ARRAYS

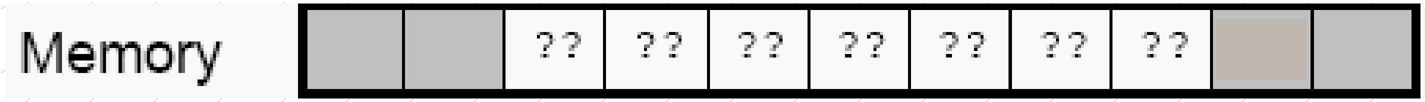
Data Structure

Definition

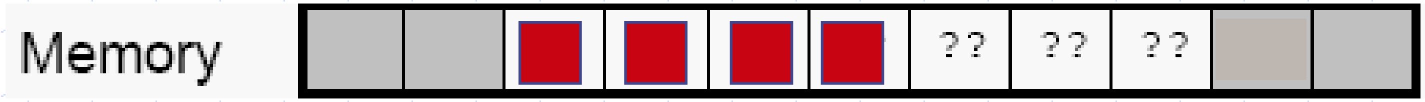
- Data Structure
- Sequence of variables - elements
  - fixed length
  - ordered
  - all elements are from the same type
  - accessed by an index

Definition

- A fixed-sized aggregation of a list of cells, each of which can hold a single values (objects).



- The number of cells in an array is called its size, or dimension.
- The number of values that are actually stored in an array is called its usage.
- The dimension MUST be a constant value (known at compile-time).
- The dimension and usage are separate values, with no association as far as the language is concerned with the array itself.



Declaration

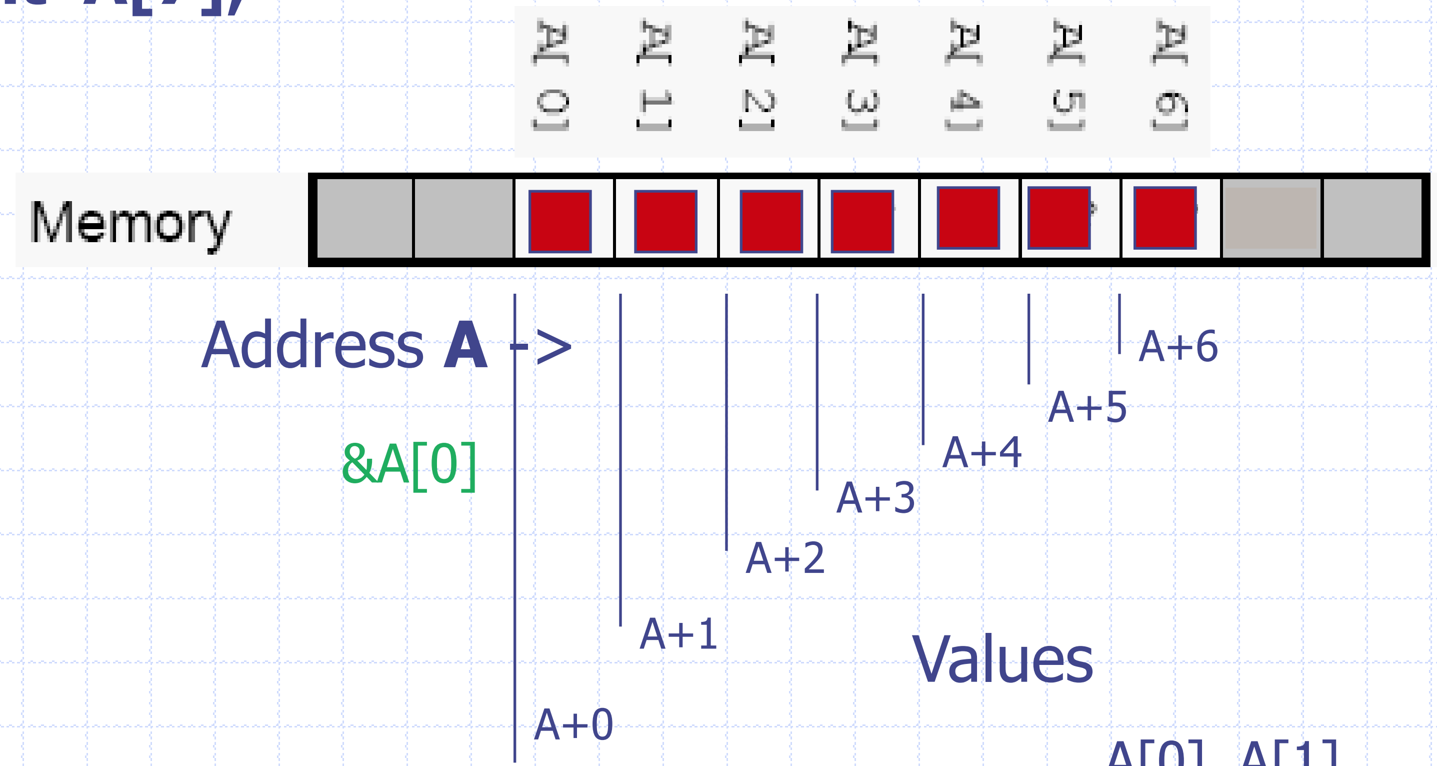
```
#define SIZE 256
#define SUMS 11

char buffer[SIZE];       // array with const integer dimension
int dice[SUMS + 1];      // array with const integer expression

int numItems = 10000;           // integer variable
int Inventory[numItems];        // NOT valid - not const
int Inventory[10000];           // integer constant
```

Characteristics

- Access to individual cells by index, or subscript
  - integer number, between 0 and n-1
- Limitations
  - There is no way to change the dimension of an array once it is declared.
  - There is no automatic aggregate operations for arrays
    - operator = does not copy the contents one array into another
    - operator == not supported for arrays

Indices

int A[7];



Address **A** ->

&A[0]

Values

A[0], A[1], ...

*(A+0), *(A+1), ...

Processing

- Initialization
  - at declaration

```
int a[100];                   // declaration
int a[100] = {0};             // initialization
int a[5] = {1,2,3,6,8};       // initialization
int a[5] = {1,2,3};           // initialization
```

  - at run time

```
for (i=0; i<n; i++)
    a[i] = 0;
```

Processing

- Input
  - certain number of elements

```
int k, int x[20];
for (k=0; k<n; k++)
                    scanf("%d", &x[k]);
```

  - un-certain number of elements

```
int k, int x[20];
do
         scanf("%d", &x[k]);
while (x[k++] > 0);
```

Processing

- Input
  - un-certain number of elements, but not bigger than the limit

```
int k = 0, int x[20];
do
        scanf("%d", &x[k]);
while (x[k] > 0 && ++k < 20);
```

  - certain number of elements, but without illegal value

```
int k = 0, int x[20];
 for ( ; k < 20; k++)
{
        scanf("%d", &x[k]);
        if (x[k] < 0) break;
}
```

```
int k = 0, int x[20];
 for ( ; k < 20; k++)
{
        scanf("%d", &x[k]);
        if (x[k] < 0) continue;
}
```

Processing

- Output
  - certain number of elements all on one line

```
int k, int x[20];
for (k=0; k<n; k++)
        printf("%d  ", x[k]);
```

  - certain number of elements one on a line

```
int k, int x[20];
for (k=0; k<n; k++)
        printf("%d\n", x[k]);
```

  - certain number of elements, M on a line

```
int k, int x[20];
for (k=0; k<n; k++)
{       printf("%d ", x[k]);
        if (k % M == 0)      printf("\n");
}
```

Multi-dimentional Arrays

- Two-dimensional array – a matrix

```
#define M 30                    // rows
#define N 20                    // columns

int mat[M][N];
int i, j;
for (i=0; i<M; i++)
   for (j=0; j<N; j++)
            mat[i][j] += mat[i][j];
```

- Multi-dimensional array

String as an Array

- String Objects
  - constants   “WORD”
  - variables   char w[5];
  - special character ‘\0’
- Processing
  - input   gets()
  - output   puts()
  - referencing   “WORD” [0], w+3
  - comparison   ??
- Library Functions