Arrays and Structures

Ikjun Yeom

Originated from Prof. Kyungran Kang

Contents

Arrays

Dynamically Allocated Arrays

Structures and Unions

Polynomials

Sparse Matrices

Strings

Definition of Array

- “a set of consecutive memory locations”
- “a collection of data of the same type”
- “a set of pairs, <index, value>, such that each index that is defined has a value associated with it”

Array as an Abstract Data Type

ADT Array is

objects: A set of pairs <*index*, *value*> where for each value of *index* there is a value from the set *item*.

index is a finite ordered set of one or more dimensions

functions: for all *A* ∈ *Array*, *i* ∈ *index*, *x* ∈ *item*, *j*, *size* ∈ integer

Array Create(*j*,*list*)::= return …

item Retrieve(*A*,*i*)::= if () return … else return …

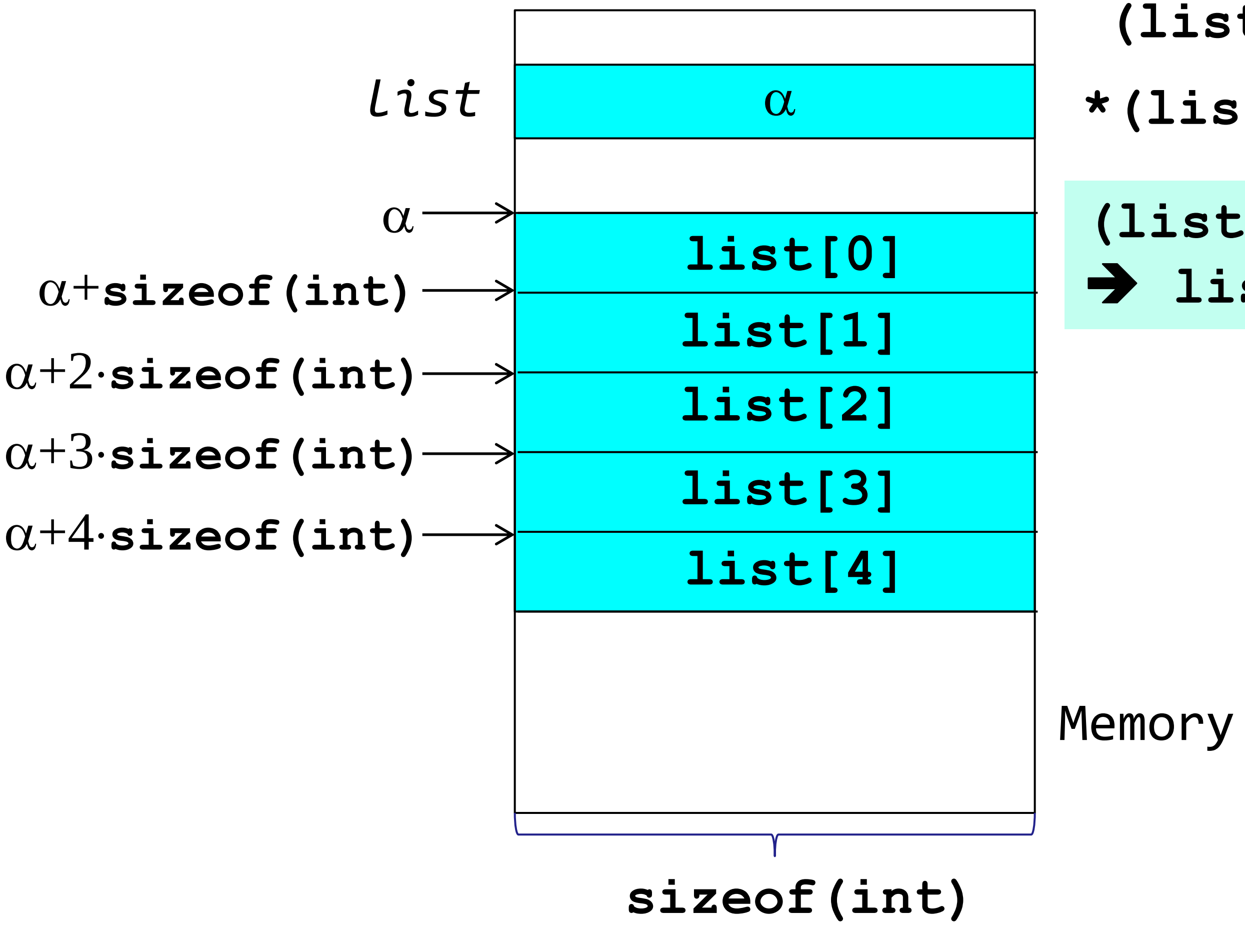
Array Store(*A*,*i*,*x*)::= if () return … else return …

end Array

Array Representation in C

Data structure for one dimensional array

- **int list[5]** ➔ int* list;



(list+2) vs. &list[2]

***(list+2) vs. list[2]**

(list+2)
➔ list + 2*sizeof(int)

Example Array Program

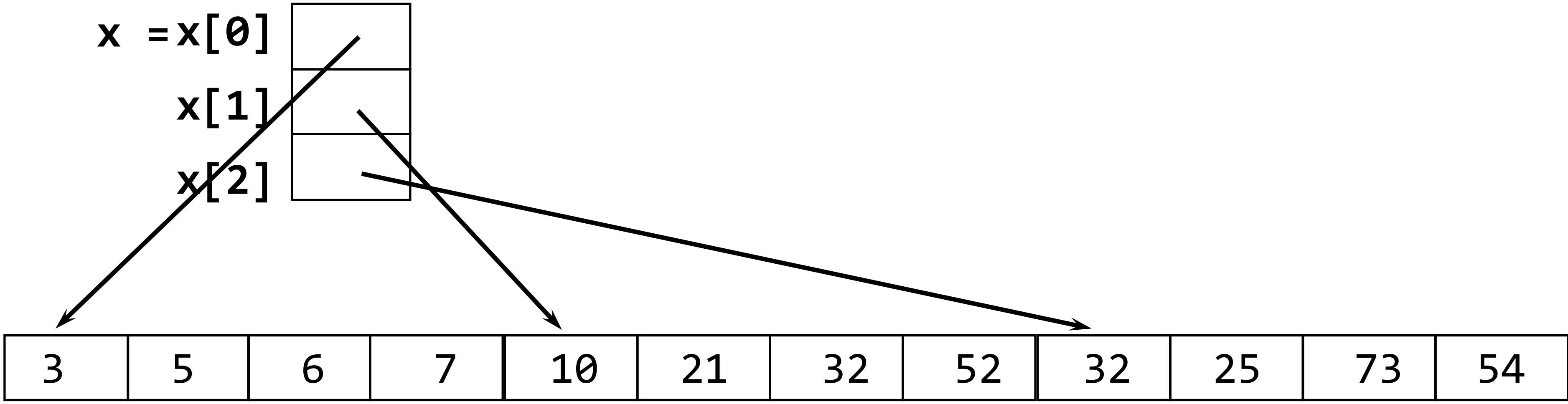
```
#define MAX_SIZE 100
float sum(float [], int);
float input[MAX_SIZE], answer;
int i;

void main(void)
{
   for (i = 0; i < MAX_SIZE; i++)
      input[i] = i;
   answer = sum(input, MAX_SIZE);
   printf("The sum is: %f\n", answer);
}

float sum(float list[], int n)
{
   int i;
   float tempsum = 0;
   for (i = 0; i < n; i++)
        tempsum += list[i];
   return tempsum;
}
```

2D Array Representation In C

```
int x[3][4];
```



x = x[0]

x[1]

x[2]

3	5	6	7	10	21	32	52	32	25	73	54

Array-of-arrays representation

- Requires contiguous memory of size 3, 4, 4, and 4 for the 4 1D arrays
- One memory block of size number of rows and number of rows blocks of size number of columns

Dynamically Allocated Arrays – One Dimension (1)

```
#include <math.h>
#include <stdio.h>
#define MAX_SIZE 101

main()
{
 int i, n, list[MAX_SIZE];

 printf(“Enter the number of number to generate: ”);
 scanf(“%d”, &n”);

 for (i=0;i<n;i++)
 {
        list[i]=rand()%1000;
        printf (“%d\n”, list[i]);
 }
}
```

What if we change MAX_SIZE to a very large number to avoid run time error??

→ We increase the likelihood that program may fail to compile due to lack of memory

Dynamically Allocated Arrays – One Dimension (2)

```
int i, n, *list;

printf(“Enter the number of number to generate: ”);
scanf(“%d”, &n);

if (n < 1) {
        fprintf(stderr, “Improper value of n \n”);
        exit (EXIT_FAILURE);
}

list=malloc(n * sizeof(int));
if (list==NULL) {
        fprintf(stderr, “lack of memory\n”);
        exit (EXIT_FAILURE);
}
```

It fails only when n<1 or we do not have sufficient memory to hold the list of numbers

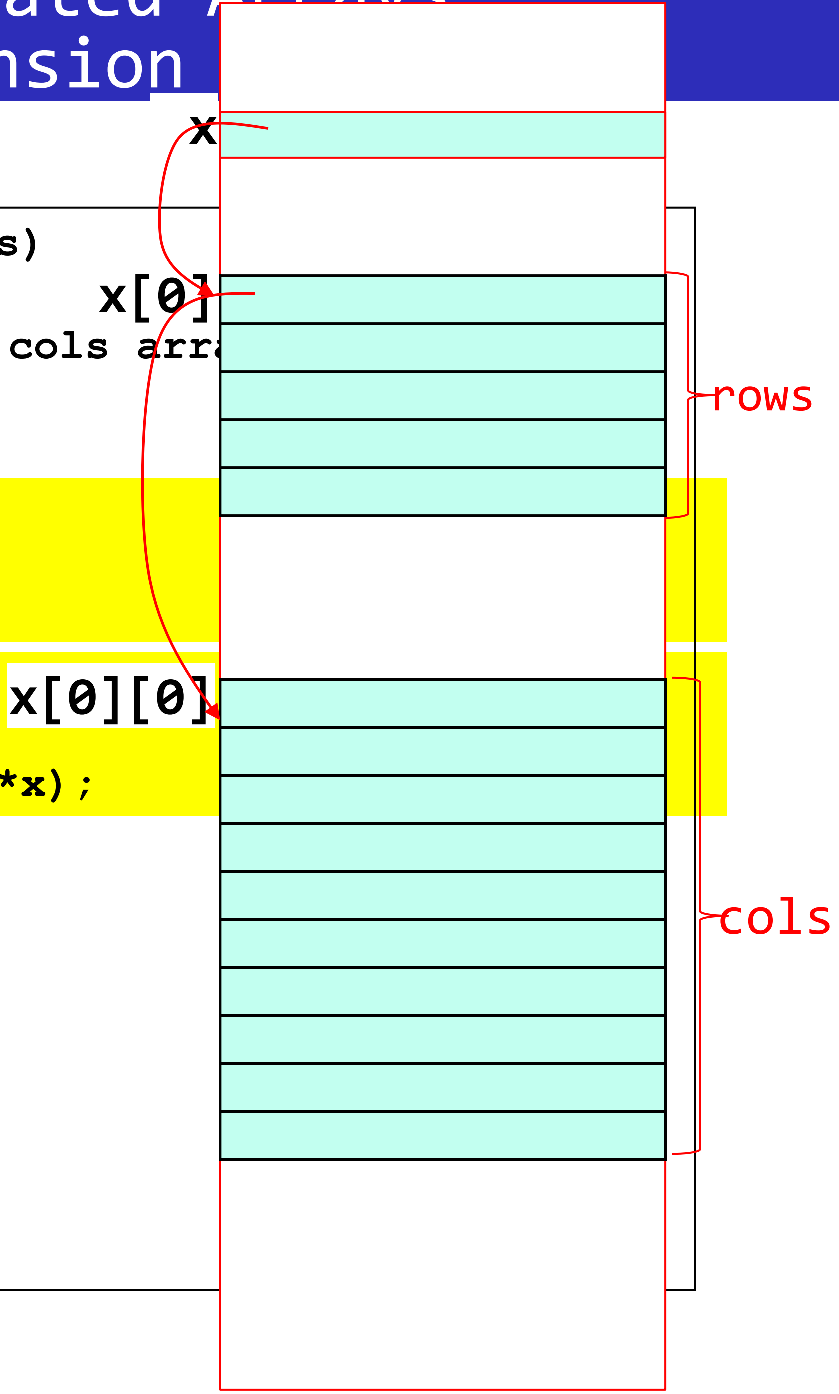
Dynamically Allocated Arrays – Two Dimension

```
int** make2dArray(int rows, int cols)
{
/* create a two dimensional rows × cols arra

  int** x, i;

  /* get memory for row pointers */
  MALLOC(x, rows * sizeof (*x));
  /* get memory for each row */
  for (i = 0; i < rows; i++)
       MALLOC(x[i], cols * sizeof(**x);
  return x;
}

void main()
{
  int** myArray;
  myArray=make2dArray(5,10);
  myArray[2][4]=6;
}
```



Quiz 4

Name and student ID

Let length[i] be the desired length of row i of a two dimensional array. Write a function similar to make2dArray() to create a two dimensional array such that row i has length[i] elements.

Structures (1)

A collection of data items, where each item is identified as to its type and name

```
struct {
  char name[10];
  int age;
  float salary;
} person;

strcpy(person.name, “korykang”);
person.age=34;
person.salary=35000;
```

Structures (2)

To create own structure data type

```
typedef struct humanBeing {
        char name[10];
        int age;
        float salary;
 };
```

or

```
typedef struct {
        char name[10];
        int age;
        float salary;
} humanBeing;
```

```
humanBeing person1, person2;
```

Assignments

```
    person1=person2;
```

Or

```
    strcpy(person1.name, person2.name);
    person1.age=person2.age;
    person1.salary=person2.salary;
```

```
if (person1 == person2) {
…}
```

????

(Chapter 2 p.61)

Structures (3)

To embed a structure within a structure

```
typedef struct {
  int month;
  int day;
  int year;
} date;

typedef struct humanBeing{
  char name[10];
  int age;
 float salary;
  date dob;
} humanBeing;

person1.dob.month=12; person1.dob.day=3;
person1.dob.year=1969;
```

Quiz 5

Name and student ID

Develop a structure to represent each of the following geometric objects: rectangle, triangle, and circle.

Applications of Array

Ordered list

- One dimensional array
- Polynomial: components will be numeric
- String: components can be non-numeric

Matrix

- Standard: multi-dimensional array
- Sparse matrix: one dimensional array
- Components may be numeric

Ordered List (1)

Example

- Days of a week

(Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday)

- Values in a deck of cards

(Ace, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, King)

- Years Switzerland fought in World War II: ()

Ordered List (2)

Operations

- Finding the length, n, of a list
- Reading the items in a list from left to right (or reversely)
- Retrieving the i-th item from a list, $0 \leq i \leq n$
- Replacing the item in the i-th position of a list, $0 \leq i \leq n$
- Inserting a new item in the i-th position of a list, $0 \leq i \leq n$
- Deleting an item from the i-th position of a list, $0 \leq i \leq n$

Polynomial Abstract Data Type (1)

Example of polynomials

A(x) = 3x^6 + 2x^5 + 4, B(x) = x^4 + 10x^3 + 3x^2 + 1

→ **A(x) + B(x) = 3x^6 + 2x^5 + x^4 + 10x^3 + 3x^2 + 5**

Manipulation of symbolic polynomials

$$A(x) = \sum_{i=0}^{n} a_i x^i \qquad B(x) = \sum_{j=0}^{m} b_j x^j$$

$$A(x) + B(x) = \sum_{i=0}^{\max(n,m)} (a_i + b_i) \cdot x^i$$

$$A(x) \cdot B(x) = \sum_{i=0}^{n} (a_i x^i \sum_{j=0}^{m} (b_j \cdot x^j))$$

Polynomial Abstract Data Type (2)

ADT *Polynomial* is

objects : $p(x) = a_1 x^{e_1} + \ldots + a_n x^{e_n}$: a set of ordered pairs of <e_i, a_i>, where a_i in *Coefficients* and e_i in *Exponents*, e_i are integers ≥ 0.

functions : for all ***poly***, ***poly1***, ***poly2*** ∈ *Polynomial*, ***coef*** ∈ *Coefficients*, ***expon*** ∈ *Exponents*

```
  Polynomial Zero() ::= …
  Boolean IsZero(poly) ::= …
  Coefficient Coef(poly, expon) ::= …
  Exponent LeadExp(poly) ::= …
  Polynomial Attach(poly, coef, expon) ::= …
  Polynomial Remove(poly, expon) ::= …
  Polynomial SingleMult(poly, coef, expon) ::= …
  Polynomial Add(poly1, poly2) ::= …
  Polynomial Mult(poly1, poly2) ::= …
```

end *Polynomial*

Polynomial Abstract Data Type (3)

```
/* d = a + b, where a, b, and d are polynomials */
d = Zero();
while (! IsZero(a) && ! IsZero(b)) do  {
 switch COMPARE(LeadExp(a), LeadExp(b)) {
  case -1:  /* if  LeadExp(a) < LeadExp(b) */
   d = Attach (d, Coef(b, LeadExp(b)), LeadExp(b));
   b = Remove(b, LeadExp(b));
   break;
  case 0: /* if  LeadExp(a) == LeadExp(b) */
   sum = Coef(a, LeadExp(a)) + Coef(b, LeadExp(b));
   if (sum) {
      d = Attach(d, sum, LeadExp(a));
      a = Remove(a, LeadExp(a));
      b = Remove(b, LeadExp(b));
   }
   break;
  case 1: /* if  LeadExp(a) > LeadExp(b) */
   d = Attach(d, Coef(a, LeadExp(a)), LeadExp(a));
   a = Remove(a, LeadExp(a));
  }
}
Insert any remaining terms of a or b into d
```

$A(x) = 3x^6 + 2x^5 + 4$, $B(x) = x^4 + 10x^3 + 3x^2 + 1$

Polynomial Representation (1)

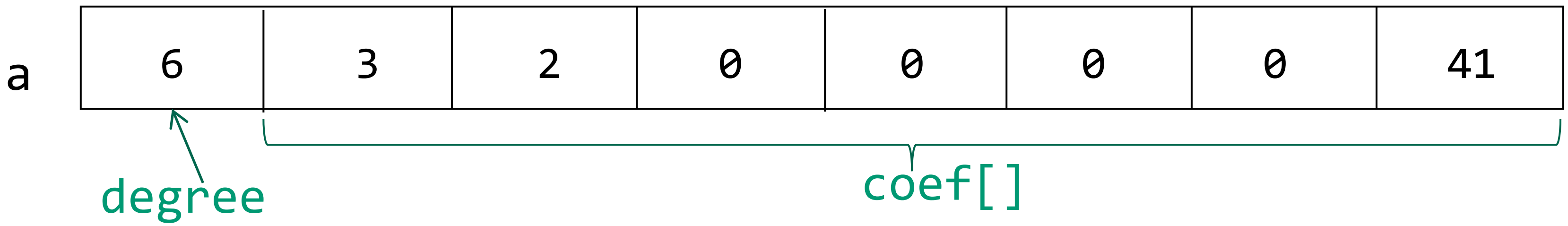
```
#define MAX_DEGREE 101
typedef struct {
      int degree;
      float coef[MAX_DEGREE];
} polynomial;
```

n < MAX_DEGREE $$A(x) = \sum_{i=0}^{n} a_i x^i$$

```
polynomial a;
```

a.degree = n, a.coef[i] = a_{n-i}, 0≤i≤n

A(x) = $3x^6 + 2x^5 + 41$ **How about $2x^{1000} + 1$?**



a							
6	3	2	0	0	0	0	41

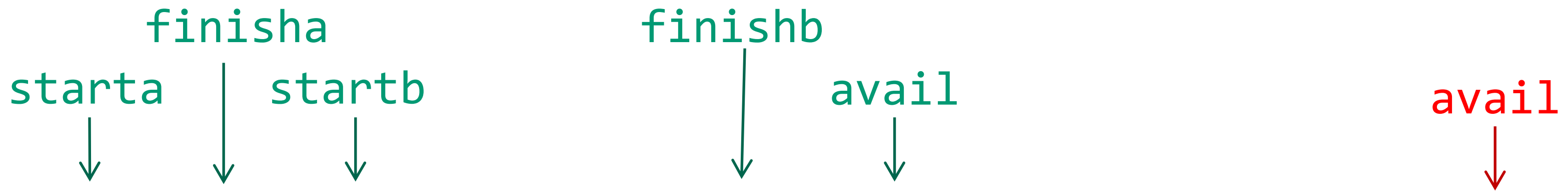
Polynomial Representation (2)

```
#define MAX_TERMS 100
typedef struct {
        float coef;
        int expon;
} polynomial;
polynomial terms[MAX_TERMS];
int starta, finisha, startb, finishb, avail;
starta=0; finisha=1; startb=2; finishb=5; avail=6;
```

~~#define MAX_DEGREE 101~~

$A(x) = 2x^{1000} + 1,$

$B(x) = x^4 + 10x^3 + 3x^2 + 1$



terms[]	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]
coef	2	1	1	10	3	1	2	1	10	3	2	
expon	1000	0	4	3	2	0	1000	4	3	2	0	

Quiz 6

Name and student ID

Implement Remove() and Attach() in slide 21 using polynomial representation in slide 22.