Язык программирования C++

История C/C++

ЯЗЫК С

- Язык С создан в начале 70х годов, когда **Кен Томпсон** и **Дэннис Ритчи** из Bell Labs разрабатывали операционную систему **UNIX.** Сначала они создали часть компилятора С, затем использовали его для компиляции остальной части компилятора С и, наконец, применили полученный в результате компилятор для компиляции UNIX. Операционная система UNIX первоначально распространялась в **исходных кодах** на С среди университетов и лабораторий, а получатель мог откомпилировать исходный код на С в машинный код с помощью подходящего компилятора С.
- Распространение исходного кода сделало операционную систему UNIX уникальной; программист мог **изменить операционную систему**, а исходный код мог быть **перенесен с одной аппаратной платформы на другую.**
- С был **третьим языком**, который разработали Томсон и Ритчи в процессе создания UNIX; первыми двумя были, разумеется, **A и B.**

ЯЗЫК С

- Поскольку С сохранил способность прямого доступа к аппаратному обеспечению, его часто относят к языкам высокого уровня или в шутку называют *"мобильным языком ассемблера".*
- С возник как **универсальный язык системного программирования**. Но он не остался в этих рамках. К концу 80-х годов язык С, оттеснив Fortran с позиции лидера, завоевал массовую популярность среди программистов во всем мире и стал использоваться в самых различных прикладных задачах. Немалую роль здесь сыграло **распространение Unix (а значит и С) в университетской среде, где** проходило подготовку новое поколение программистов.
- Как и все языки, С постепенно совершенствовался, но большинство усовершенствований не носило радикального характера. Наиболее существенным из них, пожалуй, следует считать введение строгой спецификации типов функций, которая значительно повысила **надежность межмодульного взаимодействия на С**. Все такие усовершенствования были в 1989 году закреплены в стандарте **ANSI** ((Американский Национальный Комитет Стандартов)) который и поныне определяет язык С.

ОТ С К С++

- Ахиллесовой пятой языка С стало то, что он оказался слишком **низкоуровневым для тех задач, которые поставили на повестку дня 90-е** годы. Причем у этой проблемы есть два аспекта. С одной стороны, в язык были встроены слишком низкоуровневые средства - прежде всего это работа с памятью и адресная арифметика. Недаром **смена разрядности процессоров** очень болезненно отражается на многих С-программах. С другой стороны, в С недостает средств высокоуровневых - абстрактных типов данных и объектов, полиморфизма, обработки исключений. Как следствие, в программах на С техника реализации задачи часто доминирует над ее содержательной стороной.
- Первые попытки исправить эти недостатки стали предприниматься еще в начале 80-х годов. Уже тогда **Бьерн Страуструп** в AT&T Bell Labs стал разрабатывать расширение языка С под условным названием. Стиль ведения разработки вполне соответствовал духу, в котором создавался и сам язык С, - в него вводились те или иные возможности с целью сделать более удобной работу **конкретных людей и групп**. Первый коммерческий транслятор нового языка, получившего название С++ появился в 1983 году. Он представлял собой препроцессор, транслировавший программу в код на С. Однако фактическим рождением языка можно считать выход в 1985 году **книги Страуструпа** . Именно с этого момента С++ начинает набирать всемирную популярность.

C++

- Название C++ выдумал **Рик Масситти**. Название указывает на эволюционную природу перехода к нему от C. "++" - это **операция приращения** в C. Чуть более короткое имя C+ является синтаксической ошибкой; кроме того, оно уже было использовано как имя совсем другого языка.
- Знатоки семантики C находят, что C++ хуже, чем ++C. **Названия D** язык не получил, поскольку он является **расширением C** и в нем не делается попыток исцеляться от проблем путем выбрасывания различных особенностей.
- Изначально C++ был разработан, чтобы автору и его друзьям не приходилось программировать на ассемблере, C или других современных языках высокого уровня. Основным его предназначением было сделать **написание хороших программ более простым и приятным для отдельного программиста**. Плана разработки C++ на бумаге никогда не было; проект, документация и реализация двигались одновременно. Разумеется, внешний интерфейс C++ был написан на C++. Никогда не существовало *"Проекта C++" и "Комитета по разработке C++".*
- Поэтому C++ развивался и продолжает развиваться во всех направлениях, чтобы справляться со сложностями, с которыми сталкиваются пользователи, а также в процессе дискуссий автора с его друзьями и коллегами.

C++

- Хотя язык C++ справедливо называют **продолжением C** и любая работоспособная программа на языке C будет поддерживаться компилятором C++, **при переходе от C к C++ был сделан весьма существенный скачок.**
- Язык C++ выигрывал от своего родства с языком C в течение многих лет, поскольку многие программисты обнаружили, что для того, чтобы в полной мере воспользоваться преимуществами языка C++, им **нужно отказаться от некоторых своих прежних знаний и приобрести новые**, а именно: изучить новый способ концептуальности и решения проблем программирования. Перед тем как начинать осваивать C++, Страуструп и большинство других программистов, использующих C++ считают **изучение языка C необязательным.**
- C++ в настоящее время считается господствующим языком, используемым для разработки коммерческих продуктов, **например** 90% игр пишутся на C++.

Основы программирования на языке C++

СТРУКТУРА ПРОГРАММЫ НА ЯЗЫКЕ С++

#директивы препроцессора
#директивы препроцессора
функция а ()
операторы функции в ()
операторы
void main ( ) { //функция, с которой начинается выполнение программы
операторы
описания
присваивания
функция
оператор составной выбора циклов перехода
}

1. ДИРЕКТИВЫ ПРЕПРОЦЕССОРА

- **Директивы препроцессора** - управляют преобразованием текста программы до ее компиляции. Исходная программа, подготовленная на СИ в виде текстового файла, проходит 3 этапа обработки:

- препроцессорное преобразование текста ;
- компиляция;
- компоновка (редактирование связей или сборка).

- **#include< имя заголовочного файла>** - предназначена для включения в текст программы текста из каталога *«Заголовочных файлов»*, поставляемых вместе со стандартными библиотеками.
- Каждая библиотечная функция С++ имеет соответствующее описание в одном из заголовочных файлов. Список заголовочных файлов определен стандартом языка.

2. ФУНКЦИИ

- Программа представляет собой набор описаний и определений, и **состоит из набора функций**. Среди этих функций всегда должна быть функция с именем **main**. **Без нее программа не может быть выполнена**.
- Перед именем функции помещаются сведения о **типе возвращаемого функцией значения** (тип результата). Если функция ничего не возвращает, то указывается **тип void**: *void main ( ).* Каждая функция, в том числе и main должна иметь **набор параметров**, он может быть пустым, тогда в скобках указывается (void).
- За заголовком функции размещается тело функции. **Тело функции** - это последовательность определений, описаний и исполняемых операторов, заключенных в **фигурные скобки**.
- Каждое определение, описание или оператор заканчивается **точкой с запятой**.

3. ОПРЕДЕЛЕНИЯ, ОПИСАНИЯ, ОПЕРАТОРЫ

- **Определения** - вводят объекты (объект - это именованная область памяти, частный случай объекта - переменная), необходимые для представления в программе обрабатываемых данных. **Примером** являются
- int y = 10 ; //именованная константа
- float x ; //переменная
- **Описания** - уведомляют компилятор о свойствах и именах объектов и функций, описанных в других частях программы.
- **Операторы** - определяют действия программы на каждом шаге ее исполнения.

ПРИМЕР ПРОГРАММЫ НА С++

- *#include<stdio.h>*
- *main ()*
- *{*
- *printf(“Kak delishki?\n”);*
- *}*

Самые распространенные директивы:

#include <stdio.h>

#include <conio.h>

#include <iostream.h>

#include <math.h>

ОБЪЯСНЕНИЕ ПРИМЕРА

1. **#include<stdio.h>** // сообщает компилятору, что он должен включить информацию о стандартной библиотеке ввода-вывода. Эта строка встречается в начале многих исходных файлов Си-программ.
2. **Функция main()** //в языке С++ практически любая исполнимая программа должна иметь главную функцию – **main()**. Она вызывает другие функции, осуществляет общее управление проектом.
3. **{}** //инструкции функции заключаются в фигурные скобки. Функция main содержит только одну инструкцию: printf(“Привет, ХНУРЭ\n”);
4. **(*printf(“Привет, ХНУРЭ\n”);*)** //это вызов функции *printf* с аргументом *“Привет, ХНУРЭ\n*. Функция вызывается по имени, после которого в скобках, указывается список аргументов.
5. **\n** // В Си комбинация **\n** внутри строки означает *«литера - новая строка»*. Это значит, что при печати вызывается переход к левому краю следующей строки.

ТИПЫ ДАННЫХ В С++

- Данные отображают в программе окружающий мир. **Цель программы состоит в обработке данных.** Данные различных типов хранятся и обрабатываются по-разному.
- Тип данных определяет: **внутреннее представление данных в памяти компьютера**:

1) множество значений, которые могут принимать величины этого типа;
2) операции и функции, которые можно применять к данным этого типа.

ТИПЫ ДАННЫХ В С++

- **В зависимости от требований задания** программист выбирает тип для объектов программы. Типы С++ можно разделить на **простые и составные**. К простым типам относят типы, которые характеризуются одним значением. В С++ определено 6 простых типов данных:

1. **int** (целый);
2. **char** (символьный);
3. **wchar_t** (расширенный символьный);
4. **bool** (логический);
5. **float**(вещественный);
6. **double** (вещественный с двойной точностью).

- Существует 4 спецификатора типа, уточняющих внутреннее представление и диапазон стандартных типов
- **short (короткий),**
- **long (длинный),**
- **signed (знаковый) и unsigned (беззнаковый)**

ПЕРЕМЕННЫЕ В С++

- **Переменная в С++** - именованная область памяти, в которой хранятся данные определенного типа.
- У переменной есть имя и значение. Имя служит для обращения к области памяти, в которой хранится значение.
- Перед использованием любая переменная должна быть описана. **Пример:**
- **int a; float x;**

ПЕРЕМЕННЫЕ В С++

- **Общий вид оператора описания:**
[класс памяти][const] тип имя [инициализатор];

- **Класс памяти** может принимать значения: auto, extern, static, register. Класс памяти определяет время жизни и область видимости переменной.
- Если класс памяти не указан явно, то компилятор определяет его исходя из контекста объявления. Время жизни может быть **постоянным** - в течение выполнения программы или **временным** - в течение блока.
- **Область видимости** - часть текста программы, из которой допустим обычный доступ к переменной.
- **Const** - показывает, что эту переменную нельзя изменять (именованная константа).
- При описании можно присвоить переменной начальное значение (инициализация).

КЛАССЫ ПАМЯТИ

- **auto** - автоматическая локальная переменная. Спецификатор auto может быть задан только при определении объектов блока, например, в теле функции. Этим переменным память выделяется при входе в блок и освобождается при выходе из него. Вне блока такие переменные не существуют.
- **extern** - глобальная переменная, она находится в другом месте программы (в другом файле или далее по тексту). Используется для создания переменных, которые доступны во всех файлах программы.
- **static** - статическая переменная, она существует только в пределах того файла, где определена переменная.
- **register** - аналогичны auto, но память под них выделяется в регистрах процессора. Если такой возможности нет, то переменные обрабатываются как auto.

ПРИМЕРЫ

```
int a; //глобальная переменная
void main(){
int b; //локальная переменная
extern int x; //переменная x определена в другом месте
static int c; //локальная статическая переменная
a=1; //присваивание глобальной переменной
int a; //локальная переменная a
a=2; //присваивание локальной переменной
::a=3; //присваивание глобальной переменной
}
int x=4; //определение и инициализация x
```

ЗНАКИ ОПЕРАЦИЙ В С++

- Знаки операций обеспечивают формирование выражений. Выражения состоят из операндов, знаков операций и скобок. Каждый операнд является, в свою очередь, выражением или частным случаем выражения - константой или переменной.

1. **Бинарные операции:**

- Аддитивные: +, - (операции сложения и вычитания);
- Мультипликативные: *, / (умножение, деление операндов);

2. **Поразрядные операции:**

- & - поразрядная конъюнкция (И);
- | - поразрядная дизъюнкция (ИЛИ);

ОПЕРАЦИИ В С++

1) АРИФМЕТИЧЕСКИЕ

Символ операции	Значение
*	Умножение
/	Деление
%	Остаток от деления
+	Сложение
-	Вычитание

2) ОПЕРАЦИИ СРАВНЕНИЯ И ЛОГИЧЕСКИЕ ОПЕРАЦИИ

Символ операции	Значение
!	Логическое НЕ
<	Меньше
<=	Меньше, либо равно
>	Больше
>=	Больше, либо равно
==	Равно
!=	Не равно
&&	Логическое И
II	Логическое ИЛИ

3) ОПЕРАЦИИ ПРИСВАИВАНИЯ

- **=, +=, -=, +=** и т.д.
- Формат операции простого присваивания:

  операнд1=операнд2
- **Леводопустимое значение (L-значение) -** выражение, которое адресует некоторый участок памяти, т. е. в него можно занести значение. Это название произошло от операции присваивания, т. к. именно левая часть операции присваивания определяет, в какую область памяти будет занесен результат операции.
- Переменная - это частный случай леводопустимого выражения.