- With this mapping, we can retrieve an item, replace an item, or find the length of a list in constant time
- We also can read the items in the list, from either direction, by simply changing subscripts in a controlled way.
- Insertion and deletion pose problems since we must move the items to preserve sequential mapping. This is the reason non-sequential mapping is needed.

Problem: Build a set of functions that allow for the manipulation of symbolic polynomials. $A(x)=\sum a_i x^i$ $B(x)=\sum b_i x^i$, then $A(x)+B(x)=\sum(a_i+b_i)x^i$ and $A(x)\cdot B(x)=\sum(a_i x^i \cdot \sum(b_j x^j))$. Similarly we can define subtraction and division on polynomials, as well as many other operations.

Abstract data type Polynomial

ADT Polynomial is

Objects: $P(x)=a_1 x^{e_1}+\ldots\ldots+a_n x^{e_n}$; a set of ordered pairs of $\langle e_i, a_i\rangle$ 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() ::= return the polynomial, $p(x)=0$
- Boolean IsZero(poly) ::= if(poly) return FALSE
  else return TRUE
- Coefficient Coef(poly, expon) ::= if(expon ∈ poly) return its coefficient
  else return Zero
- Exponent LeadExp(poly) ::= return the largest exponent in poly
- Polynomial Attach(poly, coef, expon) ::= if(expon ∈ poly) return error
  else return the polynomial poly
  with the term $\langle$coef, expon$\rangle$ inserted.
- Polynomial Remove(poly, expon) ::= if(expon ∈ poly)
  return the polynomial poly with the term
  whose exponent is expon deleted else return error