**Лабораторная работа № 15. Динамическое выделение памяти**

Динамическая память  – это область памяти, которая выделяется во время *выполнения* программы. После использования памяти ее надо освобождать, что позволит эффективно расходовать память.

Формирование динамических массивов можно организовать двумя способами: с использованием функций (язык **С**); с использованием операторов **new** и **delete** (язык **С++**).

|  |  |
| --- | --- |
| **Задание** | **Краткие теоретические сведения** |
| 1. Изучить формирование динамического массива с использованием стандартных функций, выполнив программу на *языке* ***С***, записанную справа.  Написать пояснения к программе.  Добавить операторы вычисления максимального по модулю элемента массива. | Ниже представлены функции работы с динамической памятью и программа с их использованием.   |  |  | | --- | --- | | Функция | Прототип и краткое описание | | **malloc** | **void \*malloc(unsigned s) −** возвращает указатель на начало области динамической памяти длиной в **s** байт, при неудачном завершении возвращает NULL | | **calloc** | **void \*calloc(unsigned n, unsigned m) −** возвращает указатель на начало динамической памяти для размещения **n** элементов длиной по **m** байт, при неудачном завершении возвращает NULL | | **realloc** | **void \*realloc(void \*p, unsigned s) −** изменяет размер блока ранее выделенной динамической памяти до размера **s** байт, **р −** адрес начала изменяемого блока, при неудачном завершении возвращает NULL | | **free** | **void \*free(void p) −** освобождает ранее выделенный участок динамической памяти, **р −** адрес первого байта |   Пример. Программа выделяет динамическую память под массив из **n** элементов, вводит массив с клавиатуры и выводит его на экран.  **#include <stdio.h>**  **#include <stdlib.h>**  **#include <math.h>**  **void main()**  **{**  **int \*ptr, i, n, sum = 0;**  **printf("Input size of massiv, n<30 \n");**  **scanf("%d", &n);**  **if(!(ptr = (int\*)malloc(n\*sizeof(int)))) //выделение памяти и проверка,**  **{ //достаточно ли для нее места**  **puts("Not enough memory");**  **return;**  **}**  **for (i = 0; i < n; i++)**  **{**  **printf("Input element [%d]\n" , i + 1);**  **scanf("%d", ptr + i);**  **}**  **printf("\nMassiv: \n", i + 1);**  **for (i = 0; i < n; i++){**  **printf("%d ", \*(ptr + i));**  **sum = sum + abs(ptr[i]);**  **}**  **cout << sum;**  **free(ptr); //освобождение динамической памяти**  **}** | |
| 2. Изучить способы выделения динамической памяти для *одномерного* *массива*, выполнив программу на *языке* ***С++***, записанную в правой части.  Опробовать работу программы с разными значениями вводимых символов. | Операция **new** позволяет выделить и сделать доступным свободный участок памяти, размеры которого соответствуют типу данных. Оператор возвращает указатель на начало выделенного блока памяти. Операция **delete** освобождает участок памяти, ранее выделенный операцией **new**. Аргументом оператора **delete** выступает адрес первой ячейки блока, который необходимо освободить.  Пример. Пусть необходимо в строке, введенной с клавиатуры, подсчитать количество повторений символа, который также вводится с клавиатуры.    В программе при выделении динамической памяти количество элементов символьного массива **s** увеличено на 1, так как в конце массива должен присутствовать признак конца строки (нуль-байт). | |
| 3. В программе, записанной справа, демонстрируется использование динамической памяти при работе с *двумерным массивом*.  Выполнить программу с различными размерами исходного массива. | **#include <ctime>**  **#include <iostream>**  **using namespace std;**  **void main()**  **{ int size, mult = 1, sum = 0, \*\*A;**  **cout << "Enter size ";**  **cin >> size;**  **srand(time(0));**  **A = new int\*[size]; //выделение динам.памяти**  **for (int i = 0; i < size; i++)**  **A[i] = new int[size];**  **for (int j = 0; j < size; j++)**  **A[i][j] = 1 + rand() % 10;**  **for (int \*\*ptr = A; ptr != A + size; ++ptr)**  **{ for (int\* it = \*ptr; it != \*ptr + size; ++it)**  **{ cout << "\t" << \*it;**  **if (\*it % 2 == 0)**  **{ mult \*= \*it;**  **sum += \*it;**  **}**  **}**  **cout << endl;**  **}**  **cout << "sum: " << sum << endl;**  **cout << "mult: " << mult << endl;**  **for(int k=0; k < size; k++) //освобождение памяти**  **delete[] A[k];**  **delete[] A;**  **}**  Пример. Дана целочисленная матрица случайных чисел размерности **n** x **m**.  Вычислить сумму и произведение ее четных элементов. | |

4. В соответствии со своим вариантом написать программы с использованием ***динамических*** массивов для условий задач из таблицы. Начальные размерности массивов ввести с клавиатуры. Первое задание выполнить с использованием ***функций*** для выделения динамической памяти на языке С, второе − с использованием операторов **new** и **delete**  на языке **С++**.

| **№ варианта** | **Условия задач** |
| --- | --- |
| **10** | 1. В одномерном массиве, состоящем из n вещественных элементов, вычислить сумму положительных элементов массива и произведение элементов массива, расположенных между максимальным по модулю и минимальным по модулю элементами.     // В одномерном массиве, состоящем из n вещественных элементов, вычислить сумму положительных элементов массива и произведение  // элементов массива, расположенных между максимальным по модулю и минимальным по модулю элементами.  #include <stdio.h>  #include <iostream>  #include <stdlib.h>  #include<math.h>  void main() {  setlocale(LC\_ALL, "RU");  int\* ptr; int n, max = 0, min = 0,sum = 0,umn = 1, m = 0, h = 0;  printf("Введите размер массива: ");  scanf\_s("%d", & n);  if (n <= 0 || n >= 30) {  printf("Неккоректный размер\n");  return;  }  ptr = (int\*)malloc(n \* sizeof(int));  if (!ptr) {  puts("Недостаточно памяти");  return;  }  for (int i = 0; i < n; i++) {  scanf\_s("%d", ptr + i);  if (ptr[i] > 0) {  sum = sum + ptr[i];  }  if (abs(ptr[i]) > abs(max)) {  max = ptr[i];  h = i;  }  }  min = abs(ptr[0]);  for (int i = 0; i < n; i++) {  if (abs(ptr[i]) < abs(min)) {  min = ptr[i];  m = i;  }  }  if (m < h) {  m = m + 1;  for (m ;m < h; m++) {  umn = umn \* ptr[m];  }  }  else if (m > h) {  h = h + 1;  for (h; h < m; m++) {  umn = umn \* ptr[m];  }  }  printf("Сумма: %d \n" , sum);  printf("Произведение: %d \n", umn);  free(ptr);  }   1. Найти в матрице первую строку, все элементы которой отрицательны. Увеличить все элементы матрицы на значение первого элемента найденной строки.     // Найти в матрице первую строку, все элементы которой отрицательны. Увеличить все элементы матрицы на значение первого элемента найденной строки.  #include <iostream>  using namespace std;  bool rowNegative(int\* row, int n) {  for (int i = 0; i < n; i++) {  if (row[i] >= 0) {  return false;  }  }  return true;  }  int main() {  setlocale(LC\_ALL, "RU");  int n = 4;  int\*\* matrix = new int\* [n];  for (int i = 0; i < n; i++) {  matrix[i] = new int[n];  }  cout << "Матрица размером" << n << "x" << n << ":" << endl;  for (int i = 0; i < n; i++) {  for (int j = 0; j < n; j++) {  cin >> matrix[i][j];  }  }  int firstNegativeRow = -1;  int firstElement = 0;  for (int i = 0; i < n; i++) {  if (rowNegative(matrix[i], n)) {  firstNegativeRow = i;  firstElement = matrix[i][0];  break;  }  }  if (firstNegativeRow != -1) {  cout << "Первая негативная строка: " << firstNegativeRow << endl;  cout << "Первый элемент строки: " << firstElement << endl;  for (int i = 0; i < n; i++) {  for (int j = 0; j < n; j++) {  matrix[i][j] += firstElement;  }  }  cout << "Измененная матрица:" << endl;  for (int i = 0; i < n; i++) {  for (int j = 0; j < n; j++) {  cout << matrix[i][j] << " ";  }  cout << endl;  }  }  else {  cout << "Нет негативной строки" << endl;  }  for (int i = 0; i < n; i++) {  delete[] matrix[i];  }  delete[] matrix;  return 0;  } |
| **11** | // Задан массив A из n элементов. Проверить, есть ли в нём элементы, равные нулю. Если есть, найти наименьшее k, при котором A[k] = 0.  #include <stdio.h>  #include <iostream>  #include <stdlib.h>  void main() {  setlocale(LC\_ALL, "RU");  int n;  int\* A;  printf("Введите размер массива, n < 30: ");  scanf\_s("%d", & n);  if (n <= 0 || n >= 30) {  printf("Неккоректный размер.\n");  return;  }  A = (int\*)malloc(n \* sizeof(int));  if(!A)  {  puts("Недостаточно памяти");  return;  }  printf("Элементы равные нулю: ");  for (int k = 1; k <= n; k++) {  scanf\_s("%d", A + k);  if (A[k] == 0) {  printf("Наименьшее K = %d", k);  break;  }  }  free(A);  }  1. Задан массив **A** из **n** элементов. Проверить, есть ли в нём элементы, равные нулю. Если есть, найти наименьшее **k**, при котором **A[k] = 0**.    2. Для заданной матрицы размером 4 на 4 найти такие **k**, при которых **k**-я строка матрицы совпадает с **k**-м столбцом.  // Для заданной матрицы размером 4 на 4 найти такие k, при которых k-я строка матрицы совпадает с k-м столбцом.  #include<iostream>  using namespace std;  bool checkMatrix(int\*\* matrix, int n, int k) {  for (int i = 0; i < n; i++) {  if (matrix[k][i] != matrix[i][k]) {  return false;  }  }  return true;  }  int main() {  setlocale(LC\_CTYPE, "RU");  int n = 4;  int\*\* matrix;  matrix = new int\* [n];  for (int i = 0; i < n; i++) {  matrix[i] = new int[n];  }  cout << "Матрица " << n << "x" << n << ":" << endl;  for (int i = 0; i < n; i++) {  for (int j = 0; j < n; j++) {  cin >> matrix[i][j];  }  }  for (int k = 0; k < n; k++) {  if (checkMatrix(matrix, n, k)) {  cout << "При k = " << k+1 << " совпадает" << endl;  }  else  {  cout << "При k = " << k+1 << " не совпадает" << endl;  }  }  for (int i = 0; i < n; i++) {  delete[] matrix[i];  }  delete[] matrix;  return 0;  } |
| **2** | 1. Задан массив **A** из **n** элементов. Проверить, есть ли в нём отрицательные элементы. Если есть, то найти наибольшее значение **k**, при котором **A[k] < 0**.   // Задан массив A из n элементов. Проверить, есть ли в нём отрицательные элементы. Если есть, то найти наибольшее значение k, при котором A[k] < 0.  #include <stdio.h>  #include <iostream>  #include <stdlib.h>  void main() {  setlocale(LC\_ALL, "RU");  int n;  int f = 0;  int\* A;  printf("Введите размер массива, n < 30: ");  scanf\_s("%d", &n);  if (n <= 0 || n >= 30) {  printf("Неккоректный размер.\n");  return;  }  A = (int\*)malloc(n \* sizeof(int));  if (!A)  {  puts("Недостаточно памяти");  return;  }  printf("Введите элементы массива: ");  for (int k = 1; k <= n; k++) {  scanf\_s("%d", A + k);  if (A[k] < 0) {  if (k > f) {  f = k;    }  }  }  printf("Наибольшее K = %d", f);  free(A);  }    // Дана матрица. Переставляя ее строки и столбцы, добиться того, чтобы наибольший элемент оказался в верхнем левом углу.  #include <iostream>  #include <vector>  #include <cstdlib>  #include <ctime>  using namespace std;  void findMaxElement(int\*\* matrix, int rows, int cols, int& maxRow, int& maxCol) {  int maxElement = matrix[0][0];  maxRow = 0;  maxCol = 0;  for (int i = 0; i < rows; ++i) {  for (int j = 0; j < cols; ++j) {  if (matrix[i][j] > maxElement) {  maxElement = matrix[i][j];  maxRow = i;  maxCol = j;  }  }  }  }  void swapRows(int\*\* matrix, int row1, int row2, int cols) {  for (int j = 0; j < cols; ++j) {  swap(matrix[row1][j], matrix[row2][j]);  }  }  void swapCols(int\*\* matrix, int col1, int col2, int rows) {  for (int i = 0; i < rows; ++i) {  swap(matrix[i][col1], matrix[i][col2]);  }  }  int main() {  setlocale(LC\_ALL, "RU");  int rows, cols;  cout << "Введите количество строк и столбцов ";  cin >> rows >> cols;  int\*\* matrix = new int\* [rows];  for (int i = 0; i < rows; ++i) {  matrix[i] = new int[cols];  }  srand(time(0));  cout << "Матрица:" << endl;  for (int i = 0; i < rows; ++i) {  for (int j = 0; j < cols; ++j) {  matrix[i][j] = rand() % 10;  cout << matrix[i][j] << " ";  }  cout << endl;  }  int maxRow, maxCol;  findMaxElement(matrix, rows, cols, maxRow, maxCol);  swapRows(matrix, 0, maxRow, cols);  swapCols(matrix, 0, maxCol, rows);  cout << "Измененная матрица:" << endl;  for (int i = 0; i < rows; ++i) {  for (int j = 0; j < cols; ++j) {  cout << matrix[i][j] << " ";  }  cout << endl;  }  for (int i = 0; i < rows; ++i) {  delete[] matrix[i];  }  delete[] matrix;  return 0;  }   1. Дана матрица. Переставляя ее строки и столбцы, добиться того, чтобы наибольший элемент оказался в верхнем левом углу. |
| **13** | 1. В одномерном массиве, состоящем из вещественных элементов, вычислить сумму элементов массива с нечетными номерами и сумму элементов массива, расположенных между первым и последним отрицательными элементами.   // В одномерном массиве, состоящем из вещественных элементов, вычислить сумму элементов массива с нечетными номерами и  // сумму элементов массива, расположенных между первым и последним отрицательными элементами.  #include <stdio.h>  #include <iostream>  #include <stdlib.h>  #include<math.h>  void main() {  setlocale(LC\_ALL, "RU");  int\* ptr; int n, max = 0, min = 0, sum = 0, umn = 1, m = 0, h = 0; int null = 0;  printf("Введите размер массива: ");  scanf\_s("%d", &n);  if (n <= 0 || n >= 30) {  printf("Неккоректный размер\n");  return;  }  ptr = (int\*)malloc(n \* sizeof(int));  if (!ptr) {  puts("Недостаточно памяти");  return;  }  for (int i = 0; i < n; i++) {  scanf\_s("%d", ptr + i);  if (null = i%2) {  sum = sum + ptr[i];  }  if (abs(ptr[i]) > abs(max)) {  max = ptr[i];  h = i;  }  }  min = abs(ptr[0]);  for (int i = 0; i < n; i++) {  if (abs(ptr[i]) < abs(min)) {  min = ptr[i];  m = i;  }  }  if (m < h) {  m = m + 1;  for (m; m < h; m++) {  umn = umn + ptr[m];  }  }  else if (m > h) {  h = h + 1;  for (h; h < m; m++) {  umn = umn + ptr[m];  }  }  printf("Сумма нечетных элементов массива: %d \n", sum);  printf("Сумма между первым и последним отрицательным элементом: %d \n", umn);  free(ptr);  }     1. Найти в матрице первую строку, все элементы которой равны нулю. Все элементы столбца с таким же номером уменьшить вдвое.   // Найти в матрице первую строку, все элементы которой отрицательны. Увеличить все элементы матрицы на значение первого элемента найденной строки.  #include <iostream>  using namespace std;  bool rowNegative(int\* row, int n) {  for (int i = 0; i < n; i++) {  if (row[i] == 0) {  return true;  }  }  return false;  }  int main() {  int n = 4;  int\*\* matrix = new int\* [n];  for (int i = 0; i < n; i++) {  matrix[i] = new int[n];  }  setlocale(LC\_ALL, "Russian");  cout << "Матрица размером " << n << "x" << n << ":" << endl;  for (int i = 0; i < n; i++) {  for (int j = 0; j < n; j++) {  cin >> matrix[i][j];  }  }  int firstNegativeRow = -1;  int firstElementIndex = 0;  for (int i = 0; i < n; i++) {  if (rowNegative(matrix[i], n)) {  firstNegativeRow = i;  firstElementIndex = i + 1;  break;  }  }  if (firstNegativeRow != -1) {  cout << "Первая негативная строка: " << firstNegativeRow << endl;  cout << "Номер первого элемента строки: " << firstElementIndex << endl;  for (int i = 0; i < n; i++) {  for (int j = 0; j < n; j++) {  matrix[i][j] += firstElementIndex;  }  }  cout << "Измененная матрица:" << endl;  for (int i = 0; i < n; i++) {  for (int j = 0; j < n; j++) {  cout << matrix[i][j] << " ";  }  cout << endl;  }  }  else {  cout << "Нет негативной строки" << endl;  }  for (int i = 0; i < n; i++) {  delete[] matrix[i];  }  delete[] matrix;  return 0;  } |