Алгоритмический язык «С»

Тема 1.1 Введение

Язык программирования (programming language) — формализованный язык для описания программ и алгоритмов решения задач на ЭВМ. Языки программирования являются искусственными; в них синтаксис и семантика строго определены, они не допускают свободного толкования выражения, что характерно для естественного языка. Языки программирования разделяются на две основные категории языки высокого уровня и языки низкого уровня:
Язык высокого уровня (high-level language) — язык программирования, средства которого обеспечивают описание задачи в наглядном, легко воспринимаемом виде, удобном для программиста. Он не зависит от внутренних машинных кодов ЭВМ любого типа, поэтому программы, написанные на языках высокого уровня, требуют перевода в машинные коды программами транслятора либо интерпретатора. К языкам высокого уровня относят Фортран, ПЛ/1, Бейсик, Паскаль, Си, Ада. Язык низкого уровня (low-level language) — язык программирования, предназначенный для определенного типа ЭВМ и отражающий его внутренний машинный код.

Различают также следующие виды языков программирования:
Алгоритмический язык (algorithmic language) — совокупность символов, соглашений и правил, используемых для однозначного описания алгоритмов и обычно являющаяся часть языка программирования;
Неалгоритмический язык (nonalgorithmic language) — язык программирования, тексты которого не содержат указаний на порядок выполнения операций и служат лишь исходным материалом для синтеза алгоритма решения задачи;
Автономный язык (freestanding language) — специализированный язык высокого уровня, в замкнутых СУБД (СУБД с автономным языком);
Базовый язык (base language) — машинный язык, общий для семейства ЭВМ, а также язык программирования в СУБД с автономным языком;
Гибридный (комбинированный) язык (hybrid language) — язык программирования, использующий также средства другого языка;
Графический язык (graphic language) — язык программирования, предназначенный для написания программ машинной графики и пользования ими;
Декларативный (непроцедурный) язык (declarative (nonprocedural) language) — язык программирования, который позволяет задавать связи и отношения между объектами и величинами, но не определяет последовательность выполнения действий (например, языки Пролог, QBE);
Императивный (процедурный) язык (imperative language) — язык программирования, который позволяет в явной форме (при помощи задания выполняемых операторов) определять действия и порядок (последовательность) их выполнения;
Исходный язык (source language) — язык программирования, на котором написана программа, в отличие от машинного языка, на котором программы выполняются компьютером. Исходные языки классифицируются на языки высокого уровня и языки низкого уровня;
Машинно-зависимый язык (машинно-ориентированный язык; computer-sensitive language, computer-oriented language) — язык программирования, учитывающий структуру и характеристики ЭВМ определенного типа или конкретной ЭВМ;¶Машинно-независимый язык (machine-independent language) — язык программирования, структура и средства которого не связаны с конкретной ЭВМ и позволяют выполнять составленные на нем программы на любой ЭВМ, снабженной трансляторами с этого языка;

Машинный язык (абсолютный язык, computer machine language) — язык программирования, предназначенный для представления программ в форме, обеспечивающей возможность их выполнения техническими средствами;¶Общесетевой командный язык (CNCL, Common Network-Command language) — стандартный в рамках вычислительной сети язык диалогового (интерактивного) поиска данных, предназначенный для унификации работы пользователей с неоднородными базами данных, управляемых различными СУБД;¶Общий язык (common language) — машинный язык, общий для группы ЭВМ и используемых ими внешних устройств;¶Проблемно-ориентированный язык (problem-oriented language) — язык программирования, предназначенный для решения определенного класса задач (проблем);¶Процедурный язык (процедурно-ориентированный язык, procedure-oriented language) — проблемно-ориентированный язык программирования, который облегчает выражение процедуры, как точного алгоритма;¶Символический язык (язык символического кодирования, symbolic language) — язык программирования, ориентированный на конкретные ЭВМ и основанный на кодировании машинных операций при помощи определенного набора символов;¶Системный язык (system language) — язык общения оператора ЭВМ с вычислительной системой, представляющий собой совокупность команд оператора и сообщений системы;¶Специализированный язык (special language) — язык программирования, ориентированный на решение определенного круга задач;¶Сценарный язык (script language) — язык, предназначенный для написания скриптов: программ, управляющих несколькими другими программами, в том числе написанными на разных языках программирования. Сценарные языки применяют для реализации системной интеграции разнородных программных компонентов и сред. К сценарным языкам относятся PHP, Python, Perl, Tcl, Lua, Rep, Ruby, Pike. Формальный язык (formal language) — язык программирования, построенный по правилам некоторого логического исчисления или формальной грамматики (formal grammar), представляющей собой систему правил построения в заданном алфавите конечных знаковых последовательностей, множество которых образует формальный язык;¶Эталонный язык (reference language) — язык программирования, являющийся основой для всех его конкретных версий, являющихся вариантами адаптации эталонного языка к определенным условиям применения и назначения;¶Язык ассемблера (assembler language) — универсальный язык программирования, относящийся к категории языков низкого уровня, структура которого определяется форматами команд, данными машинного языка и архитектурой ЭВМ. Язык ассемблера используется программистами в тех случаях, когда невозможно применение языка высокого уровня или требуются эффективные программы в машинных кодах.¶Язык конструирования интерактивных технологий — в СУБД язык программирования, предназначенный для описания технологических процессов обработки данных с учетом разделения характера операций по их типам, а также обеспечения диалога с администратором системы;¶Язык манипулирования данными (Data Manipulation Language, DML) — в СУБД язык программирования, предназначенный для обращения к базе данных и выполнения поиска, чтения и модификации ее записей;¶Язык меню (menu language) — язык диалога пользователя с системой, основанный на использовании меню;¶Язык обработки списков (list language) — специализированный язык программирования, предназначенный для описания процессов обработки данных, представленных в виде списков объектов;¶Язык общего назначения (универсальный язык, universal programming language) — язык программирования, ориентированный на решение задач из любой области и объединяющий на единой методической основе существенные свойства и средства машинно- и проблемно-ориентированных языков программирования (например, язык ассемблера, ПЛ/1);¶Язык ориентированный на пользователя (user-

oriented language) — слабоформализованный язык программирования, близкий к естественному языку;¶Язык описания данных (Data Description Language, DDL) — язык программирования, предназначенный для описания «концептуальной схемы» базы данных, создавался под большим влиянием XML Schema Language и RDF (Resource Description Framework);¶Язык описания хранения данных (Data Storage Description Language, DSDL) — язык программирования, предназначенный для описания физической структуры (схемы) базы данных;¶Язык описания страниц (Page Description Language, PDL) — специализированный язык программирования, предназначенный для печатающих устройств. Он предусматривает возможность использования изображений в формате, независимом от параметров устройства отображения. Наиболее известным языком такого типа является PostScript. Языком описания страниц называют также систему для кодировки документов, которая позволяет точно описать ее внешний вид после подготовки к выводу на печать или на дисплей. Примером использования такого языка служит PDF (Portable Document Format), разработанный Adobe для хранения и представления изображений страниц;¶Язык представления знаний (Knowledge Representation Language, KRL) — декларативный или декларативно-процедурный язык программирования, предназначенный для представления знаний в памяти ЭВМ (например, языки Лисп и Пролог);¶Язык публикаций (publication language) — язык программирования, используемый для публикации алгоритмов и программ;¶Язык реального времени (real-time language) — язык программирования, используемый для программирования задач, в которых критическим является время реакции ЭВМ на сигналы, требующие от нее немедленных действий (например, язык Ада);¶Язык спецификаций (specification language) — декларативный язык программирования для задания спецификаций программ;¶Язык управления заданиями (job-control language) — язык программирования, на котором записывается последовательность команд, управляющих выполнением задания. В отличие от обычных языков программирования, в которых объектами описания являются элементы, связанные с решением отдельной задачи, в языках управления заданиями объектами являются целые программы и выходные потоки данных, обработанных этими программами;¶Язык управления пакетом (batch control language) — набор команд, директив, квалификаторов и правил их использования для управления пакетной обработкой данных;¶Язык функционального программирования, функциональный язык (functional language) — декларативный язык программирования, основанный на понятии функций, которые задают зависимость, но не определяют порядок вычислений.

Си++ (англ. C++) — компилируемый строго типизированный язык программирования общего назначения. Поддерживает разные парадигмы программирования: процедурную, обобщённую, функциональную; наибольшее внимание уделено поддержке объектно-ориентированного программирования.

Разработка языка началась в 1979 году. Целью создания C++ было дополнение C возможностями, удобными для масштабной разработки ПО, с сохранением гибкости, скорости и портабельности C. Вместе с тем создатели C++ стремились сохранить совместимость с C: синтаксис первого основан на синтаксисе последнего, и большинство программ на C будут работать и как C++. Изначально новый язык назывался “C с классами”, но затем имя было изменено на C++ — это должно было подчеркнуть как его происходжение от C, так и его превосходство над последним.

Первый выпуск C++ для коммерческого использования состоялся в 1985 году, вместе с публикацией книги “The C++ Programming Language”, которая на долгое время

стала его неофициальным стандартом. В 1989 году вышла вторая версия языка в сопровождении книги “The Annotated C++ Reference Manual”.

В 1990-х годах язык стал одним из наиболее широко используемых языков программирования общего назначения. Первым официальным стандартом языка стал ISO/IEC 14882:1998, более известный как C++98. В 2003 году была принята его дополненная версия, C++03, а в 2005 году был опубликован “Library Technical Report 1” (сокращенно TR1) — документ, описывающий расширения стандартной библиотеки. TR1 не является стандартом, но большинство актуальных компиляторов C++ поддерживает его. Наконец, в 2011 году был принят текущий стандарт, C++11.

Нововведениями C++ в сравнении с C являются:

- поддержка объектно-ориентированного программирования через классы. C++ предоставляет все четыре возможности ООП — абстракцию, инкапсуляцию, наследование (в том числе и множественное) и полиморфизм.
- поддержка обобщённого программирования через шаблоны функций и классов;
- стандартная библиотека C++ состоит из стандартной библиотеки C (с некоторыми модификациями) и библиотеки шаблонов (Standard Template Library, STL), которая предоставляет обширный набор обобщенных контейнеров и алгоритмов;
- дополнительные типы данных;
- обработка исключений;
- виртуальные функции;
- пространства имён;
- встраиваемые (inline) функции;
- перегрузка (overloading) операторов;
- перегрузка имён функций;
- ссылки и операторы управления свободно распределяемой памятью.

Программа, подготовленная на языке высокого уровня, проходит несколько этапов:

1. этап. В текстовом редакторе пишется ***исходный код*** программы на алгоритмическом языке (source code) и сохраняется в файле с расширением *.pas.

2 этап. Трансляция, происходит преобразование ***исходного кода***программы (source code) в ***объектный код***(object code), т.е. происходит проверка синтаксиса написания операторов, и если ошибок в написании нет, осуществляется перевод на язык машинных кодов. Файл объектного кода имеет расширение *.obj;

Трансляторы предназначены для проверки правильности написания операторов и преобразования программ, написанных на языках программирования, в программы на машинном языке. Программа, подготовленная на каком-либо языке программирования, называется ***исходным модулем***. В качестве входной информации трансляторы применяют исходные модули и формируют в результате своей работы ***объектные модули***, являющиеся входной информацией для редактора связей. Объектный модуль содержит текст программы на машинном языке и дополнительную информацию, обеспечивающую настройку модуля по месту его загрузки и объединение этого модуля с другими независимо оттранслированными модулями в единую программу.

Трансляторы делятся на два класса: ***компиляторы*** (compiler) и ***интерпретаторы*** (interpreter). Компиляторы транслируют**всю** программу, но без ее выполнения. Интерпретаторы, в отличие от компиляторов, выполняют **пооператорный**перевод на машинный языки выполнение всей программы.

3. этап. ***Компоновка,*** когда происходит обработка объектного кода ***редактором связей***, специальной программой осуществляющей построение ***загрузочного модуля***(load module), пригодного к выполнению (рис 16.).

Компоновщик, или редактор связей - системная обрабатывающая программа, редактирующая и объединяющая объектные (ранее оттраслированные) модули в единые загрузочные, готовые к выполнению программные модули. Загрузочный модуль может быть помещен ОС в основную память и выполнен.

Получив исполняемый модуль, не спешите радоваться. К сожалению, устранение синтаксических ошибок еще не гарантирует того, что программа будет хотя бы запускаться, не говоря уже о правильности работы. Поэтому обязательным этапом процесса разработки является ***отладка***. На этапе ***отладки***, используя описание алгоритма, выполняется контроль правильности функционирования, как отдельных участков кода, так и всей программы в целом.

Правила записи программы на языке С++

Любой файл начинается с директив **#include**, вставляющих в текст программы так называемые заголовочные файлы, которые содержат описания функций, используемых в этом файле. В нашем примере это описания стандартных функций ввода-вывода и математических функций Далее следует заголовок главной функции программы main, операторы описания типов данных и исполняемые операторы. Директивы **#include** должны всегда начинаться с новой строки, остальные же операторы программы могут иметь произвольное положение в файле вплоть до записи в одну строку. При записи не допускается разрывать слова, числа, двухсимвольные операции. Пробелы используются для отделения слов друг от друга и для придания тексту большей выразительности и могут использоваться везде, за исключением случаев, описанных в предыдущем абзаце. Там, где допускается один пробел, можно поставить любое их количество. В любом месте программы, там, где может стоять пробел, допускается записывать комментарии. Комментарии должны помогать понять смысл выполняемых программой действий и являются обязательной ее частью. При написании комментариев следует придерживаться ряда простых истин:

- программы читаются людьми, компьютеру комментарии не нужны;
- всегда необходимы вводные комментарии, в которых указывается назначение программы, ее автор, дата написания и изменения, краткое описание алгоритма, входных и выходных данных, основных переменных и вызываемых функций
- комментарии должны содержать дополнительную информацию, а не перефразировать программу;
- комментарии должны быть расположены так, чтобы программа не была менее наглядной;
- неправильные комментарии хуже, чем их отсутствие.

Стиль программирования, расположение операторов в строках, использование пробелов, выбор имен переменных и т. д. должны быть направлены на то, чтобы сделать программу более понятной людям, ее читающим.

Алфавит языка Си:

Множество символов языка С включает:

- прописные буквы латинского алфавита;

- строчные буквы латинского алфавита;
- арабские цифры;
- разделители: , . ; : ? ! ' " | / \ ~ _ ^ ( ) { } [ ] < > # % & - = + *

Остальные символы могут быть использованы только в символьных строках, символьных константах и комментариях. Язык C++ различает большие и маленькие буквы, таким образом, *name* и *Name* – разные идентификаторы.

Список ключевых слов C++

asm	else	new	this
auto	enum	operator	throw
bool	explicit	private	true
break	export	protected	try
case	extern	public	typedef
catch	false	register	typeid
char	float	reinterpret_cast	typename
class	for	return	union
const	friend	short	unsigned
const_cast	goto	signed	using
continue	if	sizeof	virtual
default	inline	static	void
delete	int	static_cast	volatile
do	long	struct	wchar_t
double	mutable	switch	while
dynamic_cast	namespace	template	

Структура программы Си

Любая достаточно большая программа на Си состоит из файлов. Файлы транслируются Си-компилятором независимо друг от друга и затем объединяются программой-построителем задач, в результате чего создается файл с программой, готовой к выполнению. Файлы, содержащие тексты Си-программы, называются исходными.¶В языке Си исходные файлы бывают двух типов:

- заголовочные, или h-файлы
- файлы реализации, или Си-файлы

Имена заголовочных файлов имеют расширение ".h". Имена файлов реализации имеют расширения ".c".

Заголовочные файлы содержат только описания. Прежде всего, это прототипы функций. Также в h-файлах описываются имена и типы внешних переменных, константы, новые типы, структуры и т.п. При трансляции заголовочных файлов, как правило, никакие объекты не создаются.

Файлы реализации содержат тексты функций и определения глобальных переменных.

Представление исходных текстов в виде заголовочных файлов и файлов реализации необходимо для создания больших проектов, имеющих модульную структуру. Заголовочные файлы служат для передачи информации между модулями. Файлы реализации - это отдельные модули, которые разрабатываются и транслируются независимо друг от друга и объединяются при создании выполняемой программы.¶Файлы реализации могут подключать описания, содержащиеся в заголовочных файлах. Сами заголовочные файлы также могут использовать другие заголовочные файлы. Заголовочный файл подключается с помощью директивы препроцессора #include. Например, описания стандартных функций ввода-вывода включаются с помощью строки:

```
#include <stdio.h>
```

Имя h-файла записывается в угловых скобках, если этот h-файл является частью стандартной Си-библиотеки и расположен в одном из системных каталогов. Имена h-файлов, созданных самим программистом в рамках разрабатываемого проекта и расположенных в текущем каталоге, указываются в двойных кавычках, например:

```
#include "mylib.h"
```

Препроцессор - это программа предварительной обработки текста непосредственно перед трансляцией. Команды препроцессора называются директивами. Директивы препроцессора содержат символ # в начале строки.