1. Оператор continueиспользуется только в циклах. В операторах for, while, do while**,**оператор continueвыполняет пропуск оставшейся части кода тела цикла и переходит к следующей итерации цикла. Рассмотрим фрагмент кода с оператором continue.

Оператор break завершает выполнение ближайшего внешнего цикла или условного оператора, в котором он находится. Управление передается оператору, который расположен после оператора, при его наличии.

#include <iostream>

using namespace std;

int main()

{

// An example of a standard for loop

for (int i = 1; i < 10; i++)

{

cout << i << '\n';

if (i == 4)

break;

}

Постусловие

int i = 0;

do {

    printf("%d\n", i);

    i++;

} while(i < 10);

2. **Функция** — это определенная группа операций с уникальным именем, которая может:

1. Вызываться по имени в любом месте программы.
2. Получать определенный набор значений из внешней программы в момент вызова.
3. Возвращать в качестве значения некоторый результат заранее заданного типа.

Тип\_результата Имя\_функции (Тип\_пар1, Тип\_пар2, ...);

В частности, внутри своего тела функция может вызывать саму себя. Такое явление называется рекурсией.

**3.Массив** – это совокупность данных, которая обладает следующими свойствами:

* все элементы массива имеют один и тот же тип;
* массив имеет одно имя для всех элементов;
* доступ к конкретному элементу массива осуществляется по индексу (индексам).

**Индексация элементов массива в языке C++ начинается с нуля**int a[3] = {0, 1, 2};

Int d[2][3] = {{0, 1, 2},   
 {3, 4, 5}};

**Динамическим** называется [массив](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%81%D0%B8%D0%B2_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), размер которого, при необходимости, может меняться во время исполнения программы.

Для доступа к конкретному элементу массива используются так называемые индексные выражения

Указатель – переменная, значением которой является адрес ячейки памяти. То есть указатель ссылается на блок данных  из области памяти, причём на самое его начало. Указатель может ссылаться на переменную или функцию.

1. Строки в С++ представляются как массивы элементов типа char, заканчивающиеся нуль-терминатором \0  
    Символьные строки состоят из набора символьных констант заключённых в двойные кавычки

Символ – элементарная единица, некоторый набор которых несет определенный смысл.

//инициализация строки без указания размера

char string[] = "abcdefghf";

char string[10] = "abcdefghf";

|  |  |
| --- | --- |
| [**strcpy**](http://cppstudio.com/spravochnik/standartnye-zagolovochnye-fajly-iz-si-v-s/zagolovochnyj-fajl-cstring-string-h/funkciya-strcpy/)**(**s1,s2**)** | выполняет побайтное копирование символов из строки  s2 в строку s1 |
| [**strcat(s1,s2)**](http://cppstudio.com/spravochnik/standartnye-zagolovochnye-fajly-iz-si-v-s/zagolovochnyj-fajl-cstring-string-h/funkciya-strcat/) | объединяет строку s2 со строкой s1. Результат сохраняется в s1 |
| [**strchr(s,c)**](http://cppstudio.com/spravochnik/standartnye-zagolovochnye-fajly-iz-si-v-s/zagolovochnyj-fajl-cstring-string-h/funkciya-strchr/) | поиск первого вхождения символа с в строке s. В случае удачного поиска возвращает указатель на место первого вхождения символа с. Если символ не найден, то возвращается ноль. |

**5.Классы в С++** — это абстракция описывающая методы, свойства, ещё не существующих объектов.

// объявление классов в С++

class /\*имя класса\*/

{

  private:

  /\* список свойств и методов для использования внутри класса \*/

  public:

  /\* список методов доступных другим функциям и объектам программы \*/

  protected:

  /\*список средств, доступных при наследовании\*/

};

Классы имеют свои функции, которые называются методами класса.

**Конструктор** — специальная функция, которая выполняет начальную инициализацию элементов данных, причём имя конструктора обязательно должно совпадать с именем класса. Важным отличием конструктора от остальных функций является то, что он не возвращает значений вообще никаких, в том числе и void

6. **Инкапсуляция** — это свойство, позволяющее объединить в классе и данные, и методы, работающие с ними и скрыть детали реализации от пользователя.

**Наследование** — это свойство, позволяющее создать новый класс-потомок на основе уже существующего, при этом все характеристики класса родителя присваиваются классу-потомку.

**Полиморфизм** — свойство классов, позволяющее использовать объекты классов с одинаковым интерфейсом без информации о типе и внутренней структуре объекта.