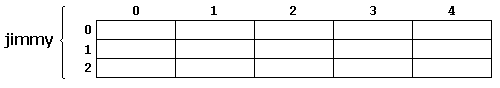
**Multidimensional arrays.**

Multidimensional arrays can be described as "arrays of arrays". For example, a bidimensional array can be imagined as a two-dimensional table made of elements, all of them of a same uniform data type.

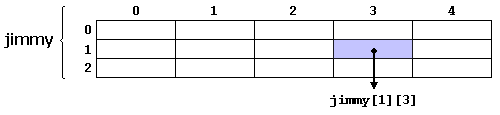


jimmy represents a bidimensional array of 3 per 5 elements of type int. The C++ syntax for this is:

|  |  |  |
| --- | --- | --- |
|  | int jimmy [3][5]; |  |

and, for example, the way to reference the second element vertically and fourth horizontally in an expression would be:

|  |  |  |
| --- | --- | --- |
|  | jimmy[1][3] |  |



(remember that array indices always begin with zero).

Multidimensional arrays are not limited to two indices (i.e., two dimensions). They can contain as many indices as needed. Although be careful: the amount of memory needed for an array increases exponentially with each dimension. For example:

|  |  |  |
| --- | --- | --- |
|  | char century [100][365][24][60][60]; |  |

declares an array with an element of type *char* for each second in a *century*. This amounts to more than 3 billion *char*! So this declaration would consume more than 3 gigabytes of memory!

At the end, multidimensional arrays are just an abstraction for programmers, since the same results can be achieved with a simple array, by multiplying its indices:

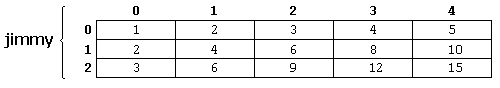
|  |  |  |
| --- | --- | --- |
| 1 2 | int jimmy [3][5]; // is equivalent to  int jimmy [15]; // (3 \* 5 = 15) |  |

With the only difference that with multidimensional arrays, the compiler automatically remembers the depth of each imaginary dimension.

The following two pieces of code produce the exact same result, but one uses a bidimensional array while the other uses a simple array:

|  |  |
| --- | --- |
| **multidimensional array** | **pseudo-multidimensional array** |
| #define WIDTH 5  #define HEIGHT 3  int jimmy [HEIGHT][WIDTH];  int n,m;  int main ()  {  for (n=0; n<HEIGHT; n++)  for (m=0; m<WIDTH; m++)  {  jimmy[n][m]=(n+1)\*(m+1);  }  } | #define WIDTH 5  #define HEIGHT 3  int jimmy [HEIGHT \* WIDTH];  int n,m;  int main ()  {  for (n=0; n<HEIGHT; n++)  for (m=0; m<WIDTH; m++)  {  jimmy[n\*WIDTH+m]=(n+1)\*(m+1);  }  } |

None of the two code snippets above produce any output on the screen, but both assign values to the memory block called jimmy in the following way:



Note that the code uses defined constants for the width and height, instead of using directly their numerical values. This gives the code a better readability, and allows changes in the code to be made easily in one place.

Below main useful needed cpp programs using arrays:

|  |  |  |
| --- | --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41 | //addition of two matrices  #include <iostream>  using namespace std;  int main()  {  int row, column, a[10][10], b[10][10], sum[10][10], i, j;  cout << "Enter number of rows (between 1 and 10): ";  cin >> row;  cout << "Enter number of columns (between 1 and 10): ";  cin >> column;  cout << endl << "Enter elements of 1st matrix:" << endl;  //Storing elements of first matrix entered by user.  for (i = 0; i < row; ++i)  for (j = 0; j < column; ++j)  {  cout << "Enter element a" << i + 1 << j + 1 << ": ";  cin >> a[i][j];  }  //Storing elements of second matrix entered by user.  cout << endl << "Enter elements of 2nd matrix: " << endl;  for (i = 0; i < row; ++i)  for (j = 0; j < column; ++j)  {  cout << "Enter element b" << i + 1 << j + 1 << ": ";  cin >> b[i][j];  }  //Adding two matrices  for (i = 0; i < row; ++i)  for (j = 0; j < column; ++j)  sum[i][j] = a[i][j] + b[i][j];  //Displaying the resultant sum matrix.  cout << endl << "Sum of two matrix is: " << endl;  for (i = 0; i < row; ++i)  for (j = 0; j < column; ++j)  {  cout << sum[i][j] << " ";  if (j == column - 1)  cout << endl;  }  return 0;  } | //subtraction of two matrices  #include <iostream>  using namespace std;  int main()  {  int row, column, a[10][10], b[10][10], sum[10][10], i, j;  cout << "Enter number of rows (between 1 and 10): ";  cin >> row;  cout << "Enter number of columns (between 1 and 10): ";  cin >> column;  cout << endl << "Enter elements of 1st matrix:" << endl;  //Storing elements of first matrix entered by user.  for (i = 0; i < row; ++i)  for (j = 0; j < column; ++j)  {  cout << "Enter element a" << i + 1 << j + 1 << ": ";  cin >> a[i][j];  }  //Storing elements of second matrix entered by user.  cout << endl << "Enter elements of 2nd matrix: " << endl;  for (i = 0; i < row; ++i)  for (j = 0; j < column; ++j)  {  cout << "Enter element b" << i + 1 << j + 1 << ": ";  cin >> b[i][j];  }  //Adding two matrices  for (i = 0; i < row; ++i)  for (j = 0; j < column; ++j)  sum[i][j] = a[i][j] - b[i][j];  //Displaying the resultant matrix.  cout << endl << "Sum of two matrix is: " << endl;  for (i = 0; i < row; ++i)  for (j = 0; j < column; ++j)  {  cout << sum[i][j] << " ";  if (j == column - 1)  cout << endl;  }  return 0;  } |

|  |  |  |
| --- | --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13  14  15  16  17  18  19  20  21  22  23  24 | //The largest element in an array  #include <iostream>  using namespace std;  int main()  {  int i, n;  double arr[100];  cout << "Enter total number of elements(1 to 100): ";  cin >> n;  // Store number entered by the user  for (i = 0; i < n; ++i)  {  cout << "Enter number " << i + 1 << ": ";  cin >> arr[i];  }  // Loop to store largest number to arr[0]  for (i = 1; i < n; ++i)  {  if (arr[0] < arr[i])  arr[0] = arr[i];  }  cout << "Largest element in the array = " << arr[0];  return 0;  } | //The smallest element in an array  #include <iostream>  using namespace std;  int main()  {  int i, n;  double arr[100];  cout << "Enter total number of elements(1 to 100): ";  cin >> n;  // Store number entered by the user  for (i = 0; i < n; ++i)  {  cout << "Enter number " << i + 1 << ": ";  cin >> arr[i];  }  // Loop to store largest number to arr[0]  for (i = 1; i < n; ++i)  {  if (arr[0] > arr[i])  arr[0] = arr[i];  }  cout << "Smallest element in the array = " << arr[0];  return 0;  } |

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61 | //multiplication of two matrices  #include <iostream>  using namespace std;  int main()  {  int a[10][10], b[10][10], mult[10][10], r1, c1, r2, c2, i, j, k;  cout << "Enter rows and columns for first matrix: ";  cin >> r1 >> c1;  cout << "Enter rows and columns for second matrix: ";  cin >> r2 >> c2;  /\*If column of first matrix in not equal to row of second matrix,  ask the user to enter the size of matrix again.\*/  while (c1 != r2)  {  cout << "Error! column of first matrix not equal to row of second.";  cout << endl  cout << "Enter rows and columns for first matrix: ";  cin >> r1 >> c1;  cout << "Enter rows and columns for second matrix: ";  cin >> r2 >> c2;  }  //Storing elements of first matrix.  cout << endl << "Enter elements of matrix 1:" << endl;  for (i = 0; i < r1; ++i)  for (j = 0; j < c1; ++j)  {  cout << "Enter element a" << i + 1 << j + 1 << ": ";  cin >> a[i][j];  }  //Storing elements of second matrix.  cout << endl << "Enter elements of matrix 2:" << endl;  for (i = 0; i < r2; ++i)  for (j = 0; j < c2; ++j)  {  cout << "Enter element b" << i + 1 << j + 1 << ": ";  cin >> b[i][j];  }  //Initializing elements of matrix mult to 0.  for (i = 0; i < r1; ++i)  for (j = 0; j < c2; ++j)  {  mult[i][j] = 0;  }  //Multiplying matrix a and b and storing in array mult.  for (i = 0; i < r1; ++i)  for (j = 0; j < c2; ++j)  for (k = 0; k < c1; ++k)  {  mult[i][j] += a[i][k] \* b[k][j];  }  //Displaying the multiplication of two matrices.  cout << endl << "Output Matrix: " << endl;  for (i = 0; i < r1; ++i)  for (j = 0; j < c2; ++j)  {  cout << " " << mult[i][j];  if (j == c2 - 1)  cout << endl;  }  return 0;  } |

Above in multiplication matrix code, it should be noted that when initializing the elements of the matrix *mult* to zero, it is not necessary to use loops for the elements of them to have value of zero. We could have just initialized all elements to zero by declaring them using braces like below:

int mult[2][3]{}; ***OR*** int mult[2][3] = {}; // here all elements now initialized to zero

Observe below two codes. The first one on the left have an array *mult* where all elements have an undetermined value (including zero). While the one which is on the right side has been declared and initialized with braces ({}) even with no value(s), thence all elements in the arrays *mult* have value zero:

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43 | //Find transpose of the matrix using arrays  #include <iostream>  using namespace std;  int main()  {  int a[10][10], transpose[10][10], row, column, i, j;  cout << "Enter rows and columns of matrix: ";  cin >> row >> column;  cout << "Enter elements of matrix: " << endl;  // Storing matrix elements  for (int i = 0; i < row; ++i) {  for (int j = 0; j < column; ++j) {  cout << "Enter element a" << i + 1 << j + 1 << ": ";  cin >> a[i][j];  }  }  // Printing the a matrix  cout << "Entered Matrix: " << endl;  for (int i = 0; i < row; ++i)  {  for (int j = 0; j < column; ++j)  {  cout << " " << a[i][j];  if (j == column - 1)  cout << endl;  }  }  // Computing transpose of the matrix  for (int i = 0; i < row; ++i)  for (int j = 0; j < column; ++j) {  transpose[j][i] = a[i][j];  }  // Printing the transpose  cout << "\nTranspose of Matrix: " << endl;  for (int i = 0; i < column; ++i)  for (int j = 0; j < row; ++j)  {  cout << " " << transpose[i][j];  if (j == row - 1)  cout << endl;  }  return 0;  } |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 2 3  4  5  6  7  8  9  10  11  12  13 | #include <iostream>  using namespace std;  int main()  {  int mult[2][3];  for (int i = 1; i <= 2; i++)  {  cout << endl;  for (int j = 1; j <= 3; j++)  cout << mult[i][j] << “ ”;  }  } | #include <iostream>  using namespace std;  int main()  {  int mult[2][3]{}; //or int mult[2][3] = {};  for (int i = 1; i <= 2; i++)  {  cout << endl;  for (int j = 1; j <= 3; j++)  cout << mult[i][j] << “ ”;  }  } |  |
|  | Output:  10925407 0 4127232  11350869 10883502 10883502 | Output:  0 0 0  0 0 0 |  |