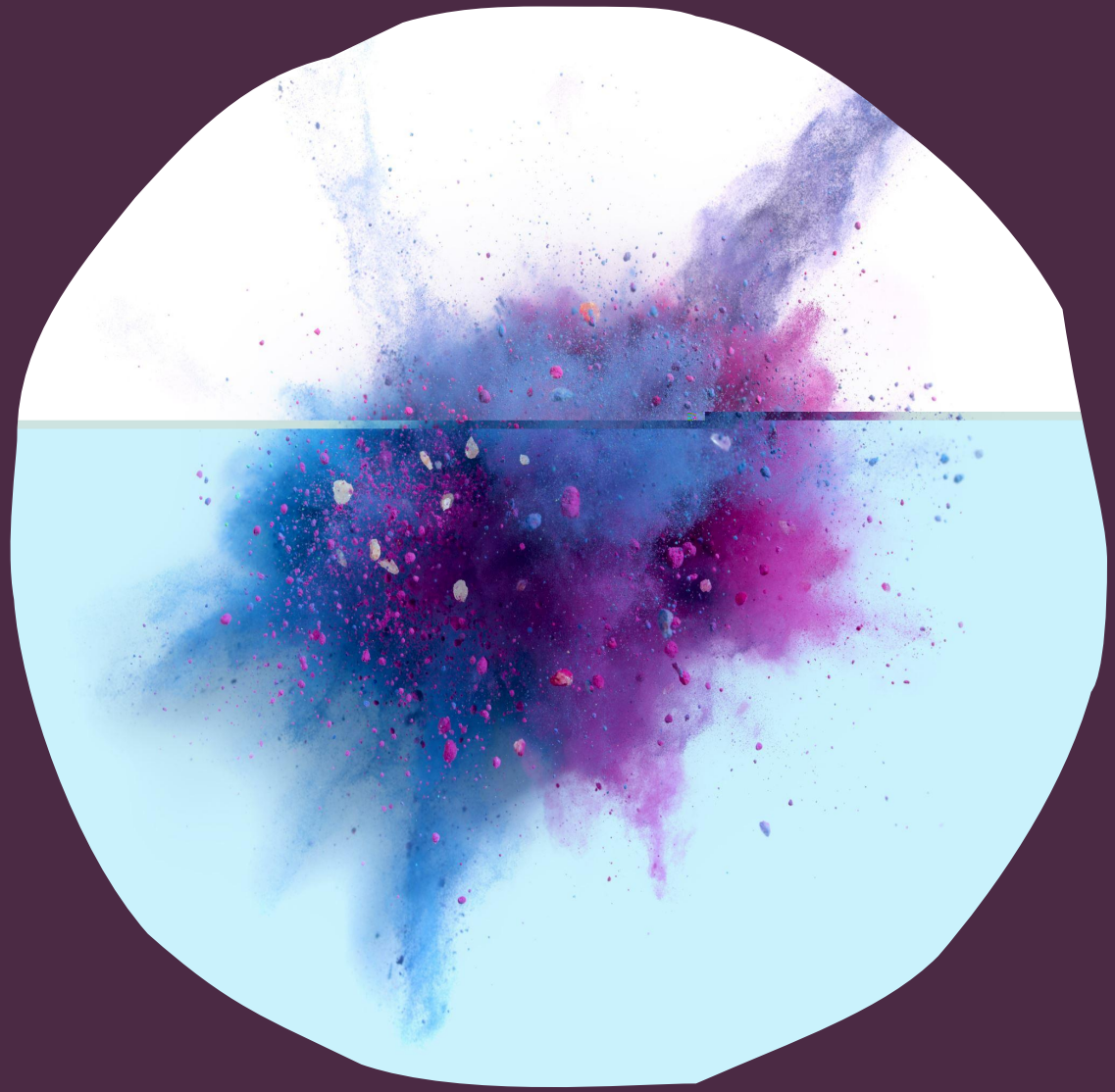
# RAVESLI.COM
# METANIT.COM

Подготовил: Осиповский Д. С.



# Написать функцию сортировки массива интов

Важно, очень простая задачка вверху!

```
void sortArr(int* arr, int size_arr) {
  for (int i{}; i < size_arr - 1; ++i) {
    for (int j{i + 1}; j < size_arr; ++j) {
      if (arr[i] > arr[j]) {
        std::swap(arr[i], arr[j]);
      }
    }
  }
}
```

# Написать функцию сортировки массива даблов

```
void sortArr(double* arr, int size_arr) {
  for (int i{}; i < size_arr - 1; ++i) {
    for (int j{i + 1}; j < size_arr; ++j) {
      if (arr[i] > arr[j]) {
        std::swap(arr[i], arr[j]);
      }
    }
  }
}
```

# Написать функцию сортировки массива char - ов

```
void sortArr(char* arr, int size_arr) {
  for (int i{}; i < size_arr - 1; ++i) {
    for (int j{i + 1}; j < size_arr; ++j) {
      if (arr[i] > arr[j]) {
        std::swap(arr[i], arr[j]);
      }
    }
  }
}
```

Написать функцию сортировки массива bool – ов (ну, а что? а вдруг?!)

```
void sortArr(bool* arr, int size_arr) {
  for (int i{}; i < size_arr - 1; ++i) {
    for (int j{i + 1}; j < size_arr; ++j) {
      if (arr[i] > arr[j]) {
        std::swap(arr[i], arr[j]);
      }
    }
  }
}
```

Псс, люди, ничего не заметили? Давайте ещё раз, найдите 5 отличий:

```
void sortArr(bool* arr, int size_arr) {
  for (int i{}; i < size_arr - 1; ++i) {
    for (int j{i + 1}; j < size_arr; ++j) {
      if (arr[i] > arr[j]) {
        std::swap(arr[i], arr[j]);
      }
    }
  }
}
```

```
void sortArr(int* arr, int size_arr) {
  for (int i{}; i < size_arr - 1; ++i) {
    for (int j{i + 1}; j < size_arr; ++j) {
      if (arr[i] > arr[j]) {
        std::swap(arr[i], arr[j]);
      }
    }
  }
}
```

```
void sortArr(double* arr, int size_arr) {
  for (int i{}; i < size_arr - 1; ++i) {
    for (int j{i + 1}; j < size_arr; ++j) {
      if (arr[i] > arr[j]) {
        std::swap(arr[i], arr[j]);
      }
    }
  }
}
```

```
void sortArr(char* arr, int size_arr) {
  for (int i{}; i < size_arr - 1; ++i) {
    for (int j{i + 1}; j < size_arr; ++j) {
      if (arr[i] > arr[j]) {
        std::swap(arr[i], arr[j]);
      }
    }
  }
}
```

Везде меняется только тип! Не слишком ли жирно, писать кучу кода РУКАМИ, когда в нём меняется только одно слово???! – СЛИШКОМ!

Именно для этого, придумали такую штуку, как шаблон кода! А ещё, такую замечательную парадигму, как – **ОБОБЩЁННОЕ ПРОГРАММИРОВАНИЕ!**

# Обобщённое программирование -

- Принцип написания кода согласно которому, следует писать такие алгоритмы, которые могут одинаково работать с различными типами данных.

# Шаблон кода

Написание куска кода, пренебрегая типами данных, которыми он управляет.

# Шаблон кода (более формально – беее, много букв)

Это обобщенное описание поведения функций, которые могут вызываться для объектов разных типов. Другими словами, шаблон функции (шаблонная функция, обобщённая функция) представляет собой семейство разных функций (или описание алгоритма). По описанию шаблон функции похож на обычную функцию: разница в том, что некоторые элементы не определены (типы, константы) и являются параметризованными.

# Хватит теории, давайте практику!

# BO!

```
template<>
```

# Превратим кучу функций в одну:

```
template<typename T>
void sortArr(T* arr, int size_arr) {
   for (int i{}; i < size_arr - 1; ++i) {
      for (int j{i + 1}; j < size_arr; ++j) {
         if (arr[i] > arr[j]) {
            std::swap(arr[i], arr[j]);
         }
      }
   }
}
```

# Рассмотрим, что ещё можно делать:
# Типов может быть больше одного

```
template<typename T, typename A, typename B, typename K>
void testFunc(T* arr_1, A* arr_2, B* arr_3, K* arr_4) {

}
```

# Или вообще бесконечное кол – во:

```
void myPrint() {
    cout << endl;
}

template<typename firstType, typename ...moreTypes>
void myPrint(const firstType &value, const moreTypes &...other) {
    cout << value << endl;
    myPrint(other...);
}
```

```
int main() {
    myPrint(
            "This is string",
            'C',
            12,
            48.5,
            "WOW!"
    );

    return 0;
}
```

# Вызов шаблонных функций:

```
template<typename T>
T sum(T a, T b) {
   return a + b;
}
```

```
int main() {

   // либо так
   cout << sum(1, 2) << endl;

   // либо так
   cout << sum<int>(1, 2) << endl;

   return 0;
}
```

# Параметры, которые не являются типами:

```
template<typename T, int S>
T getSumArr(T (&arg)[S]) {
    T sum{};
    for (const T &it : arg) {
        sum += it;
    }
    return sum;
}
```

# Вызов, вообще магия )

```
int arr[5]{1, 2, 3, 4, 5};

cout << getSumArr(arr) << endl;
```

# Шаблон шаблона, шаблона, шаблона, шаб…, ну вы поняли

```
template<typename T, template<typename, int> typename Arr>
void testFunc() {
   T var;
   Arr<T, 10> a;
};
```

# Аргументы шаблона по умолчанию

```
template <class T, class Allocator = allocator<T>>
class vector;
```

# Специализация шаблонов

```
template<typename T>
    void sortArr(T* arr, int size_arr) {
        for (int i{}; i < size_arr - 1; ++i) {
            for (int j{i + 1}; j < size_arr; ++j) {
                if (arr[i] > arr[j]) {
                    std::swap(arr[i], arr[j]);
                }
            }
    }
}
```

```
template<>
void sortArr<bool>(bool* arr, int size_arr) {
    cout << "Кх, ну ты это, ну, не шути так, не надо" << endl;
}
```

# Экземпляры шаблонов

```
template<typename T>
T sum(T a, T b) {
  return a + b;
}

template int sum(int, int);
```

# Инстанциирование шаблонов

Всё на столько просто, что писать нечего :)

# Перегрузка шаблонов

```
template<class T>
T sqrt(T);

template<class T>
complex<T> sqrt(complex<T>);
double sqrt(double);

void fun(complex<double> z) {
  sqrt(2);
  sqrt(2.0);
  sqrt(z);
}
```

# Шаблоны классов

Всё стандартно, точно также, как для функций

# RAVESLI.COM
# -
# METANIT.COM