    char d;
} S_A;
typedef struct {
    int a;
    int b;
    char c;
    char d;
} S_A;

1 word

0x0 0x1 0x2 0x3 0x4 0x5 0x6 0x7 0x8 0x9 0xa 0xb 0xc 0xd 0xe 0xf
typedef struct {
    int id;
    char gender;
} student;

student t = student{1, ‘m’};
student * p = &t;
p->id = 2
Malloc

```c
int a[10];
```

Global variables are allocated space before program execution.

Local variables are allocated space at the entrance of a function/block and deallocated space upon exit.
Malloc

Dynamically allocate a space
- `malloc`: allocate storage of a given size
- `free`: de-allocate previously `malloc`-ed storage

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

A void pointer is a pointer that has no associated data type with it. A void pointer can hold address of any type and can be typcasted to any type.

```c
void free(void *ptr);
```
Malloc

Dynamically allocate a space
− malloc: allocate storage of a given size
− free: de-allocate previously malloc-ed storage

#include <stdlib.h>

int *newArr(int n) {
    int *p = (int*)malloc(sizeof(int) * n);
    return p;
}
Reverse a linked list

```c
struct node {
    int val;
    struct node *next;
};

struct node*
reverseList(struct node* head) {
    // your code here
}
```
Reverse a linked list

head
prev
cur
next
Reverse a linked list:

```
cur->next = prev
```
Reverse a linked list

head
prev cur

next

cur->next = prev
prev = cur
Reverse a linked list

```
cur->next = prev
prev = cur
cur = next
```
Reverse a linked list

cur->next = prev
prev = cur
cur = next
next = cur->next
Reverse a linked list

cur->next = prev
prev = cur
cur = next
next = cur->next
Reverse a linked list

cur->next = prev
prev = cur
cur = next
next = cur->next
Reverse a linked list

cur->next = prev
prev = cur
cur = next
next = cur->next
Reverse a linked list

cur->next = prev
prev = cur
cur = next
next = cur->next
Reverse a linked list

cur->next = prev
prev = cur
cur = next
next = cur->next
Reverse a linked list

cur->next = prev
prev = cur
cur = next
next = cur->next
Reverse a linked list

`cur->next = prev`
`prev = cur`
`cur = next`
`next = cur->next`
Reverse a linked list

cur->next = prev
prev = cur
cur = next
next = cur->next
Reverse a linked list

cur->next = prev
prev = cur
cur = next
next = cur->next
Reverse a linked list

cur->next = prev
prev = cur
cur = next
next = cur->next
Reverse a linked list

```c
struct node {
    int val;
    struct node *next;
};

struct node*
reverseList(struct node* head) {

    node *prev = null;
    node *curr = head;
    while (curr != null) {
        node *next = curr->next;
        curr->next = prev;
        prev = curr;
        curr = next;
    }
    return prev;
}
```
Remove linked list element

```c
struct node {
    int val;
    struct node *next;
};

struct node*
removeElements(struct node* head, int val)
{
    // your code here
}
```

Example
Given: 1 → 2 → 6 → 3 → 6 → 8, val = 6
Return: 1 → 2 → 3
Remove linked list element

check prev->next->val
Remove linked list element

check prev->next->val
if prev->next->val == val {
    prev->next->next = prev->next
}

But how to remove the first element?
Remove linked list element

Basic idea: add a fake node at beginning
struct node {
    int val;
    struct node *next;
};

struct node*
removeElements(struct node* head, int val) {
    struct node *n = (struct node *)malloc(sizeof(struct node));
    struct node *r = n;

    n->next = head;
    while(n->next != NULL) {
        if (n->next->val == val) {
            n->next = n->next->next;
        } else {
            n = n->next;
        }
    }

    return r->next;
}
Lab 2

Approximate document matching (Due: 10/11)