Лекция 2

Основы программирование на языке C++

Введение

Microsoft Visual Studio - это набор инструментов разработки, основанных на использовании компонентов и других технологий для создания мощных, производительных приложений.

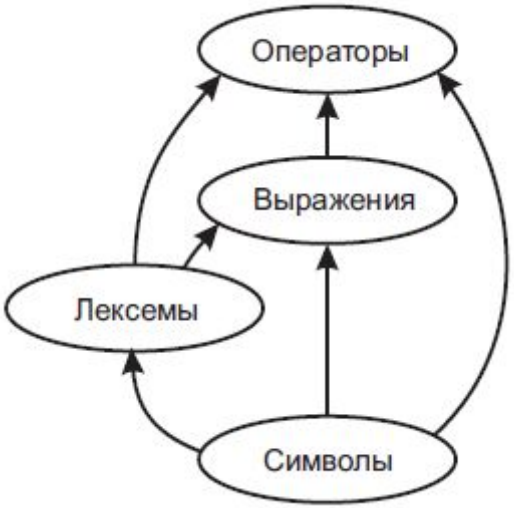
Кроме того, среда Visual Studio оптимизирована для совместного проектирования, разработки и развертывания корпоративных решений.

Также Visual Studio позволяет создавать проекты, имеющие пользовательский интерфейс (GUI), работая с разными компонентами, такими как формы, кнопки, списки, меню и т.д.

Содержание

Состав языка C++

В тексте на любом естественном языке можно выделить четыре основных элемента: символы, слова, словосочетания и предложения. Подобные элементы содержит и алгоритмический язык, только слова называют лексемами (элементарными конструкциями), словосочетания - выражениями, а предложения - операторами. Лексемы образуются из символов, выражения - из лексем и символов, а операторы - из символов, выражений и лексем



Состав языка C++

- **Алфавит языка**, или его символы - это основные неделимые знаки, с помощью которых пишутся все тексты на языке.
- **Лексема**, или элементарная конструкция, - минимальная единица языка, имеющая самостоятельный смысл.
- **Выражение** задает правило вычисления некоторого значения.
- **Оператор** задает законченное описание некоторого действия.

Алфавит языка

Алфавит языка C++ включает в себя:

- прописные и строчные латинские буквы и знак подчеркивания;
- арабские цифры от 0 до 9, шестнадцатеричные цифры от A до F;
- специальные знаки:
- « { } , | [ ] ( ) + - / % * .
- ' : ? < = > ! & # ~ ; ^
- пробельные символы: пробел, символы табуляции, символы перехода на новую строку.

Идентификаторы

Идентификатор - это имя программного объекта. В идентификаторе могут использоваться латинские буквы, цифры и знак подчеркивания. Прописные и строчные буквы различаются, например, **sysop**, **SySoP** и **SYSOP** - три различных имени.

Первым символом идентификатора может быть буква или знак подчеркивания, но не цифра. Пробелы внутри имен не допускаются.

Длина идентификатора по стандарту не ограничена, но некоторые компиляторы и компоновщики налагают на нее ограничения. Идентификатор создается на этапе объявления переменной, функции, типа и т. п., после этого его можно использовать в последующих операторах программы.

Идентификаторы

При выборе идентификатора необходимо иметь в виду следующее:

- идентификатор не должен совпадать с ключевыми словами и именами используемых стандартных объектов языка;
- не рекомендуется начинать идентификаторы с символа подчеркивания, поскольку они могут совпасть с именами системных функций или переменных, и, кроме того, это снижает мобильность программы;
- на идентификаторы, используемые для определения внешних переменных, налагаются ограничения компоновщика (использование различных компоновщиков или версий компоновщика накладывает разные требования на имена внешних переменных).

Ключевые слова

Ключевые слова - это зарезервированные идентификаторы, которые имеют специальное значение для компилятора.

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	

Список ключевых слов языка 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	

Знаки операций

Знак операции - это один или более символов, определяющих действие над операндами. Внутри знака операции пробелы не допускаются. Операции делятся на унарные, бинарные и тернарную по количеству участвующих в них операндов. Один и тот же знак может интерпретироваться по-разному в зависимости от контекста. Все знаки операций за исключением [ ], ( ) и ? : представляют собой отдельные лексемы.

Знаки отношений

== эквивалентность

!= не равно

< меньше;

> больше;

<=меньше или равно;

>= больше или равно.

Арифметические знаки

* — умножение;

/ — деление;

+ — сложение;

— — вычитание;

% — остаток от целочисленного деления.

++ — инкрементирование (увеличение на 1);

−− — декрементирование (уменьшение на 1);

— — изменение знака.

Логические знаки

&& — И (бинарная) — требуется одновременное выполнение всех операций отношения;

|| — ИЛИ (бинарная) — требуется выполнение хотя бы одной операции отношения;

! — НЕ (унарная) — требуется невыполнение операции отношения.

Константы

Константами называют неизменяемые величины. Различаются целые, вещественные, символьные и строковые константы. Компилятор, выделив константу в качестве лексемы, относит ее к одному из типов по ее внешнему виду.

Константа	Формат	Примеры
Целая	Десятичный: последовательность десятичных цифр, начинающаяся не с нуля, если это не число нуль	8, 0, 199226
	Восьмеричный: нуль, за которым следуют восьмеричные цифры (0,1,2,3,4,5,6,7)	01, 020, 07155
	Шестнадцатеричный: 0x или 0X, за которым следуют шестнадцатеричные цифры (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F)	0xA, 0x1B8, 0X00FF

Константы

Константа	Формат	Примеры
Вещественная	Десятичный: [цифры].[цифры] Экспоненциальный: [цифры][.][цифры]{E|e}[+|-][цифры]	5.7, .001, 35. 0.2E6, .11e-3, 5E10
Символьная	Один или два символа, заключенных в апострофы	'A', 'ю', '*', 'db', '\0', '\n', '\012', '\x07\x07'
Строковая	Последовательность символов, заключенная в кавычки	«Здесь был Vasia», «\tЗначение r=\0xF5\n»

Символьные константы

Символьные константы, состоящие из одного символа, занимают в памяти один байт и имеют стандартный тип **char**. Двухсимвольные константы занимают два байта и имеют тип **int**, при этом первый символ размещается в байте с меньшим адресом (о типах данных рассказывается в следующем разделе).

Символ обратной косой черты используется для представления:

- кодов, не имеющих графического изображения (например, \a - звуковой сигнал, \n - перевод курсора в начало следующей строки);
- символов апострофа ('), обратной косой черты (\ ), знака вопроса (?) и кавычки (");
- любого символа с помощью его шестнадцатеричного или восьмеричного кода, например, \073, \0xF5. Числовое значение должно находиться в диапазоне от 0 до 255.

Последовательности символов, начинающиеся с обратной косой черты, называют управляющими, или escape-последовательностями.

Изображение	Шестнадцатеричный код	Наименование
\a	7	Звуковой сигнал
\b	8	Возврат на шаг
\f	C	Перевод страницы (формата)
\n	A	Перевод строки
\r	D	Возврат каретки
\t	9	Горизонтальная табуляция
\v	B	Вертикальная табуляция
\\	5C	Обратная косая черта
\'	27	Апостроф
\"	22	Кавычка
\?	3F	Вопросительный знак
\0ddd	-	Восьмеричный код символа
\0xddd	ddd	Шестнадцатеричный код символа

Управляющие последовательности

Управляющие последовательности могут использоваться и в строковых константах, называемых иначе строковыми литералами. Например, если внутри строки требуется записать кавычку, ее предваряют косой чертой, по которой компилятор отличает ее от кавычки, ограничивающей строку:

- "Издательский дом \"Питер\""

Все строковые литералы рассматриваются компилятором как *различные объекты*. Строковые константы, отделенные в программе только пробельными символами, при компиляции объединяются в одну. Длинную строковую константу можно разместить на нескольких строках, используя в качестве знака переноса обратную косую черту, за которой следует перевод строки. Эти символы игнорируются компилятором, при этом следующая строка воспринимается как продолжение предыдущей. Например, строка:

- "Никто не доволен своей \
- внешностью, но все довольны \
- своим умом"

полностью эквивалентна строке:

- "Никто не доволен своей внешностью, но все довольны своим умом"

Комментарии

Комментарий либо начинается с двух символов «прямая косая черта» (//) и заканчивается символом перехода на новую строку, либо заключается между символами - скобками /* и */. Внутри комментария можно использовать любые допустимые на данном компьютере символы, а не только символы из алфавита языка C++, поскольку компилятор комментарии игнорирует. Вложенные комментарии - скобки стандартом не допускаются, хотя в некоторых компиляторах разрешены.

Стандартные типы данных языка C++

Основная цель любой программы состоит в обработке данных. Данные различного типа хранятся и обрабатываются по-разному. В любом алгоритмическом языке каждая константа, переменная, результат вычисления выражения или функции должны иметь определенный тип.

Тип данных определяет:

- внутреннее представление данных в памяти компьютера;
- множество значений, которые могут принимать величины этого типа;
- операции и функции, которые можно применять к величинам этого типа.

Типы данных языка C++

Все типы языка C++ можно разделить на основные и составные. В языке C++ определено шесть основных типов данных для представления целых, вещественных,

символьных и логических величин. На основе этих типов программист может вводить описание составных типов. К ним относятся массивы, перечисления, функции, структуры, ссылки, указатели, объединения и классы.

Основные типы данных

Основные (стандартные) типы данных часто называют арифметическими, поскольку их можно использовать в арифметических операциях. Для описания основных типов определены следующие ключевые слова:

- int (целый);
- char (символьный);
- wchar_t (расширенный символьный);
- bool (логический);
- float (вещественный);
- double (вещественный с двойной точностью).

Первые четыре типа называют целочисленными (целыми), последние два - типами с плавающей точкой. Код, который формирует компилятор для обработки целых величин, отличается от кода для величин с плавающей точкой.

Существует четыре спецификатора типа, уточняющих внутреннее представление и диапазон значений стандартных типов:

- short (короткий);
- long (длинный);
- signed (знаковый);
- unsigned (беззнаковый).

По умолчанию все *целочисленные* типы считаются знаковыми, то есть спецификатор signed можно опускать.

Под величину *символьного* типа отводится количество байт, достаточное для размещения любого символа из набора символов для данного компьютера, что и обусловило название типа. Как правило, это 1 байт.

Тип char, как и другие целые типы, может быть со знаком или без знака. В величинах со знаком можно хранить значения в диапазоне от -128 до 127. При использовании спецификатора unsigned значения могут находиться в пределах от 0 до 255. Этого достаточно для хранения любого символа из 256-символьного набора ASCII. Величины типа char применяются также для хранения целых чисел, не превышающих границы указанных диапазонов.

Тип wchar_t предназначен для работы с набором символов, для кодировки которых недостаточно 1 байта, например, Unicode. Размер этого типа зависит от реализации; как правило, он соответствует типу short. Строковые константы типа wchar_t записываются с префиксом L, например, L»Gates».

Величины логического типа могут принимать только значения true и false, являющиеся зарезервированными словами. Внутренняя форма представления значения false - 0 (нуль). Любое другое значение интерпретируется как true. При преобразовании к целому типу true имеет значение 1.

Стандарт C++ определяет три типа данных для хранения вещественных значений: float, double и long double.

Типы данных с *плавающей точкой* хранятся в памяти компьютера иначе, чем целочисленные. Внутреннее представление вещественного числа состоит из двух частей - мантиссы и порядка. В IBM PC-совместимых компьютерах величины типа float занимают 4 байта, из которых один двоичный разряд отводится под знак мантиссы, 8 разрядов под порядок и 23 под мантиссу. Мантисса - это число, большее 1.0, но меньшее 2.0. Поскольку старшая цифра мантиссы всегда равна 1, она не хранится.

Для величин типа double, занимающих 8 байт, под порядок и мантиссу отводится 11 и 52 разряда соответственно. Длина мантиссы определяет точность числа, а длина порядка - его диапазон. Как можно видеть из табл. 1.4, при одинаковом количестве байтов, отводимом под величины типа float и long int, диапазоны их допустимых значений сильно различаются из-за внутренней формы представления.

Спецификатор long перед именем типа double указывает, что под величину отводится 10 байтов.

Константы с плавающей точкой имеют по умолчанию тип double. Можно явно указать тип константы с помощью суффиксов F, f (float) и L, l (long). Например, константа 2E+6L будет иметь тип long double, а константа 1.82f - тип float.

Диапазоны значений простых типов данных для IBM PC

Тип	Диапазон значений	Размер (байт)
Bool	true и false	1
Signed char	-128 … 127	1
Unsigned char	0 … 255	1
Signed short int	-32 768 … 32 767	2
Unsigned short inr	0 … 65 535	2
Signed long int	-2 147 483 648 … 2 147 483 647	4
Unsigned long int	0 … 4 294 967 295	4
Float	3.4e-38 … 3.4e+38	4
Double	1.7e-308 … 1.7e+308	8
Long double	3.4e-4932 … 3.4e+4932	10

Переменные

Переменная - это 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. Во время выполнения программы значение переменной можно изменять. Перед использованием любая переменная должна быть описана.

Пример описания целой переменной с именем a и вещественной переменной **x: int a; float x;**

Примеры:

- *short int a = 1; // целая переменная a*
- *const char C = 'C'; // символьная константа C*
- *char s, sf = 'f'; // инициализация относится только к sf*
- *char t (54);*

Если тип инициализирующего значения не совпадает с типом переменной, выполняются преобразование типа по определенным. Описание переменной, кроме типа и класса памяти, явно или по умолчанию задает ее область действия. Класс памяти и область действия зависят не только от собственно описания, но и от места его размещения в тексте программы.

Область действия идентификатора - это часть программы, в которой его можно использовать для доступа к связанной с ним области памяти. В зависимости от области действия переменная может быть локальной или глобальной.

Если переменная определена внутри блока (напомню, что блок ограничен фигурными скобками), она называется **локальной**, область ее действия - от точки описания до конца блока, включая все вложенные блоки. Если переменная определена вне любого блока, она называется глобальной и областью ее действия считается файл, в котором она определена, от точки описания до его конца.

Класс памяти определяет время жизни и область видимости программного объекта (в частности, переменной). Если класс памяти не указан явным образом, он определяется компилятором исходя из контекста объявления. Время жизни может быть постоянным (в течение выполнения программы) и временным (в течение выполнения блока).

Спецификаторы класса памяти

Для задания класса памяти используются следующие спецификаторы:

- auto - автоматическая переменная. Память под нее выделяется в стеке и при необходимости инициализируется каждый раз при выполнении оператора, содержащего ее определение. Освобождение памяти происходит при выходе из блока, в котором описана переменная.
- extern - означает, что переменная определяется в другом месте программы (в другом файле или дальше по тексту). Используется для создания переменных, доступных во всех модулях программы, в которых они объявлены.
- static - статическая переменная. Время жизни - постоянное. Инициализируется один раз при первом выполнении оператора, содержащего определение переменной.
- register - аналогично auto, но память выделяется по возможности в регистрах процессора. Если такой возможности у компилятора нет, переменные обрабатываются как auto.

Описание переменной

- Имя переменной должно быть уникальным в своей области действия (например, в одном блоке не может быть двух переменных с одинаковыми именами).
- Описание переменной может выполняться в форме объявления или определения. Объявление информирует компилятор о типе переменной и классе памяти, а определение содержит, кроме этого, указание компилятору выделить память в соответствии с типом переменной. В C++ большинство объявлений являются одновременно и определениями.
- Переменная может быть объявлена многократно, но определена только в одном месте программы, поскольку объявление просто описывает свойства переменной, а определение связывает ее с конкретной областью памяти.

Выражения

Выражения состоят из операндов, операций и скобок и используются для вычисления некоторого значения определенного типа. Каждый операнд является, в свою очередь, выражением или одним из его частных случаев - константой или переменной.

Примеры выражений:

- (a + 0.12)/6
- x && y || !z
- (t * sin(x)-1.05e4)/((2 * k + 2) * (2 * k + 3))

Операции выполняются в соответствии с приоритетами. Для изменения порядка выполнения операций используются круглые скобки. Если в одном выражении записано несколько операций одинакового приоритета, унарные операции, условная операция и операции присваивания выполняются справа налево, остальные - слева направо.

Например, a = b = c означает a = (b = c), а a + b + c означает (a + b) + c.

Порядок вычисления подвыражений внутри выражений не определен: например, нельзя считать, что в выражении (sin(x + 2) + cos(y + 1)) обращение к синусу будет выполнено раньше, чем к косинусу, и что x + 2 будет вычислено раньше, чем y + 1.

Результат вычисления выражения характеризуется значением и типом. Например, если a и b - переменные целого типа и описаны так: int a = 2, b = 5;

то выражение a + b имеет значение 7 и тип int, а выражение a = b имеет значение, равное помещенному в переменную a (в данному случае 5) и тип, совпадающий с типом этой переменной.

В выражение могут входить операнды различных типов. Если операнды имеют одинаковый тип, то результат операции будет иметь тот же тип. Если операнды разного типа, перед вычислениями выполняются преобразования типов по определенным правилам, обеспечивающим преобразование более коротких типов в более длинные для сохранения значимости и точности.

Преобразования бывают двух типов:

- изменяющие внутреннее представление величин (с потерей точности или без потери точности);
- изменяющие только интерпретацию внутреннего представления.

К первому типу относится, например, преобразование целого числа в вещественное (без потери точности) и наоборот (возможно, с потерей точности), ко второму - преобразование знакового целого в беззнаковое.

В любом случае величины типов char, signed char, unsigned char, short int и unsigned short int преобразуются в тип int, если он может представить все значения, или в unsigned int в противном случае.

После этого операнды преобразуются к типу наиболее длинного из них, и он используется как тип результата.

Контрольные вопросы

1. Что такое состав языка программирования?
2. Что включает в себя алфавит языка?
3. Что чего предназначены идентификаторы?
4. Объясните ключевые слова, константы, комментарии?
5. Для чего нужны управляющие последовательности?
6. Перечислите стандартные типы данных C++
7. Как определяются переменные?
8. Как записываются выражения?

Список литературы

- Павловская Т.А. С/С++. Программирование на языке высокого уровня / Т. А. Павловская. - СПб.: Питер, 2004. - 461 с.: ил.
- Павловская Т.А. С/С ++. Структурное программирование: Практикум / Т.А. Павловская, Ю.А. Щупак. СПб.: Питер, 2007. - 239 с.: ил.
- Павловская Т. А., Щупак Ю. А. С++. Объектно-ориентированное программирование: Практикум. - СПб.: Питер, 2006. - 265 с: ил.
- Кольцов Д.М. 100 примеров на Си. - СПб.: “Наука и техника”, 2017 - 256 с.
- 5 Доусон М. Изучаем С++ через программирование игр. - СПб.: “Питер”, 2016. - 352.
- Седжвик Р. Фундаментальные алгоритмы на С++. Анализ/Структуры данных/Сортировка/Поиск: Пер. с англ. Роберт Седжвик. - К.: Издательство “Диасофт”, 2001. - 688с.
- Сиддкхартха Р. Освой самостоятельно С++ за 21 день. - М.: SAMS, 2013. - 651 с.
- Стивен, П. Язык программирования С++. Лекции и упражнения, 6-е изд. Пер. с англ. - М.: ООО "И.Д. Вильямс", 2012. - 1248 с.
- Черносвитов, А. Visual C++: руководство по практическому изучению / А. Черносвитов . - СПб. : Питер, 2002. - 528 с. : ил.

Список литературы

- Страуструп Б. Дизайн и эволюция языка C++. - М.: ДМК, 2000. - 448 с.
- Мейерс С. Эффективное использование C++. - М.: ДМК, 2000. - 240 с.
- Бадд Т. Объектно-ориентированное программирование в действии. - СПб: Питер, 1997. - 464 с.
- Лаптев В.В. С ++. Объектно-ориентированное программирование: Учебное пособие.- СПб.: Питер, 2008. - 464 с.: ил.
- Страуструп Б. Язык программирования C++. Режим доступа: http://8361.ru/6sem/books/Straustrup-Yazyk_programmirovaniya_c.pdf.
- Керниган Б., Ритчи Д. Язык программирования Си. Режим доступа: http://cpp.com.ru/kr_cbook/index.html.
- Герберт Шилдт: C++ базовый курс. Режим доступа: https://www.bsuir.by/m/12_100229_1_98220.pdf,
- Богуславский А.А., Соколов С.М. Основы программирования на языке Си++. Режим доступа: http://www.ict.edu.ru/ft/004246/cpp_p1.pdf.
- Линский, Е. Основы C++. Режим доступа: https://www.lektorium.tv/lecture/13373.
- Конова Е. А., Поллак Г. А. Алгоритмы и программы. Язык C++: Учебное пособие. Режим доступа: https://vk.com/doc7608079_489807856?hash=e279524206b2efd567&dl=f85cf2703018eeaa2