CS 211
Instructor: Kangmei Yang
Eight numbers in a Cross
# Eight numbers in a Cross

![Cross Number grid](image)

## Cross Number Grid

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Cross Number Table

<table>
<thead>
<tr>
<th></th>
<th>-1</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Page 3
Eight numbers in a Cross

```c
bool ok(int q[8], int c){
    static int help_table[8][5] = {
        {-1},
        {0, -1},
        {...},
        {...},
        {...},
        ...
    };

    A for loop to check whether this value is good for this position

    return false;
    return true;
}
```
Static addressing

Static array:
You can not assign an array to another.

Ex:
int a[5];
int b[5];
a = b; //compile error

int x = 5, y = 0;
x = y;
C++ provides a operator specially for working with address.

* operator $\rightarrow$ “pointer” operator

Declare an integer pointer as: int *ip

Do not confuse pointer with other operator, such as,

```cpp
int a = 4;
a = a * 5; //this is multiplication

a = a * /*this is also multiplication with comments*/ 5;
```
int a[5] = {0}, x = 5;
int *ip = &x;

Note: int *ip != int **ip
Pointer

```c
int a[5] = {0}, x = 5;
int *ip = a;
```

Note: `cout << a << endl;` This prints out the address of array `a`
De-reference

Why do we use the integer pointer for both single integer and array of integers?

- Because we want to get the integer value.

De-reference of an int:
- Take the 4 bytes that it points to, represent the 32 bits as an integer.
- Use * operator for de-referencing.
What does it mean to add 1 to an pointer?

```c
int a[3];
int *ip = a;
ip = ip + 1 \leftarrow \text{the address of next integer}
```

Note: `int *ip != int **ip`
Syntactic Sugar

```c
int q[5]; // declare an array
q[4] == *(q+4)
```

```
07F0
```

```c
q + 4 == ?
```
2D Array

Instead of int or string or etc, as the data type, this array has 1D array as its data type.

Ex:
int array2[5][2]

0  1
2  3
4  5
6  7
8  9

0  1
2  3
4  5
6  7
8  9
2D Array

Ex:
int table[5][2]

- What is table[2]?
table[2] == *(table + 2)

- one int? or one array of int?

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>
Hexadecimal

Internally, all addresses (memory locations) are calculated as hex.

Memory unit is byte. The size of int is 4 bytes

Example:
integer array

<table>
<thead>
<tr>
<th>7F00</th>
<th>7F04</th>
<th>7F08</th>
<th>7F0C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimals</td>
<td>Binary (Base 2)</td>
<td>Hexadecimal (Base 16)</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>----------------</td>
<td>-----------------------</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1010</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1011</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>1100</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>1101</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>1110</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>1111</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>10000</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>10001</td>
<td>11</td>
<td></td>
</tr>
</tbody>
</table>
Converting Decimals to Hex

Use division – divide by the base 172:

\[
\begin{array}{c|c|c}
16 & 10 & 16 & 172 \\
16 & 16 & 160 & \\
10 & 12 & \\
\end{array}
\]

172 == AC

Converting Hex to Decimals

Use multiplication – times 16

\[
10F == 1 \times 16^2 + 0 \times 16^1 + F \times 16^0
\]

== 256 + 0 + 15

== 271
Hex-Decimal Addition

\[
\begin{array}{ccc}
0 & + & 0 \\
+ & 0 & \\
\hline
0 & & \\

4 & + & 5 \\
+ & 5 & \\
\hline
9 & & A
\end{array}
\]

\[
\begin{array}{ccc}
1 & + & 1 \\
+ & 4 & \\
\hline
2 & & 8
\end{array}
\]

\[
\begin{array}{ccc}
7 & + & 7 \\
+ & 7 & \\
\hline
E & & F
\end{array}
\]

\[
\begin{array}{ccc}
7 & + & 8 \\
+ & 8 & \\
\hline
8 & & 10
\end{array}
\]

\[
\begin{array}{ccc}
17 & + & 7 \\
+ & 98 & \\
\hline
1E & & AF
\end{array}
\]

\[
\begin{array}{ccc}
18 & + & 98 \\
+ & B0 & \\
\hline
& & \\
\end{array}
\]