- One way to represent polynomials in C is as follows:

```
#define MAX_DEGREE 101 /* Max degree of polynomial+1 */
typedef struct {
        int degree;
        float coef[MAX_DEGREE];
} polynomial;
```

- Now if a is of type polynomial and $n < MAX\_DEGREE$, the polynomial $A(x) = \sum_{i=0}^{n} a_i x^i$ would be represented as: 1) a.degree = n 2) a.coeff[i] = a_{n-i} (coefficient of x^{n-i}), $0 \le i \le n$.
- Although this representation leads to very simple algorithms for most of the operations, it wastes a lot of space.
- For ex, if a.degree $\ll$ MAX_DEGREE, then we will not need most of the positions in a.coef[MAX_DEGREE].
- The same argument applies if the polynomial is sparse, that is, the number of terms with nonzero coefficient is small relative to the degree of the polynomial.

Alternate Representation

- Global array terms → stores all our polynomials

C declarations

```
MAX_TERMS 100 /* size of terms array */
typedef struct {
        float coef;
        int expon;
} polynomial;
polynomial terms[MAX_TERMS];
int avail = 0;
```

	startA ↓	finishA ↓	startB ↓			finishB ↓	avail ↓
coef	2	1	1	10	3	1	
exp	1000	0	4	3	2	0	
	0	1	2	3	4	5	6.