Язык программирования Си++

Иванов А.П., Князева О.С.

Семинар 2. Массивы объектов. Массивы переменной длины. Операторы new, delete. Указатели. Ссылки.

1. Массивы объектов

Встроенное в язык Си++ понятие массива позволяет создавать сразу заданное количество любых описанных ранее объектов. При этом для каждого элемента массива будет вызван его конструктор. При выходе из области видимости этого массива деструкторы также будут вызваны для каждого его элемента. Эти вызовы осуществляются неявно — компилятор вставляет в нужные места соответствующий код.

Доступ к одному из элементов массива может осуществляться как по индексу, так и по указателю.

```
class CPoint { // описание класса "точка"
public:
  int x,y;   // координаты точки
  ~CPoint() { cout << x << " " << y << endl; }          // деструктор
  void Set( int newX=0, int newY=0 ) { x=newX,y=newY; } // установка значений
  CPoint( int newX=0, int newY=0 ) { Set(newX,newY); }  // конструктор
};

...
{
// начало области видимости массива
  const int N=12;    // объявление константы, задающей размерность массива
  CPoint PtArray[N]; // объявление массива объектов - "точек"
// здесь неявно будут вызваны конструкторы "точек"
  CPoint * pPoint;   // объявление указателя на "точку"
  ...
// доступ по индексу
  PtArray[3].Set(12,24); // установка новых значений 3-го элемента
  PtArray[4].x=15, PtArray[4].y=28; // изменение 4-го элемента

// доступ по адресу
  pPoint = PtArray + 5; // получение адреса 5-го элемента
  pPoint->x = 16; pPoint->y = 32; // изменение 5-го элемента
  pPoint += 4; // получение адреса 9-го элемента
  pPoint->Set(2,2); // установка новых значений 9-го элемента
...
// здесь перед выходом из области видимости будут вызываться деструкторы
}
...
```

2. Операторы new и delete

В языке Си все переменные могли размещаться в памяти несколькими способами. Переменные первого типа размещаются в памяти в момент загрузки и запуска программы и находятся там конца ее выполнения. Такие переменные называются статическими. Переменные второго типа создаются при входе в блок и существуют до выхода из него. Такие переменные называются автоматическими.

Есть и третий вид размещения переменных в памяти – размещение в динамической памяти, с которой можно было работать при помощи функций `malloc()/realloc()/free()`. В язык Си++ также встроена возможность создавать объекты в динамической памяти, время жизни которых не регламентировано так строго, как в случае статических или автоматических переменных. Эта возможность реализована в виде операторов `new` и `delete`, используемых в соответствии со схемой:

```
CPoint *pPoint = new CPoint(1,3); // создание объекта заданного типа
.............       // использование созданного объекта
delete pPoint;      // уничтожение ранее созданного объекта
```

По действию оператора new в свободной памяти выделяется место для нового представителя объектов указанного типа и вызывается соответствующий конструктор. По действию оператора `delete` вызывается деструктор того объекта, на который ссылается указатель и после этого освобождается занятая память. Уничтожать оператором `delete` можно только те объекты, которые были созданы посредством оператора `new`.

Особенности:

- Аналога функции `realloc()` в Си++ нет.
- Допускается вызов `delete p;` где `p` – нулевой указатель. Такой вызов просто ничего не делает.

3. Массивы переменной длины

Создание обычных статических или автоматических переменных-массивов имеет два существенных недостатка: во-первых, для описания длины массива не может быть использована переменная величина – допускается только константа, а во-вторых, если массив был объявлен внутри некоторой функции, то он будет автоматически уничтожен при выходе из этой функции. Часто же возникают ситуации, когда созданный внутри некоторого модуля массив должен иметь длину задаваемую переменным параметром, и кроме того должен продолжить свое существование после возврата управления из этого модуля. В таких ситуациях следует завести объект в динамической памяти.

```
// функция, создающая массив "точек" заданного объема N
CPoint * CreatePtArray( int N )
{
  CPoint * pPoint;
  pPoint = new CPoint [N]; // создание массива переменной длины
// явно будут вызваны N конструкторов по умолчанию
  return pPoint; // возврат адреса созданного массива
// неявного вызова деструкторов не будет, так как уничтожается только
// переменная-указатель, а не весь массив.
}
...
{
// использование массива переменной длины
  int nDim=10;
  CPoint * pPtArray;
...
  pPtArray = CreatePtArray( nDim );
// оператор new может вернуть NULL, при нехватке памяти
  if( pPtArray!=NULL ) {
    for( int i=0 ; i<nDim ; i++ ) {
      pPtArray[i].Set(i,i); // инициализация массива
    }
```

```
    // работа с массивом...
    ...
    // по окончании работы массив следует уничтожить явно
    delete [] pPtArray; // [] - означают уничтожение массива
  }
}
```

4. Указатели и ссылки

Как известно, в языке Си, передача параметров в функцию при вызове может осуществляться двумя способами:

- передача по адресу;
- передача по значению.

В языке Си++ добавлено новое понятие – ссылка, и еще один способ передачи данных – по ссылке. Ссылка по смыслу является просто другим именем уже существующего объекта. Отличительной особенностью ссылки является то, что она обязана быть проинициализирована одновременно с ее объявлением (т.к. ссылка – имя уже существующего объекта).

По способу ее использования ссылка очень похожа на новое имя объекта (его псевдоним), но по свойствам, которые предоставляет ссылка, она больше похожа на указатель. При передаче параметра по ссылке модуль, в который эта ссылка передана, получает доступ непосредственно к самому объекту, а не к его копии, как это происходит при передаче по значению. Дополнительные возможности появляются и при использовании ссылок в качестве возвращаемого значения функции. Так, например, функция, возвращающая ссылку, может использоваться в левой части оператора присваивания – присвоение будет сделано тому объекту, ссылку на который вернула эта функция.

5. Пример. Контейнерный класс.

Устройство класса «вектор» (динамический массив) иллюстрирует весь материал семинара – создание динамических массивов, работу со ссылками и указателями и т.д.

```
#include <iostream>
#include <stdlib.h>

class Vector
{
protected:
int *v;
int len;
public:
//==========Конструкторы объектов класса======//
// Конструктор
      Vector(int N=0) : v(0), len(0)
            { if (N>0) v = new int[N];
              if (v) len = N;
            }

// Конструктор копирования – конструктор, создающий объект (типа vector)
// по уже имеющемуся объекту того же класса
      Vector(const Vector & temp) : v(0), len(0)
            { if (temp.len>0) v=new int [temp.len];
              if ( v ) {
                  len=temp.len;
                  for (int i=0;i<len;i++) {v[i]=temp.v[i];}
```

```
            }
          }

//=============Размерность==========//
int size() const { return len; }

//=============Перегрузка []==========//
// для использования в левой части присваивания:
int & operator [] (int index)
{ return v[index]; }

// для использования в правой части присваивания:
int operator [] (int index) const
{ return v[index]; }

//=============Деструктор===========//
// Определение деструктора ~имя_класса () {тело деструктора}
      ~ Vector() { delete [] v; }
};
...
{
  Vector T(100);
  T[0] = 3;
  int x = T[0]++;
  int y = T[0];
}
...
```

Типовое задание: реализовать класс – динамический массив объектов с возможностью добавления (в произвольное место) и удаления любого элемента. Должны быть реализованы: конструктор копирования и деструктор, освобождающий память, перегружен оператор `[]`, а также оператор вывода в поток.

Для вставки нового элемента в динамический массив (вектор) нужно:

- выделить память большего размера и сохранить указатель во временной переменной, проверить результат (выделилось ли?);
- скопировать по адресу нового указателя все элементы старого вектора, предшествующие индексу вставляемого элемента;
- сохранить в нужную позицию нового массива вставляемый элемент;
- скопировать по адресу нового указателя за вставленным элементом все элементы старого массива, начинающиеся с индекса вставляемого элемента;
- уничтожить старый массив, на который продолжает указывать переменная-член класса v из примера выше;
- скопировать указатель нового вектора в переменную-член класса v из примера выше;
- размерность вектора увеличить на единицу.

1. Вариант

Создайте **класс – динамический массив**. В качестве элементов массива выберите **объекты из Задания № 1**.

Определите в нем конструктор (можно несколько), деструктор, конструктор копирования. Перегрузите операцию []. Определите методы класса, которые позволяли бы добавлять и удалять элемент из массива (insert, erase). Функция insert – добавляет элемент в конец массива. Функция erase – принимает индекс элемента, который нужно удалить из массива. Кроме того, определите метод класса (merge), позволяющий объединять два массива (второй добавляется в конец первого).

2. Вариант

Создайте **класс – динамический массив**. В качестве элементов массива выберите **объекты из Задания № 1**.

Определите в нем конструктор (можно несколько), деструктор, конструктор копирования. Перегрузите операцию []. Определите методы класса, которые позволяли бы добавлять и удалять элемент из массива (insert, erase). Функция insert – добавляет элемент в начало массива. Функция erase – принимает индекс элемента, который нужно удалить из массива. Кроме того, определите метод класса, позволяющий складывать два массива.

3. Вариант

Создайте **класс – динамический массив**. В качестве элементов массива выберите **объекты из Задания № 1**.

Определите в нем конструктор (можно несколько), деструктор, конструктор копирования. Перегрузите операцию []. Определите методы класса, которые позволяли бы добавлять и удалять элемент из массива (insert, erase). Функция insert – добавляет элемент в конец массива. Функция erase – принимает значение элемента, который нужно удалить из массива, если таких элементов больше чем 1, нужно удалить их все.

4. Вариант

Создайте **класс – динамический массив**. В качестве элементов массива выберите **объекты из Задания № 1**.

Определите в нем конструктор (можно несколько), деструктор, конструктор копирования. Перегрузите операцию []. Определите методы класса, которые позволяли бы добавлять и удалять элемент из массива (insert, erase). Функция insert – принимает индекс элемента, после которого нужно вставить новый. Функция erase – принимает индекс элемента, который нужно удалить из массива. Кроме того, определите метод класса – reverse (переставляет элементы в массиве в обратном порядке).

5. Вариант

Создайте **класс – динамический массив**. В качестве элементов массива выберите **объекты из Задания № 1**.

Определите в нем конструктор (можно несколько), деструктор, конструктор копирования. Перегрузите операцию []. Определите методы класса, которые позволяли бы добавлять и удалять элементы из массива (insert, erase). Функция insert – добавляет элемент в конец массива. Функция erase – принимает индекс элемента, с которого нужно начать удаление и количество элементов, которых нужно удалить из массива.

6. Вариант

Создайте **класс – динамический массив**. В качестве элементов массива выберите **вещественные числа**.

Определите в нем конструктор (можно несколько), деструктор, конструктор копирования. Перегрузите операцию []. Определите методы класса, которые позволяли бы добавлять и удалять элемент из массива (insert, erase). Функция insert – добавляет элемент в конец массива. Функция erase – принимает индекс элемента, который нужно удалить из массива. Кроме того, определите метод класса – sort (сортирует массив, реализуйте возможность выбора - сортировать по убыванию или по возрастанию).

7. Вариант

Создайте **класс – динамический массив**. В качестве элементов массива выберите **вещественные числа**.

Определите в нем конструктор (можно несколько), деструктор, конструктор копирования. Перегрузите операцию []. Определите методы класса, которые позволяли бы добавлять и удалять элемент из массива (insert, erase). Функция insert – добавляет элемент в начало массива. Функция erase – принимает индекс элемента, который нужно удалить из массива. Кроме того, определите метод класса (merge), позволяющий объединять два массива (второй добавляется в конец первого).

8. Вариант

Создайте **класс – динамический массив**. В качестве элементов массива выберите **вещественные числа**.

Определите в нем конструктор (можно несколько), деструктор, конструктор копирования. Перегрузите операцию []. Определите методы класса, которые позволяли бы добавлять и удалять элемент из массива (insert, erase). Функция insert – добавляет элемент в конец массива. Функция erase – принимает значение элемента, который нужно удалить из массива, если таких элементов больше чем 1, нужно удалить их все.

9. Вариант

Создайте **класс – динамический массив**. В качестве элементов массива выберите **вещественные числа**.

Определите в нем конструктор (можно несколько), деструктор, конструктор копирования. Перегрузите операцию []. Определите методы класса, которые позволяли бы добавлять и удалять элемент из массива (insert, erase). Функция insert – принимает индекс элемента, после которого нужно вставить новый. Функция erase – принимает индекс элемента, который нужно удалить из массива. Кроме того, определите метод класса – reverse (переставляет элементы в массиве в обратном порядке).

10. Вариант

Создайте **класс – динамический массив**. В качестве элементов массива выберите **вещественные числа**.

Определите в нем конструктор (можно несколько), деструктор, конструктор копирования. Перегрузите операцию []. Определите методы класса, которые позволяли бы добавлять и удалять элементы из массива (insert, erase). Функция insert – добавляет элемент в конец массива. Функция erase – принимает два индекса i и j, между которыми нужно удалить элементы.

11. Вариант

Создайте **класс – динамический массив**. В качестве элементов массива выберите **целые числа**.

Определите в нем конструктор (можно несколько), деструктор, конструктор копирования. Перегрузите операцию []. Определите методы класса, которые позволяли бы добавлять и удалять элемент из массива (insert, erase). Функция insert – добавляет элемент в конец массива. Функция erase – принимает индекс элемента, который нужно удалить из массива. Кроме того, определите метод класса – sort (сортирует массив, реализуйте возможность выбора - сортировать по убыванию или по возрастанию).

12. Вариант

Создайте **класс – динамический массив**. В качестве элементов массива выберите **целые числа**.

Определите в нем конструктор (можно несколько), деструктор, конструктор копирования. Перегрузите операцию []. Определите методы класса, которые позволяли бы добавлять и удалять элемент из массива (insert, erase). Функция insert – добавляет элемент в начало массива. Функция erase – принимает индекс элемента, который нужно удалить из массива. Кроме того, определите метод класса (merge), позволяющий объединять два массива (второй добавляется в конец первого).

13. Вариант

Создайте **класс – динамический массив**. В качестве элементов массива выберите **целые числа**.

Определите в нем конструктор (можно несколько), деструктор, конструктор копирования. Перегрузите операцию []. Определите методы класса, которые позволяли бы добавлять и удалять элемент из массива (insert, erase). Функция insert – добавляет элемент в конец массива. Функция erase – принимает значение элемента, который нужно удалить из массива, если таких элементов больше чем 1, нужно удалить их все.

14. Вариант

Создайте **класс – динамический массив**. В качестве элементов массива выберите **целые числа**.

Определите в нем конструктор (можно несколько), деструктор, конструктор копирования. Перегрузите операцию []. Определите методы класса, которые позволяли бы добавлять и удалять элемент из массива (insert, erase). Функция insert – принимает индекс элемента, после которого нужно вставить новый. Функция erase – принимает индекс элемента, который нужно удалить из массива. Кроме того, определите метод класса – reverse (переставляет элементы в массиве в обратном порядке).

15. Вариант

Создайте **класс – динамический массив**. В качестве элементов массива выберите **целые числа**.

Определите в нем конструктор (можно несколько), деструктор, конструктор копирования. Перегрузите операцию []. Определите методы класса, которые позволяли бы добавлять и удалять элементы из массива (insert, erase). Функция insert – добавляет элемент после заданного по значению элемента (т.е. например добавить число 5 после числа 3 в массиве). Функция erase – принимает индекс элемента, с которого нужно начать удаление и количество элементов, которых нужно удалить из массива.

16. Вариант

Создайте **класс - динамический массив**. В качестве элементов массива выберите **символы**.

Определите в нем конструктор (можно несколько), деструктор, конструктор копирования. Перегрузите операцию []. Определите методы класса, которые позволяли бы добавлять и удалять элемент из массива (insert, erase). Функция insert – добавляет элемент в конец массива. Функция erase – принимает индекс элемента, который нужно удалить из массива. Кроме того, определите метод класса – sort (сортирует массив, реализуйте возможность выбора - сортировать по убыванию или по возрастанию).

17. Вариант

Создайте **класс - динамический массив**. В качестве элементов массива выберите **символы**.

Определите в нем конструктор (можно несколько), деструктор, конструктор копирования. Перегрузите операцию []. Определите методы класса, которые позволяли бы добавлять и удалять элемент из массива (insert, erase). Функция insert – добавляет элемент в начало массива. Функция erase – принимает индекс элемента, который нужно удалить из массива. Кроме того, определите метод класса (merge), позволяющий объединять два массива (второй добавляется в конец первого).

18. Вариант

Создайте **класс - динамический массив**. В качестве элементов массива выберите **символы**.

Определите в нем конструктор (можно несколько), деструктор, конструктор копирования. Перегрузите операцию []. Определите методы класса, которые позволяли бы добавлять и удалять элемент из массива (insert, erase). Функция insert – добавляет элемент в конец массива. Функция erase – принимает значение элемента, который нужно удалить из массива, если таких элементов больше чем 1, нужно удалить их все.

19. Вариант

Создайте **класс - динамический массив.** В качестве элементов массива выберите **символы.**

Определите в нем конструктор (можно несколько), деструктор, конструктор копирования. Перегрузите операцию []. Определите методы класса, которые позволяли бы добавлять и удалять элемент из массива (insert, erase). Функция insert – принимает индекс элемента, после которого нужно вставить новый. Функция erase – принимает индекс элемента, который нужно удалить из массива. Кроме того, определите метод класса – reverse (переставляет элементы в массиве в обратном порядке).

20. Вариант

Создайте **класс - динамический массив.** В качестве элементов массива выберите **символы.**

Определите в нем конструктор (можно несколько), деструктор, конструктор копирования. Перегрузите операцию []. Определите методы класса, которые позволяли бы добавлять и удалять элементы из массива (insert, erase). Функция insert – добавляет элемент в конец массива. Функция erase – принимает индекс элемента, с которого нужно начать удаление и количество элементов, которых нужно удалить из массива.

21. Вариант

Создайте **класс - динамический массив.** В качестве элементов массива выберите **символы.**

Определите в нем конструктор (можно несколько), деструктор, конструктор копирования. Перегрузите операцию []. Определите методы класса, которые позволяли бы добавлять и удалять элемент из массива (insert, erase). Функция insert – добавляет элемент в конец массива. Функция erase – принимает индекс элемента, который нужно удалить из массива. Кроме того, определите метод класса, позволяющий складывать два массива.

22. Вариант

Создайте класс - динамический массив. В качестве элементов массива выберите символы.

Определите в нем конструктор (можно несколько), деструктор, конструктор копирования. Перегрузите операцию []. Определите методы класса, которые позволяли бы добавлять и удалять элемент из массива (insert, erase). Функция insert – добавляет элемент в начало массива. Функция erase – принимает индекс элемента, который нужно удалить из массива. Кроме того, определите метод класса – find, ищущий подстроку в массиве символов (принимает char *) и возвращающий указатель на начало этой подстроки в массиве, если подстрока не найдена, возвращает NULL.

23. Вариант

Создайте **класс - динамический массив.** В качестве элементов массива выберите **целые числа.**

Определите в нем конструктор (можно несколько), деструктор, конструктор копирования. Перегрузите операцию []. Определите методы класса, которые позволяли бы добавлять и удалять элемент из массива (insert, erase). Функция insert – добавляет элемент в конец массива. Функция erase – принимает индекс элемента, который нужно удалить из массива. Кроме того, определите метод класса – find, ищущий элемент в массиве по его значению и возвращающий индекс этого элемента в массиве.

24. Вариант

Создайте **класс - динамический массив.** В качестве элементов массива выберите **целые числа.**

Определите в нем конструктор (можно несколько), деструктор, конструктор копирования. Перегрузите операцию []. Определите методы класса, которые позволяли бы добавлять и удалять элемент из массива (insert, erase). Функция insert – принимает индекс элемента, после которого нужно вставить новый. Функция erase – принимает индекс элемента, который нужно удалить из массива. Кроме того, определите метод класса, позволяющий складывать два массива.

25. Вариант

Создайте **класс - динамический массив**. В качестве элементов массива выберите **целые числа**.

Определите в нем конструктор (можно несколько), деструктор, конструктор копирования. Перегрузите операцию []. Определите методы класса, которые позволяли бы добавлять и удалять элемент из массива (insert, erase). Функция insert – добавляет элемент в конец массива. Функция erase – принимает индекс элемента, который нужно удалить из массива. Кроме того, определите метод класса – swap (меняет местами 2 элемента в массиве).