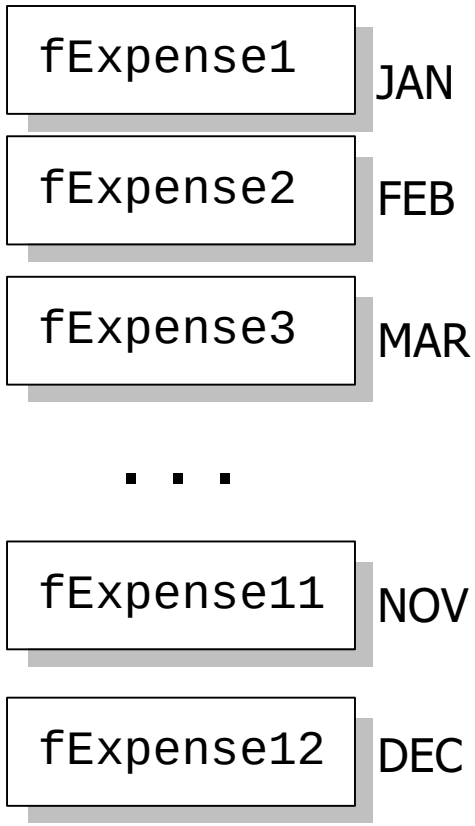
Outline

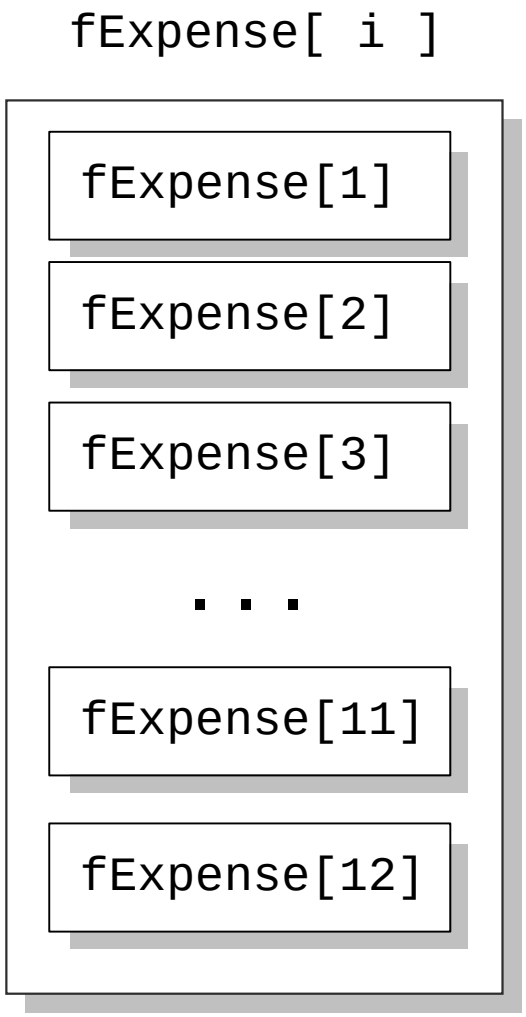
- Array manipulation
- Multidimensional arrays
- Arrays as parameters
- Structures
- Arrays of structures

Why do we need arrays ?




```
int    m;
float  e, fExpense1, fExpense2, fExpense3;
float  fExpense4, fExpense5, fExpense6;
float  fExpense7, fExpense8, fExpense9;
float  fExpense10, fExpense11, fExpense12;

scanf("%d %f", &m, &e );
switch ( m ) {
  case 1  : fExpense1  += e; break;
  case 2  : fExpense2  += e; break;
  case 3  : fExpense3  += e; break;
  case 4  : fExpense4  += e; break;
  case 5  : fExpense5  += e; break;
  case 6  : fExpense6  += e; break;
  case 7  : fExpense7  += e; break;
  case 8  : fExpense8  += e; break;
  case 9  : fExpense9  += e; break;
  case 10 : fExpense10 += e; break;
  case 11 : fExpense11 += e; break;
  case 12 : fExpense12 += e; break;
  default : error(MONTH_ERROR); break;
}
```

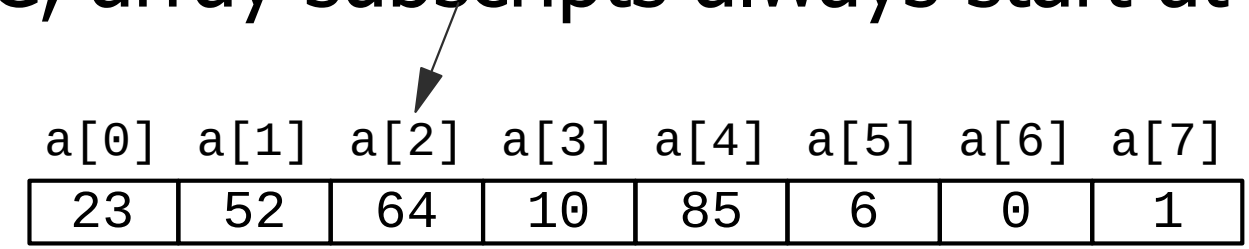
Why do we need arrays ?




```
int    m;
float  e;
float  fExpense[13];

scanf("%d %f", &m, &e );
if ( m >= 1 && m <= 12 )
  fExpense[ m ] += e;
else
  error(MONTH_ERROR);
```

What is an Array ?

- A collection of data storage locations, each having the same data type and the same name.
- In C, array subscripts always start at zero.




```
int    a[8];
```

number of array elements

Subscript

```
#define NUMBER_DEPT  20
#define NUMBER_EMPLOYEE  300
...
/* Department's expense */
float    fDeptExpense[NUMBER_DEPT];
/* Employee's department */
int      iDeptOf[NUMBER_EMPLOYEE];

...
fDeptExpense[ 0 ] = 0;
fDeptExpense[ i+3 ] *= 1.5;
fDeptExpense[ NUMBER_DEPT - i ] =
t = fDeptExpense[ iDeptOf[id] ];
...
```

```
int    m;
float  e;
float  fExpense[12];

scanf("%d %f", &m, &e );
if ( m >= 1 && m <= 12 )
  fExpense[ m-1 ] += e;
else
  error(MONTH_ERROR);
```

Example : Digit Counting

```
#include  <stdio.h>

main()
{
  char   c;
  int    i, iCount[10];

  for (i=0; i<10; i++) iCount[i] = 0;
  while ( (c=getchar()) != EOF )
    if ( c >= '0' && c <= '9' )
      ++iCount[ c - '0' ];
  printf("Digit\tCount\n");
  printf("-----\t-----\n");
  for (i=0; i<10; i++)
    printf(" %d\t  %d\n", i, iCount[i] );
}
```

```
ASCII

'0'  0x30   '0'-'0' = 0
'1'  0x31   '1'-'0' = 1
'2'  0x32   '2'-'0' = 2
'3'  0x33   '3'-'0' = 3
'4'  0x34   '4'-'0' = 4
'5'  0x35   '5'-'0' = 5
'6'  0x36   '6'-'0' = 6
'7'  0x37   '7'-'0' = 7
'8'  0x38   '8'-'0' = 8
'9'  0x39   '9'-'0' = 9
```

Multidimensional Arrays

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Admin												
Personnel												
Package												
R&D												
PR												
Maint												
QC												
Market												

Monthly expenses by departments

Multidimensional Arrays

```
float    fExpenseJan[0];
float    fExpenseFeb[1];
float    fExpenseMar[2];
...
float    fExpenseDec[7];
```

```
float    fExpenseAdmin[0];
float    fExpensePersonnel[1];
float    fExpensePackage[2];
...
float    fExpenseMarket[11];
```

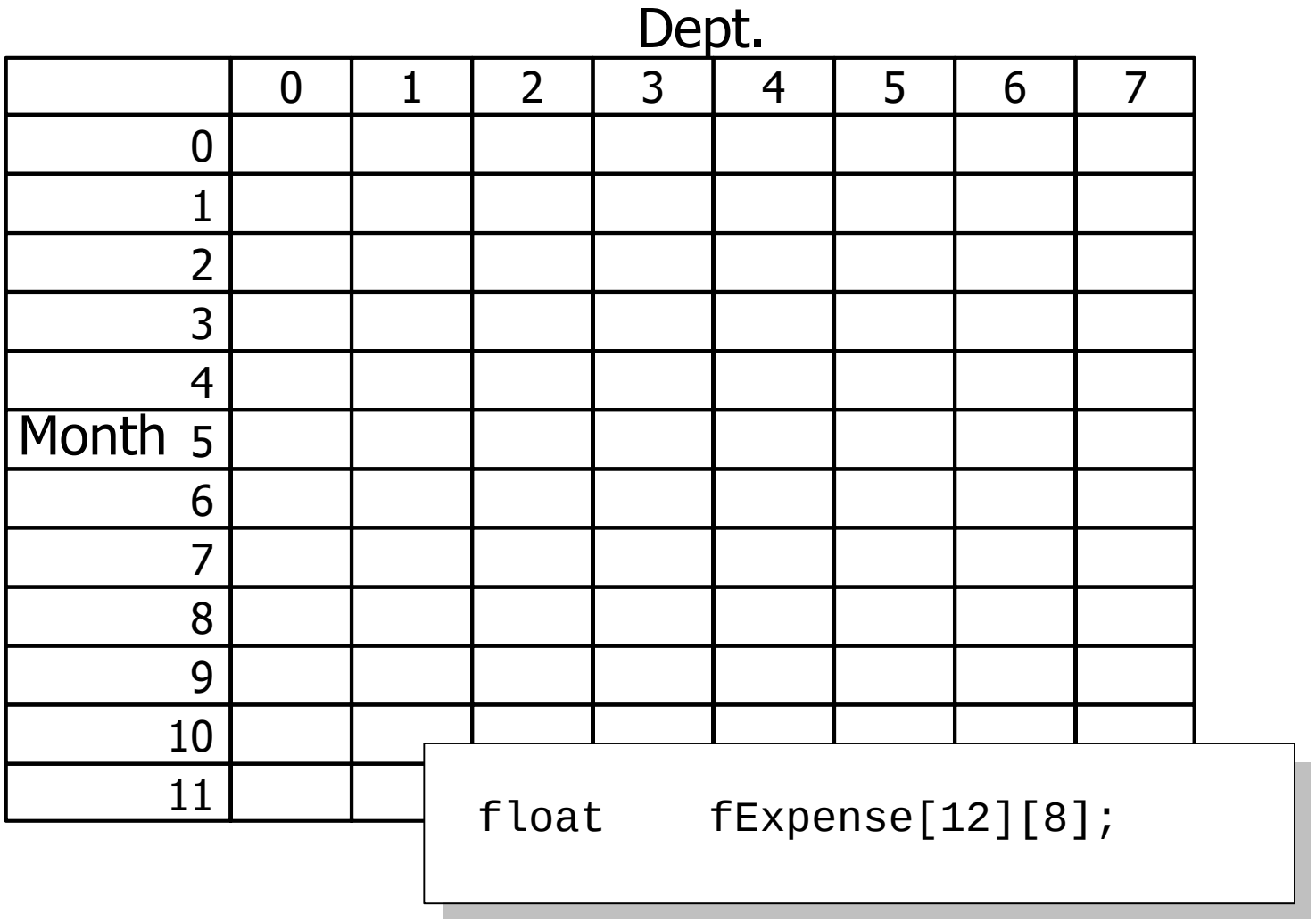
Multidimensional Arrays

Month

Dept.	0	1	2	3	4	5	6	7	8	9	10	11
0												
1												
2												
3												
4												
5												
6												
7												

```
float    fExpense[8][12];
```

Multidimensional Arrays



Multidimensional Arrays

```
float    fExpense[8][12];
float    fTotalExpense;
int      d, m;


...
fTotalExpense = 0;
for (d=0; i<8; i++) {
  for (m=0; j<12; j++) {
    fTotalExpense += fExpense[d][m];
  }
}
...
```

Symbolic Constants

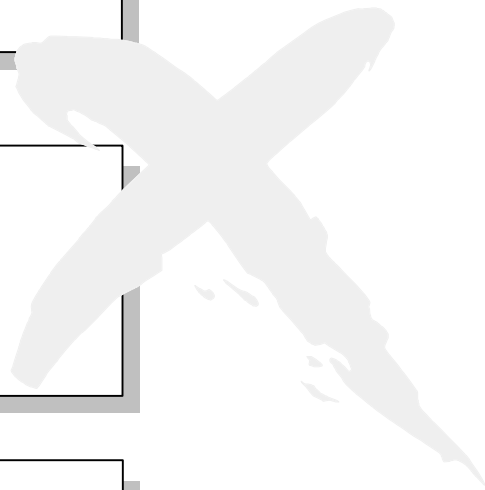
```
#define NUM_DEPT   8;
#define NUM_MONTH  12;

float    fExpense[NUM_DEPT][NUM_MONTH];
float    fTotalExpense;
int      d, m;

...
fTotalExpense = 0;
for (d=0; i<NUM_DEPT; i++) {
  for (m=0; j<NUM_MONTH; j++) {
    fTotalExpense += fExpense[d][m];
  }
}
...
```

You can't do this !!

```
int       iNumCandidate = 100;
double    dScore[ iNumCandidate ];
```

```
const  int  iNumCandidate = 100;
double    dScore[ iNumCandidate ];
```

```
double    dScore[ 25.0 ];
```



More Dimensions

```
#define NUM_DEPT   8;
#define NUM_YEAR   10;
#define NUM_MONTH  12;

float    fExpense[NUM_DEPT][NUM_YEAR][NUM_MONTH];
float    fTotalExpense;
int      d, m;
...
fTotalExpense = 0;
for (d=0; i<NUM_DEPT; i++)
  for (y=0; y<NUM_YEAR; y++)
    for (m=0; j<NUM_MONTH; j++)
      fTotalExpense += fExpense[d][y][m];
```

8x10x12 = 960 elements

Initializing Arrays

```
int   iAnswer[10] = {1,2,1,2,1,3,3,1,4,4};

/* auto. create array large enough to hold data */
int   iAnswer[]   = {1,2,1,2,1,3,3,1,4,4};

int   iMatrix[4][3] = { { 1,  2,  3},
                        { 4,  5,  6},
                        { 7,  8,  9},
                        {10, 11, 12} };

int   iMatrix[4][3] = { 1,2,3,4,5,6,7,8,9,10,11,12 };

int   iMatrix[][3]  = { { 1,  2,  3},
                        { 4,  5,  6},
                        { 7,  8,  9},
                        {10, 11, 12} };
```

Q & A

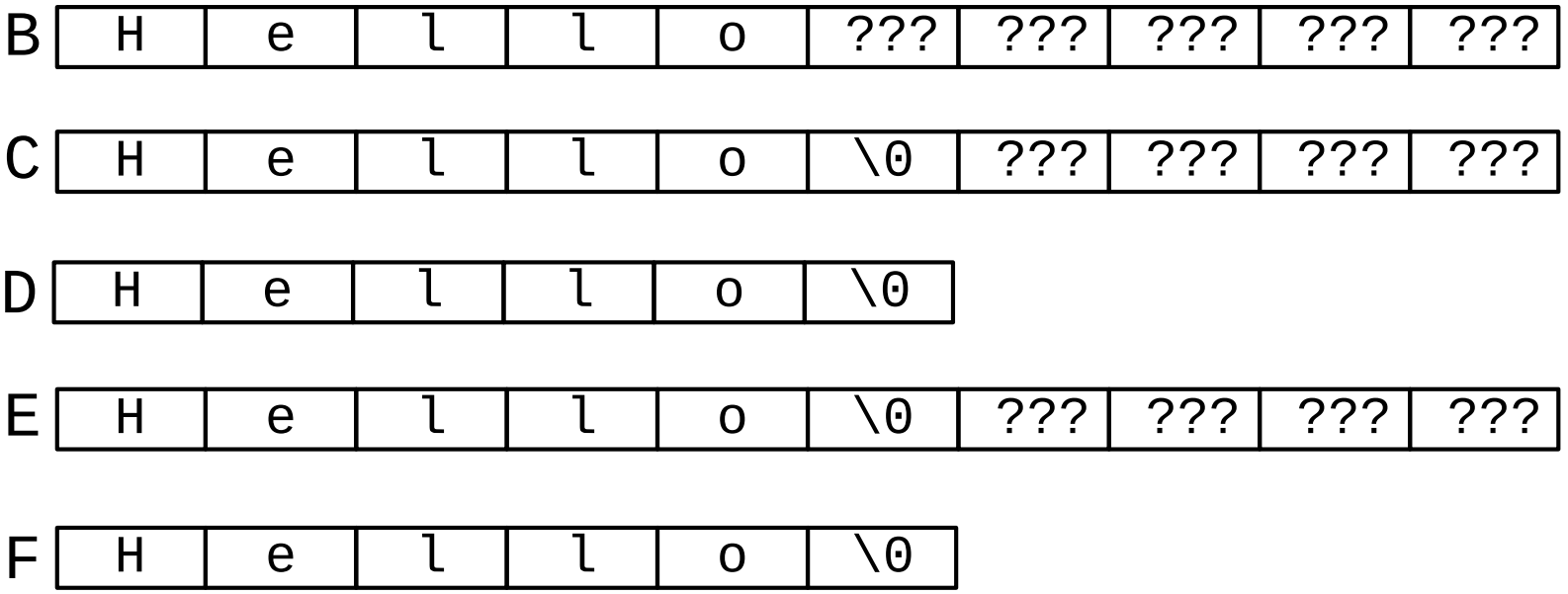
- What happen if I use a subscript on an array that is larger than the number of elements in the array ?
- How many dimensions can an array have ?
- Is there an easy way I can initialize an entire array at once ?
- Can I add two arrays together ?

```
int    a[10], b[10], c[10];
...
c = a + b;
...
```

```
int    a[10], b[10], c[10];
...
for (i=0; i<10; i++)
  c[i] = a[i] + b[i];
...
```

Arrays of Characters

```
char   A[10];
char   B[10] = { 'H', 'e', 'l', 'l', 'o' };
char   C[10] = { 'H', 'e', 'l', 'l', 'o', '\0' };
char   D[]   = { 'H', 'e', 'l', 'l', 'o', '\0' };
char   E[10] = "Hello";
char   F[]   = "Hello";
```



Arrays as Parameters

- An argument can be an `int`, a `float`, or any other basic data type.
- To pass an entire array, use the array name.

```
void func1( int, int, int [] );
...
main()
{
  int     a = 0;
  int     g[5] = {1,2,3,4,5};
  ...
  func1( a, g[1], g );
  ...
}
```

g is actually a starting address of the array g.
g = &(g[0])

Arrays as Parameters

```
void func1( int [], int[][5], int [][3][4] );
...
main()
{
  int     d1[10], d2[3][5], d3[2][3][4];
  ...
  func1( d1, d2, d3 );
  ...
}
void func1( int x1[], int x2[][5], int [][3][4] )
{
  ...
}
```

An Example

```
#include  <stdio.h>

int Largest( int x[], int n );
main()
{
  int      i, g[10];
  for (i=0; i<10; i++) scanf("%d", &(g[i]) );
  printf( "The largest value is %d\n", Largest( g, 10 ) );
}

int Largest( int x[], int n )
{
  int    max, i;
  max = x[0];
  for (i=0; i<n; i++)
    max = x[i] > max ? x[i] : max;
  return max;
}
```