Государственное бюджетное общеобразовательное учреждение

Школа №690 Невского района Санкт-Петербурга

ИТОГОВЫЙ ПРОЕКТ

на тему «Язык программирования **С++**»

Выполнила: ученица 10 «Э» класса

Алиева Айдан Мобил кызы

 Руководитель проекта:

 Дмитриев Максим Сергеевич,

 учитель информатики

г. Санкт-Петербург

2023 год

**Содержание:**

[Введение **3**](#_Toc127474584)

**Глава 1 “История С++”…………………..………………….…………...……..4**

**1.1** “Преимущество С++”...….………………..……………………………...…...4

**1.2** “**Недостатки C++**”.………………..……………………………………..……5

**1.3** “**Особенности языка программирования C++**”.………………...……..……5

[**Глава 2 “Основы языка С++ ”……………………………..…………………..**](#_Toc127474585)**6**

[Г**лава** **3 “Классы и объекты”.**…….………………………….……………….](#_Toc127474585)...7

**3.1** “**Классы**”.………………..………………………………………………..……7

**3.2** “**Объекты**”.…………...…………………………………………………..……8

[Г**лава** **4 “Наследование и перезагрузка операторов”……………..………...**](#_Toc127474585)8

**4.1** “**Наследование**”.………………..………………………………………..……8

**4.2** “**Перезагрузка операторов** ”.…………...…...…………………………..……9

[Г**лава** **5 “Исключения”……………..………………………………….…...**](#_Toc127474585)**.....10**

[Г**лава** **6 “Шаблоны”……………..……….…………………….………….…...**](#_Toc127474585)**12**

**6.1** “template”...…………………..……………………………...………………..12

**6.2** “**print**”.………………..…………….………………………………………....12

**6.3** “Обобщенные классы”………………………………………………………13

**6.4** “**Stack**”.………………..………………………………………………..…..…14

[Г**лава** **7 “Средства ввода – вывода в С++”……………..…………………...**](#_Toc127474585)**14**

**7.1** “Системные вызовы”...……………...……………………...………………..12

**7.2** “**Файловый поток**”.………………..………………………………………....12

**7.3** “Системные вызовы”…………………...……………………………………13

[Г**лава** **8 “Применение С++”……………..…………….…...**](#_Toc127474585)**.............................17**

[Заключение…………...…………..………………………………………………1**8**](#_Toc127474586)

#

# **Введение**

 Язык программирования С++ является одним из наиболее популярных и широко используемых языков программирования в мире. Он представляет собой продолжение языка С и добавляет в него новые возможности, такие как объектно-ориентированное программирование и шаблоны.

 Язык программирования С++ был разработан в 1979 году Бьёрном Струструпом как расширение языка С. Сейчас С++ является одним из самых популярных языков программирования в мире. Имеет мощные средства работы с памятью, такие как указатели и ссылки, и широко применяется во многих областях. Благодаря своей эффективности и мощности, С++ стал неотъемлемой частью многих сфер деятельности, от игровой индустрии до финансовой и научной сфер.

**“Язык программирования С++”**

**Проблема –** одной из основных проблем при изучении языка С++ является его сложность и множество особенностей. Это может вызывать затруднения у начинающих программистов и препятствовать полному овладению языком.

**Цель –** изучение основ языка программирования С++, его возможностей и применения в различных отраслях.

**Гипотеза** – я предполагаю, что с усовершенствованием старых и созданием новых языков программирования людям стало легче работать, а программирование стало не таким сложным.

**Глава 1 “История С++”**

Язык С++ является объектно-ориентированным языком программирования и представляет собой комбинацию как низкоуровневого, так и высокоуровневого языка – языка среднего уровня. Язык программирования был создан, спроектирован и разработан датским ученым–компьютерщиком Бьярне Страуструпом в Bell Telephone Laboratories (ныне известной как Nokia Bell Labs) в Мюррей-Хилл, штат Нью-Джерси. Поскольку он хотел гибкий и динамичный язык, который был бы похож на C со всеми его функциями, но с дополнительной возможностью активной проверки типов, базовым наследованием, аргументом функционирования по умолчанию, классами, встраиванием и т. Д. Следовательно, был запущен C с классами (C ++).

**C++ изначально был известен как “C с классами”, а в 1983 году был переименован в C++. ++ — это сокращение для добавления единицы к разнообразию в программировании; поэтому C ++ примерно означает, что “единица выше, чем C.”**

Таким образом, C++ задумывался как язык Cи с расширенными возможностями. Бoльшaя часть языка Cи вошла в C++ как подмножество, поэтому многие программы Cи могут быть скомпилированы (т. е. превращены в набор низкoуровневых инструкций, которые компьютер может выполнять напрямую) с помощью компилятора C++.

* 1. **“Преимущество С++.”**

Одним из основных преимуществ языка C++ является его возможность для разработки настольных приложений, используемых на компьютерах и серверах, а также его поддержка промышленных стандартов, таких как ISO и ANSI. C++ является многоплатформенным языком программирования и может работать на различных операционных системах, включая Windows и Unix.

Другим преимуществом языка C++ является его высокая скорость и эффективность в расходе ресурсов процессора, что делает его идеальным выбором для разработки высоконагруженных приложений, таких как компиляторы, базы данных и игры.

* 1. “**Недостатки C++**”.

У языка C++ также есть некоторые недостатки. Например, C++ может быть сложным для начинающих программистов, поскольку он требует широкого знания парадигмы объектно-ориентированного программирования и низкоуровневых концепций программирования, таких как указатели. Кроме того, C++ может быть менее безопасным, чем другие языки, такие как Java и Python, поскольку он не имеет встроенных механизмов для обработки ошибок и защиты от переполнения буфера.

Несмотря на некоторые его недостатки язык C++ до сих пор широко используется для разработки высокоэффективных и масштабируемых приложений, и продолжает оставаться одним из наиболее важных языков программирования в мире.

* 1. “**Особенности языка программирования C++**”.

**Пять лучших особенностей C ++: -**

1. Насколько известно, это наиболее часто используемый язык в конкурентном программировании. Большинство программистов с самым высоким рейтингом обычно используют C ++ для криптографии. Вы увидите это на любых онлайн-сайтах.
2. STL (стандартная библиотека руководств): Это чрезвычайно экономит время для людей, которые знают C и все еще пишут код для пузырькового типа.
3. Перегрузка оператора
4. Множественное наследование. Объемные языки не имеют такой возможности.
5. В конце концов, способность модулировать код, инкапсуляция и полиморфизм.

# **Глава 2 “Основы языка программирования”.**

Основные концепции, которые нужно знать при изучении языка С++, можно разделить на несколько основных категорий.

**1. “ Переменные и типы данных”.**

Переменные используются для хранения данных в памяти компьютера. Язык С++ поддерживает множество типов данных, включая целочисленные, числа с плавающей точкой, символы и строки. Для объявления переменной используется ключевое слово "int", "float", "char" или "string", за которым следует имя переменной.

**2. “ Операторы и выражения”.**

Операторы используются для выполнения операций над переменными в языке С++. Они могут быть арифметическими (+, -, \*, /, %), логическими (&&, ||,!) или сравнения (==, !=, <, >, <=, >=). Выражения создаются путем сочетания переменных и операторов.

**3. “Условные операторы”.**

Условные операторы используются для выполнения определенного блока кода в зависимости от определенного условия. В языке С++ есть два основных условных оператора: if и switch.

**4. “Циклы”.**

Циклы используются для повторения определенных действий определенное количество раз или пока выполняется определенное условие. В языке С++ есть три основных типа циклов: for, while и do-while.

**5. “Функции”.**

Функции используются для группирования определенного блока кода, который может быть вызван из других частей программы. Функции могут иметь параметры и возвращать значения.

**6. “ООП”. (**объектно-ориентированное программирование)

Язык С++ является объектно-ориентированным языком программирования, и он поддерживает концепции ООП, такие как классы, объекты, наследование, полиморфизм и инкапсуляцию.

Эти основные концепции С++ обеспечивают программистам гибкость и мощь в разработке программ, а также позволяют им создавать более эффективный и чистый код.

**Глава 3 “Классы и объекты”.**

Классы и объекты — это основные концепции объектно-ориентированного программирования, которые широко используются в С++.

В С++ классы и объекты широко используются в различных областях программирования, таких как разработка игр, веб-приложений, системного программирования и многих других. Они предоставляют мощный инструментарий для создания эффективных и масштабируемых программ.

**3.1 “ Классы”.**

**Класс** — это пользовательский тип данных, который определяет набор свойств и методов, которые могут быть использованы для создания объектов. Классы могут быть рассмотрены как шаблоны для создания объектов, а объекты - как экземпляры классов.

Описание класса включает в себя определение его свойств (переменные) и методов (функции), которые могут быть использованы для работы с этими свойствами. Свойства класса могут быть публичными, приватными или защищенными. Публичные свойства доступны для всех членов программы, приватные свойства доступны только внутри класса, а защищенные свойства доступны только внутри класса и его наследников.

Методы класса могут быть публичными или приватными. Публичные методы могут быть вызваны из любого места программы, а приватные методы доступны только внутри класса.

**3.1 “ Объекты”.**

**Объект** — это экземпляр класса. Он содержит конкретные значения свойств класса и может использоваться для вызова методов класса. Объекты могут быть созданы в любой части программы, где определен класс.

Для создания объекта необходимо использовать оператор new и имя класса. Например, MyClass\* myObject = new MyClass (); создаст объект типа MyClass и присвоит его указатель переменной myObject.

Одним из основных преимуществ объектно-ориентированного программирования является возможность создания модульных и масштабируемых программ. Классы и объекты позволяют разделять код на логические блоки, что упрощает его понимание и редактирование. Кроме того, объекты могут быть наследованы от других классов, что позволяет создавать более сложные объекты на основе уже существующих.

# **Глава 4 “ Наследование и перезагрузка операторов”.**

Наследование и перегрузка операторов являются уникальными особенностями языка C++, позволяющими эффективно работать с объектно-ориентированным кодом и определять новые типы данных с необходимой функциональностью.

**4.1 “ Наследование”.**

**Наследование** — это механизм объектно-ориентированного программирования, который позволяет создавать новый класс на основе уже существующего. Новый класс называется производным классом, а существующий - базовым классом. Производный класс наследует все свойства и методы базового класса и может добавлять к ним свои собственные свойства и методы.

Для создания производного класса необходимо использовать ключевое слово "class" или "struct", после которого указывается имя производного класса и имя базового класса в скобках.

**Пример наследования:**

class Animal {

public:

  int age;

  virtual void sound () {}  // Виртуальная функция

};

class Cat : public Animal {  // Определение класса-подкласса

public:

  void sound() override {

    std:: cout << "Meow!" << std::endl;  // Переопределение функции базового класса

  }

};

int main () {

  Cat myCat;

  myCat.age = 2;  // Наследование свойства базового класса

  myCat.sound();  // Вызов новой функции подкласса

}

**4.2 “Перезагрузка оператора”.**

**Перегрузка операторов** — это возможность определить новое поведение для операторов, таких как +, -, \*, / и др. Как и в случае с наследованием, перегрузка операторов позволяет дополнить уже существующую функциональность класса или изменить ее в соответствии с требованиями разработчика. Например, можно определить, как должен работать оператор "+" для объектов пользовательского класса.

**Пример перегрузки оператора** +**:**

using namespace std;

class Vector {

public:

  double x, y;

  Vector operator+ (const Vector& other) const {  // Оператор сложения векторов

    double newX = x + other.x;

    double newY = y + other.y;

    return Vector{newX, newY};

  }

};

int main () {

  Vector a {1, 2};

  Vector b {3, 4};

  Vector c = a + b;  // Использование оператора сложения векторов

  cout << c.x << " " << c.y << endl;  // Вывод результата: 4 6

}

# **Глава 5 “Исключения”.**

**Исключения в языке программирования C++** — это механизм обработки ошибок, который позволяет программисту контролировать и обрабатывать исключительные ситуации в своей программе. Исключение может быть выброшено в любом месте программы, где происходит ошибка, и передано в блок try-catch для дальнейшей обработки.

Исключение представляет собой объект, который содержит информацию об ошибке, такую как тип ошибки и сообщение об ошибке. Когда исключение выбрасывается, программа переходит к ближайшему блоку try-catch, который может обработать это исключение. Если нет соответствующего блока catch, то программа завершается с ошибкой.

Блок try-catch состоит из двух частей: блока try и одного или нескольких блоков catch. Блок try содержит код, который может выбросить исключение. Блок catch содержит код, который будет выполнен, если соответствующее исключение было выброшено в блоке try. Блок catch может содержать несколько исключений, каждое из которых соответствует определенному типу ошибки.

**Пример использования исключений:**

using namespace std;

try {

    // код, который может выбросить исключение

    if (x == 0) {

        throw std::runtime\_error("x не может быть равен 0");

    }

} catch (const std::exception& e) {

    // код, который будет выполнен при выбросе исключения

    std::cout << "Ошибка: " << e.what() << std::endl;

}

В этом примере, если значение переменной x равно 0, то будет выброшено исключение типа std::runtime\_error с сообщением об ошибке. Блок catch содержит тип исключения const std::exception&, что означает, что он будет перехватывать все исключения, производные от класса std::exception. В блоке catch выводится сообщение об ошибке, которое было передано при выбросе исключения.

Использование исключений в языке C++ позволяет более точно контролировать ошибки в программе и предоставляет возможность обработки ошибок в месте, где это наиболее удобно и эффективно. Однако, следует использовать исключения с умом, чтобы избежать излишней нагрузки на производительность программы.

# **Глава 6 “Шаблоны”.**

**Шаблоны в С++** — это механизм, который позволяет создавать обобщенные функции и классы. Они представляют собой шаблонный код, который может быть использован для генерации конкретного кода при компиляции программы.

**6.1 “template”.**

Использование шаблонов позволяет уменьшить количество дублирующегося кода и повторно использовать одинаковый функционал для разных типов данных. Например, можно написать обобщенную функцию сортировки массива любого типа данных (int, double или пользовательский класс), используя шаблоны. Шаблон определяется с помощью ключевого слова **template**, за которым следует список параметров шаблона, разделенных запятыми, и тело шаблона, в котором могут быть определены обобщенные классы или функции.

**Пример определения шаблона функции:**

template <typename T>

void print (T value) {

    std::cout << value << std::endl;

}

**6.2 “print”.**

В этом примере шаблон функции **print** определен с помощью ключевого слова template и параметра шаблона T. Функция принимает один аргумент типа T и выводит его на экран. Тип T может быть любым типом данных, таким как целочисленный, вещественный или пользовательский тип.

**Пример использования шаблона функции:**

print (10); // выводит 10

print (3.14); // выводит 3.14

print("Hello"); // выводит "Hello"

В этом примере шаблон функции print используется для вывода различных типов данных на экран.

**6.3 “Обобщенные классы”.**

Шаблоны также могут использоваться для создания обобщенных классов, которые могут работать с различными типами данных. Обобщенные классы могут содержать обобщенные методы и переменные.

**Пример определения шаблона класса:**

template<typename T>

class Stack {

private:

    T\* data;

    int size;

public:

    Stack() {

        data = new T[100];

        size = 0;

    }

    void push(T value) {

        data[size++] = value;

    }

    T pop() {

        return data[--size];

    }

};

**6.4 “Stack”.**

В этом примере шаблон класса Stack используется для создания стека, который может содержать элементы различных типов данных. Создаются два объекта класса Stack - s1 для хранения целых чисел и s2 для хранения строк.

**Пример использования шаблона класса:**

Stack<int> s1;

s1.push(10);

s1.push(20);

std::cout << s1.pop() << std::endl; // выводит 20

Stack<std::string> s2;

s2.push("Hello");

s2.push("World");

std::cout << s2.pop() << std::endl; // выводит "World"

В целом использование шаблонов в C++ помогает достичь большей гибкости программирования за счет возможности создания универсальных функций и классов, которые могут работать с различными типами данных.

# **Глава 7 “Средства ввода - вывода в С++”.**

Средства ввода-вывода в языке программирования C++ позволяют осуществлять операции ввода и вывода данных из и в различные источники и назначения. Они включают в себя стандартные потоки ввода-вывода (stdin, stdout, stderr), файловые потоки, строковые потоки, а также потоки для работы с сетью.

В C++ существует несколько способов ввода-вывода данных:

**7.1 “Стандартный поток”.**

**Стандартный поток ввода-вывода (iostream)** — это наиболее распространенный и удобный способ работы с консольным вводом-выводом. Для этого используются объекты cin и cout, которые позволяют читать данные из стандартного потока ввода (клавиатура) и выводить данные на стандартный поток вывода (экран).

**Пример:**

```

#include <iostream>

using namespace std;

int main() {

   int x;

   cout << "Введите число: ";

   cin >> x;

   cout << "Вы ввели число: " << x;

}

```

**7.2 “Файловый поток”.**

 **Файловый поток** — это способ чтения/записи данных из файлов на жестком диске или других накопителях.

Для работы с файлами нужно использовать классы ifstream для чтения файла и ofstream для записи файла.

**Пример чтения файла:**

```

#include <fstream>

using namespace std;

int main() {

    ifstream file("example.txt");

    string line;

    while(getline(file, line)) { //читаем построчно

        cout<<line<<endl; // выводим строку

    }

}

```

**7.3 “Системные вызовы”.**

 **Системные вызовы** — это более низкоуровневый подход к работе с данными, который позволяет обращаться к системным функциям операционной системы напрямую.

Например, функция read() позволяет читать данные из файлового дескриптора, а write() - записывать данные в файл.

**Пример чтения из файла с помощью системного вызова:**

```

#include <unistd.h>

#include <fcntl.h>

int main(){

    int fd = open("example.txt", O\_RDONLY); //открытие файла на чтение

    char buffer[1024];

    ssize\_t nread;

    while((nread = read(fd, buffer, sizeof(buffer))) > 0) { //читаем по блокам

        if(write(STDOUT\_FILENO, buffer,nread) != nread){

            perror("write error");

        }

   }

}

```

# **Глава 8 “Применение С++”.**

**Язык программирования С++ применяется в различных отраслях, например:**

С++ используется в широком диапазоне областей, начиная от создания системного программного обеспечения до написания игр и программ для мобильных устройств.

**Системное программное обеспечение**

С++ является одним из предпочтительных языков для создания системного программного обеспечения, такого как операционные системы, драйверы устройств, и т.д. Это связано с тем, что С++ предоставляет программистам полный контроль над системой, включая доступ к памяти и другим ресурсам.

**Игры**

C++ часто используется для написания игр благодаря своей быстроте и мощности. Большинство современных компьютерных игр были разработаны на С++.

**Мобильные устройства**

С++ используется для создания приложений для мобильных устройств, таких как Android и iOS. Это связано с тем, что C++ обеспечивает высокую производительность, что важно для приложений, работающих на мобильных устройствах.

#  **Заключение**

Язык программирования С++ является одним из наиболее распространенных и перспективных языков, который предоставляет программистам множество возможностей для создания высококачественного программного обеспечения. Правильное использование его особенностей может значительно улучшить производительность и эффективность программ. Однако, для использования языка С++ нужны достаточные знания и опыт программирования. Данный проект предлагает основные сведения, которые помогут программистам овладеть языком и успешно использовать его в своей работе.