**Практическая работа № 17**

**Тема: Реализация программ, использующих двоичный поиск в массиве.**

**Цель**: Изучить средства языка программирования для управления динамической памятью массива. Изучить принципы алгоритмизации и разработки программ, связанных с обработкой массивов.

**Что такое динамические переменные**

**Динамические переменные** — **это переменные, которые созданы напрямую с помощью указателей. Для них существует функция удаление.**

Чтобы мы могли полноценно создавать динамические переменные, нам понадобится изучить **конструктор — new**, после его использования в оперативной памяти компьютера выделяются ячейки на тот тип данных, который мы указали.

На каждый тип данных выделяется разное количество ячеек.

**Как создать динамические переменные в C++**

Для создания динамических переменных нам понадобится применять конструкцию ниже:

|  |  |
| --- | --- |
| **1** | **<тип данных указателя> \*<имя указателя> = new <тип данных>(<первоначальное значение>);** |

Давайте подробно ее разберем:

* <тип данных указателя> — указанный тип данных почти ни на что не повлияет.
* new — это конструктор, который и будет заключительным звеном для создания нашей переменной.
* <тип данных> — здесь нам понадобится указать тип, какой будет храниться в переменной. Он необязательно должен совпадать с типом указателя.
* <первоначальное значение> — с помощью круглых скобок можно указать значение переменной еще при ее инициализации. Использование круглых скобок в этой конструкции необязательно.

Пример использования динамических переменных

**Задача 1**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | #include <iostream>    using namespace std;    int main() {    setlocale(0, "");    int \*a = new int;      int b = 10;      \*a = b;      cout <<"Теперь переменная a равна "<< \*a << endl;      cout <<"Пришло время удалить эту переменную!";      system("pause");    return 0;  } |

* **В строке 7:** мы объявили переменную, оперируя конструктором new.
* Дальше **в строке 11:** значение нашей переменной становится равно 10.
* И в самом конце, **в строке 15:** выводим значение нашей переменной на экран.

Важно помнить! Динамические переменные — это указатели, и поэтому перед ними обязательно должен стоять **оператор \*.**

**Удаление динамических переменных**

Как мы говорили выше, у нас есть возможность освобождать память переменной или, если понятным языком, удалять переменную из оперативной памяти ПК.

Конечно, эта переменная и так удалится из оперативной памяти компьютера при завершении программы. Но если нам захотелось удалить ее еще в середине программы, то это будет возможно благодаря оператору delete.

|  |  |
| --- | --- |
| **1** | **delete <имя переменной>;** |

* В самом начале мы используем оператор delete.
* Дальше идет имя переменной.

Вы должны обратить внимание **на отсутствие оператора \*** перед именем переменной.

**Статическое и динамическое объявление переменных**

Статическое объявление переменных имеет такой вид: **int number;**

Использование динамических переменных имеет маленький плюс. Он заключается в освобождении памяти переменной до завершения программы. Благодаря этому мы можем сначала удалить переменную, а потом ее снова создать в другом участке программы (когда это нам будет нужно).

Например, пользователь захотел вписать 1000 чисел в массив, а мы из-за незнания этого факта сделали массив всего лишь на 500 ячеек.

**Динамический массив** — это массив, у которого количество ячеек можно задавать и переменной, и числовой константой. Это большой плюс перед использованием статического массива.

**Как работают динамические массивы**

Для работы динамических массивов нам понадобится при инициализации [указатель](https://codelessons.ru/cplusplus/ukazateli-v-c-podrobnoe-rukovodstvo.html) (всего лишь при инициализации!) и уже знакомый конструктор new.

**Как создать динамический массив в C++**

Чтобы **создать динамический массив**мы будем использовать конструкцию ниже:

|  |  |
| --- | --- |
| **1** | **<тип данных> \*<имя массива> = new <тип переменных> [<количество ячеек>];** |

* <тип данных> — без разницы какой тип данных тут будет находиться, но лучше тот, который будет совпадать с типом переменных.
* <тип переменных> — указанный сюда тип и будут иметь ячейки массива.
* <количество ячеек> — здесь мы задаем размер массива (например [n] или [25]).

Динамический массив полностью идентичен обычному массиву, кроме:

* Своей инициализации
* Возможностью своевременно освободить память.

**Задача 2**

|  |  |
| --- | --- |
| 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 | int main() {    setlocale(0, "");      int n;      cout << "Введите количество чисел, которое вы хотите ввести: ";    cin >> n;      cout << "Введите " << n << " чисел: ";      int \*dinamich\_array = new int [n];    // создаем                                          // динамический массив    for (int i = 0; i < n; i++) {      cin >> dinamich\_array[i];  // считываем числа в ячейки массива    }      cout << "Теперь давайте выведем элементы массива в обратном порядке: ";      for (int i = n - 1 ; i >= 0; i--) {      cout << dinamich\_array[i] << " ";  // выводим значение всех ячеек    }      cout << endl << "Удаляем массив!";      delete [] dinamich\_array;  // удаляем динамический массив      return 0;  } |

**Вот что будет при выполнении программы:**

**Задайте количество чисел, которое вы хотите ввести: 5**

**Введите 5 чисел: 2 4 6 8 16**

**Теперь давайте выведем элементы массива в обратном порядке: 16 8 6 4 2**

**Удаляем массив!**

**Press any key to continue.**

**Удаление динамического массива**

Для удаления динамического массива нам понадобится уже знакомый оператор — delete.

|  |  |
| --- | --- |
| 1 | **delete [] <имя массива>;** |

Важно запомнить, что **квадратные скобки нужно ставить перед <именем массива>.**

**Как создать двумерный динамический массив в C++**

Для создания двумерного динамического массива мы будем использовать похожую конструкцию (как и в одномерном динамическом массиве):

|  |  |
| --- | --- |
| **1** | **<тип данных> \*\*<имя массива> = new <тип данных массива>\* [<количество ячеек>];** |

Вам нужно обратить внимание на:

* Дополнительный оператор **\* перед <имя массива> и после <тип данных массива>.**

Дальше для каждой ячейки мы должны создать одномерный массив. Чтобы это сделать, нам понадобится цикл [for](https://codelessons.ru/cplusplus/syntax/cikl-for-v-c-izuchaem-s-primerami.html) и конструктор new.

|  |  |
| --- | --- |
| 1  2  3  4  5 | for (int i = 0; i < n; i++) {      <имя массива>[i] = new <тип ячеек> [<количество ячеек>];    } |

В <количество ячеек> можно задавать разные значения. Поэтому сначала для первого массива можно задать длину 1 (new int [1]), потом для второго — длину 2 (new int [2]), как в примере ниже.

Внизу находится пример двумерного динамического массива:

Задача 3

|  |  |
| --- | --- |
| 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 | #include <iostream>    using namespace std;    int main() {    setlocale(0, "");      int \*\*dinamic\_array2 = new int\* [5];   // создаем    for (int i = 0; i < 5; i++) {          // двумерный      dinamic\_array2[i] = new int [i + 1]; // массив    }                                      // !      for (int i = 0; i < 5; i++) {      cout << "Введите числа" << "(" << i + 1 << ")" << ":";      for (int j = 0; j < i + 1; j++) {        cin >> dinamic\_array2[i][j];      }    }      for (int i = 0; i < 5; i++) {      int sum = 0;      for (int j = 0; j < i + 1; j++) {        sum += dinamic\_array2[i][j];      }      cout << "Сумма " << i + 1 << " массива равна " << sum << endl;    }      for (int i = 0; i < 5; i++) {      delete [] dinamic\_array2[i];  // удаляем массив    }      system("pause");    return 0;  } |

* **В строках 8 — 11:** создали двумерный динамический массив.
* **В строках 13 — 18:** заполнили массив.
* **В строках 20 — 26:** подсчитали и вывели по отдельности на экран сумму всех массивов.
* **В строках 28 — 30:** происходит удаление массива (об этом ниже).

**Удаление двумерного динамического массива**

Для удаление двумерного динамического массива мы будем использовать уже похожую схему. Но в ней присутствует цикл for, и после <имя массива> находится индекс того массива который будет удален.

|  |  |
| --- | --- |
| 1  2  3  4  5 | **for (int i = 0; i < <количество элементов в массиве>; i++) {**    **delete [] <имя массива>[i];**    } |

**Задания для самостоятельного решения**

**Задача №1,№2 (на 4)**

Разработаем программу, в которой создадим одномерный динамический массив,

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | // new\_delete\_array.cpp: определяет точку входа для консольного приложения.    #include "stdafx.h"  #include <iostream>  // в заголовочном файле <ctime> содержится прототип функции time()  #include <ctime>  // в заголовочном файле <iomanip> содержится прототип функции setprecision()  #include <iomanip>  using namespace std;    int main(int argc, char\* argv[])  {      srand(time(0)); // генерация случайных чисел  **float \*ptrarray = new float [10];** // создание динамического массива вещественных чисел на десять элементов  **for (int count = 0; count < 10; count++)**  **ptrarray[count] = (rand() % 10 + 1) / float((rand() % 10 + 1));** //заполнение массива случайными числами с масштабированием от 1 до 10          cout << "array = ";          for (int count = 0; count < 10; count++)              cout << setprecision(2) << ptrarray[count] << "    ";  **delete [] ptrarray; // высвобождение памяти**          cout << endl;      system("pause");      return 0;  } |

Созданный одномерный динамический массив заполняется случайными вещественными числами, полученными c помощью функций генерации случайных чисел, причём числа генерируются в интервале от 1 до 10, интервал задается так — rand() % 10 + 1**.**Чтобы получить случайные вещественные числа, выполняется операция деления, с использованием явного приведения к вещественному типу знаменателя — float((rand() % 10 + 1)). Чтобы показать только два знака после запятой используем функцию setprecision(2)**,**прототип данной функции находится в заголовочном файле<iomanip>**.**Функция **time(0)** засевает генератор случайных чисел временным значением, таким образом, получается, воспроизводить случайность возникновения чисел

Результат:

**array = 0.8 0.25 0.86 0.5 2.2 10 1.2 0.33 0.89 3.5**

**Для продолжения нажмите любую клавишу . . .**

**Задача №2**

Создадим двумерный массив произвольного размера, с возможностью задавать этот размер при помощи клавиатуры. Заполним его случайными числами и выведем на экран:

#include <iostream>

#include <**conio.h**>

#include <iomanip>

#include <**time.h**>

using namespace std;

int get\_lenght(int buffer[]);

int main(int argc, char\* argv[])

{

// Русский язык в консольном приложении

setlocale(LC\_ALL, "Rus");

int rows, columns;

srand(time(NULL));

// наш контейнер с яблоками в каждой из

// секций которой лежит определенное количество яблок

cout << "Введите размер строк" << endl;

cin >> rows;

cout << "Введите размер столбцов" << endl;

cin >> columns;

int\*\* apples = new int\* [rows];

for (size\_t i = 0; i < rows; i++) {

apples[i] = new int [columns];

}

// Заполняем массив случайнми числами

for (size\_t i = 0; i < rows; i++) {

for (size\_t j = 0; j < columns; j++) {

apples[i][j] = rand() % 10 + 1;

}

}

cout << "Выводим массив на экран: " << endl;

// Это цикл, в котором мы идем и заглядываем в каждую секцию

// после чего печатаем количество каждой секции на экран

// секции задаются вводом с клавиатуры, это наш размер массива (строки и столбцы)

// **setw(4) это форматирование строки на экране, 4 - это примерно**

**// 4 пробела между символами.**

for (size\_t i = 0; i < rows; i++) {

for (size\_t j = 0; j < columns; j++) {

cout << setw(4) << apples[i][j];

}

cout << endl;

}

// Освобождаем память

for (size\_t i = 0; i < rows; i++) {

**delete[] apples[i];**

}

cout << endl;

\_**getch();** // Задержка.

return 0;

}

**Задача №3,№4 (на 5)**

Найти сумму (произведение) вещественных элементов динамического массива.

|  |  |
| --- | --- |
| 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 | //Пример использования функции **malloc и free**  #include "stdafx.h"  #include <iostream>  using namespace std;  int main()  {  setlocale(LC\_ALL,"Rus");  int i, n;  float \*a; //указатель на float  float s;  cout<<"\n"; cin>>n; //ввод размерности массива  //выделение памяти под массив из n вещественных элементов  a=(float \*)malloc(n\*sizeof(float));  cout<<"Введите массив A \n";  //ввод элементов массива  for (i=0; i<n; i++)  {  cin>>\*(a+i);  }  //накапливание суммы элементов массива  for (s=0, i=0; i<n; i++)  s+=\*(a+i);  //вывод значения суммы  cout<<"S="<<s<<"\n";  //освобождение памяти  free(a);  system("pause");  return 0;  } |

**Задача 4**

Изменить динамический массив целых чисел таким образом, чтобы его положительные элементы стали отрицательными и наоборот. Для решения задачи мы будем умножать каждый элемент на -1.

|  |  |
| --- | --- |
| 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 | //Пример использования операторов new и delete  #include "stdafx.h"  #include <iostream>  using namespace std;  int main()  {  setlocale(LC\_ALL,"Rus");  int i, n;  //ввод количества элементов массива  cout<<"n="; cin>>n;  //выделение памяти  int \*a=new int[n];  cout<<"Введите элементы массива:\n";  //ввод массива  for (i=0; i<n; i++)  cin>>a[i];  //вывод заданного массива  for (i=0; i<n; i++)  cout<<a[i]<<"\t"<<"\n";  //преобразование массива  for (i=0; i<n; i++)  a[i]=-a[i];  //вывод преобразованного массива  for (i=0; i<n; i++)  cout<<a[i]<<"\t"; cout<<"\n";  //освобождение памяти  delete []a;  system("pause");  return 0;  } |

**Дополнительная задача №5**

**Двумерный массив треугольной формы**

[+3] Создайте с помощью операторов new двумерный массив, в

нулевой строке которого 1 элемент, в первой строке 2 элемента,

во второй строке 3 элементов, и так далее, вплоть до 10 строки, в

которой 11 элементов. В нулевую строку занесите число 1, в

первую числа 1, и 1, во вторую 1, 2, 1, в третью 1, 3, 3, 1, в чет-

вертую 1, 4, 6, 4, 1, в пятую 1, 5, 10, 10, 5, 1, и т.д. Напечатайте

массив в виде таблицы треугольной формы.

Решение

{

myofs << "Двумерный треугольный массив." << endl;

int M=8;

int\*\* x = new int\*[M];

for(int m=0; m<M; m++) x[m]=new int[m+1];

x[0][0]=1;

x[1][0]=1; x[1][1]=1;

for(int m=2; m<M; m++)

{

x[m][0]=1; x[m][m]=1;

for(int n=1; n<m; n++)

x[m][n]=x[m-1][n]+x[m-1][n-1];

}

for(int m=0; m<M; m++){

myofs << "[" << setw(2) << m << "] ";

for(int n=0; n<=m; n++)

myofs << x[m][n] << " ";

myofs << endl;

}

myofs << endl;

for(int m=0; m<M; m++){

if(x[m]) delete[]x[m];

x[m]=NULL;

}

if(x){ delete[]x; x=NULL; }

}

Результат в файле протокола:

Двумерный треугольный массив.

[ 0] 1

[ 1] 1 1

[ 2] 1 2 1

[ 3] 1 3 3 1

[ 4] 1 4 6 4 1

[ 5] 1 5 10 10 5 1

[ 6] 1 6 15 20 15 6 1

[ 7] 1 7 21 35 35 21 7 1

[ 8] 1 8 28 56 70 56 28 8 1

**Для всех**

**Тест на тему**

**«Динамические массивы и переменные». Проверь себя!**

|  |
| --- |
| **Вопрос 1** |

На какой из видов инициализации переменной, компилятор не будет ругаться?

|  |  |  |
| --- | --- | --- |
| A | double array = 0; | |
| B | int a = 5.15; | |
| C | int \*num = new dauble; | |
| D | int \*digital = new int; | |
| **Вопрос 2** | |

Какая библиотека нужна для создания динамических переменных и массивов? — кроме iostream.

|  |  |
| --- | --- |
| A | cmath |
| B | iterator |
| C | vector |
| D | никакой |

|  |
| --- |
| **Вопрос 3** |

Как удалять динамические массивы?

|  |  |  |
| --- | --- | --- |
| A | delete <имя массива> | |
| B | delite [] <имя массива> | |
| C | delete <имя массива>[] | |
| D | delete [] <имя массива> | |
| **Вопрос 4** | |

В чем плюс использования динамического массива вместо обычного?

|  |  |
| --- | --- |
| A | В дин. массиве можно указать больше ячеек, чем для обычного. |
| B | У дин. массива инициализация происходит быстрее, чем у обычного. |
| C | Дин. массив можно удалить. |
| D | Для дин.массива кол-во ячеек можно указать переменной. |