

Лекция 2.

Общие сведения о языке C/C++, типы данных и переменные, операции и их приоритеты

История создания Си

Язык С был разработан Дэнисом Ритчи в 1970-х годах для компьютера PDP компании DEC (Digital Equipment Corporation), в котором использовалась операционная система UNIX.

Он вырос из кризиса программного обеспечения 1960-х годов и революционного перехода к структурному программированию. До этого многие программисты испытывали трудности при написании больших программ, поскольку обозначилась тенденция вырождения программной логики и появления так называемого "спагетти-кода" с большим размером процедур и интенсивным использованием оператора перехода `goto`. Такие программы были трудны для изучения и модификаций. В структурных языках программирования эта проблема решалась посредством добавления точно определенных управляющих конструкций, вызова подпрограмм с локальными переменными и других усовершенствований. С был первым языком, в котором удачно сочетались мощь, элегантность, гибкость и выразительность.

Создание ООП и С++

Одним из недостатков Си была невозможность справиться с большими программами. Если проект достигал определенного размера, то дальнейшая его поддержка и развитие были связаны с определенными трудностями.

К концу 1970-х размер проектов стал приближаться к критическому, при превышении которого методика структурного программирования и язык С давали сбой. Поэтому стали появляться новые подходы к программированию. Один из них получил название объектно-ориентированного программирования (ООП). Желание работать с объектно-ориентированной версией языка С в конце концов и привело к созданию С++.

Язык С++ был разработан Бьярни Страуструпом в 1979 г. Первоначально язык назвался "С с классами", но в 1983 году это имя было изменено на С++.

Развитие C++

C++ полностью включает элементы языка C. На протяжении 1980-х годов C++ интенсивно развивался и к началу 1990-х уже был готов для широкого использования. В наши дни язык C++ по-прежнему имеет неоспоримое превосходство при разработке высокопроизводительных программ системного уровня.

Важно понимать, что создание C++ не было попыткой изобрести совершенно новый язык программирования. Это было своего рода усовершенствование и без того очень успешного языка.

Программой на языке C/C++ является текстовый файл с произвольным именем и расширением сpp.

В начало программы включают директивы препроцессора – команды, начинающиеся с символа #.

Например:

```
#include <iostream>
```

```
#include <cmath>
```

Алфавит языка

включает:

- буквы латинского алфавита;
- цифры от 0 до 9;
- специальные знаки:

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

Комментарии

В языке Си++ имеется два вида комментариев:

многострочный – текст, размещенный между символами `/* */`;

однострочный – последовательность символов одной строки, перед которой записаны `//`

`/* Многострочный комментарий в Си и С++ */`

`// Однострочный комментарий в С++`

Идентификатор

последовательность из букв, цифр и символов подчеркивания, которая начинается не с цифры:

`MAX` `x` `y0`

Прописные и строчные буквы различаются.

Ключевые слова

идентификаторы, зарезервированные
для служебного использования

Служебные слова C++:

<code>asm</code>	<code>double</code>	<code>new</code>	<code>switch</code>
<code>auto</code>	<code>else</code>	<code>operator</code>	<code>template</code>
<code>break</code>	<code>enum</code>	<code>private</code>	<code>this</code>
<code>case</code>	<code>extern</code>	<code>protected</code>	<code>throw</code>
<code>catch</code>	<code>float</code>	<code>public</code>	<code>try</code>
<code>char</code>	<code>for</code>	<code>register</code>	<code>typedef</code>
<code>class</code>	<code>friend</code>	<code>return</code>	<code>typeid</code>
<code>const</code>	<code>goto</code>	<code>short</code>	<code>union</code>
<code>continue</code>	<code>if</code>	<code>signed</code>	<code>unsigned</code>
<code>default</code>	<code>inline</code>	<code>sizeof</code>	<code>virtual</code>
<code>delete</code>	<code>int</code>	<code>static</code>	<code>void</code>
<code>do</code>	<code>long</code>	<code>struct</code>	<code>volatile</code>
			<code>while</code>

Константы

представляют изображение фиксированного числового, символьного или строкового значения.

Бывают **целыми, вещественными, перечислимыми, символьными и строковыми.**

Целые константы

Могут быть: **десятичными, восьмеричными и шестнадцатеричными.**

- **Десятичная константа** это последовательность цифр от 0 до 9, начинающаяся не с 0, если это не число ноль. Например, **16, 484216, 0, 4.**
- **Отрицательные константы** – это константы без знака, к которым применена операция изменения знака.
- **Восьмеричные константы** начинаются с нуля, например, **012 имеет десятичное значение 10.**
- Последовательность, состоящая из цифр от 0 до 9 и латинских букв от a до f (или от A до F), перед которой расположены символы 0x называется **шестнадцатеричной константой.**

Вещественные константы

В отличие от целых, по-другому представлены в компьютере и требуют использования арифметики с плавающей точкой.

Перечислимые константы

Задаются с помощью служебного слова **enum**.

Определение перечислимой константы

```
enum {имя_1=значение_1,..., имя_N=значение_N};
```

Например:

```
enum {one = 1, two = 2, three = 3};
```

По умолчанию при отсутствии значений самый левый идентификатор будет равен 0, а каждый последующий увеличивается на 1.

Значения констант могут быть записаны в виде выражений.

Можно ввести имя типа, которое помещают между **enum** и открывающейся фигурной скобкой. Например:

```
enum day {sunday, monday, tuesday, wednesday, thursday, friday, saturday};
```

Символьные и строковые константы

Символьная константа - это один или два символа, заключенные в апострофы.

Примеры:

's', 'i', '\0', '\t' – односимвольные константы

'db', '\n\t' – двухсимвольные константы.

Строковая константа –

последовательность символов, заключенная в кавычки, в которой могут встречаться эскейп - последовательности. Транслятор добавляет в конец любой строковой константы символ '\0', т.е. **нулевой байт**.

Эскейп - последовательности

Изображение	Реакция или смысл
<code>\a</code>	Звуковой сигнал
<code>\b</code>	Возврат на шаг
<code>\r</code>	Возврат каретки
<code>\t</code>	Табуляция горизонтальная
<code>\v</code>	Табуляция вертикальная
<code>\f</code>	Перевод страницы
<code>\n</code>	Перевод строки (новая строка)
<code>\"</code>	Двойная кавычка
<code>\?</code>	Вопросительный знак
<code>\'</code>	Апостроф
<code>\\</code>	Обратная косая черта
<code>\000</code>	Восьмеричный код символа
<code>\xhh</code>	Шестнадцатеричный код символа

Именованные константы

1. Определение:

`const тип имя = значение;`

Значение должно соответствовать типу. Если тип не указан, то по умолчанию `int`.

Примеры:

```
const double PI=3.14;
```

```
const F=76975;
```

2. Можно задавать именованные константы с помощью препроцессорной директивы

`#define имя значение`

Переменные

Переменная – именованная область памяти, которой можно присваивать разные значения, определенного типа. Перед использованием переменную нужно определить, выделив ей память. Размер памяти зависит от типа переменной. Определение:

тип список_имен;

Имена отделяются запятыми. Например:

```
int a,b; float min;
```

Основные типы данных и диапазоны значений

Переменные	Тип данных	Размер, бит (байт)	Диапазон значений
Символьные	unsigned char	8 бит(1 байт)	0...255
	char	8 бит(1 байт)	-128...127
Целочисленные	unsigned int	16 бит(2 байта)	0...65535
	short int	16 бит(2 байта)	-32768...32767
	int	16 бит(2 байта)	-32768...32767
	unsigned long	32 бит(4 байта)	0...4294967295
	long	32 бит(4 байта)	-2147483648... 2147483647
Перечисления	enum	16 бит(2 байта)	-32768...32767
Вещественные	float	32 бит(4 байта)	3.4E-38.....3.4E+38
	double	64 бит(8 байт)	1.7E-308...1.7E+308
	long double	80 бит(10 байт)	3.4E-4932...1.1E+4932

Знаки операций

Унарные операции			
Операция	Ранг	Ассоциативность	Действие
&	2	←	Получение адреса
*		←	Обращение по адресу, применяется к указателям
-		←	Изменение знака
~		←	Поразрядное инвертирование двоичного кода целого аргумента – побитовое отрицание

Знаки операций

Унарные операции			
Операция	Ранг	Ассоциативность	Действие
!	2	←	Логическое отрицание (НЕ).
++		←	Увеличение значения на 1
--		←	Уменьшение на 1 значения операнда
sizeof		←	Вычисление размера (в байтах) объекта того типа, который имеет операнд. <code>sizeof выражение;</code> <code>sizeof (тип);</code>
(тип) операнд		←	Операция преобразования (приведения) типа

Знаки операций

Бинарные операции			
Операция	Ранг	Ассоциативность	Действие
+	4	→	Сложение арифметических операндов или указателя с целочисленным операндом
-		→	Вычитание арифметических операндов или указателей
*	3	→	Умножение операндов арифметического типа
/		→	Деление операндов арифметического типа. Для целочисленных операндов результат целый
%		→	Получение остатка от деления целочисленных операндов

Знаки операций

Бинарные операции

Опера-ция	Ранг	Ассоциати-вность	Действие
<<	5	→	сдвиг влево битового представления значения левого операнда на количество разрядов, равное значению правого
>>	5	→	сдвиг вправо битового представления значения левого операнда на количество разрядов, равное значению правого операнда
&	8	→	поразрядная конъюнкция (И) битовых представлений значений целочисленных операндов
	10	→	поразрядная дизъюнкция (ИЛИ) битовых представлений значений целочисленных операндов
^	9	→	поразрядное исключающее ИЛИ битовых представлений значений целочисленных операндов

Знаки операций

Бинарные операции

Опера-ция	Ранг	Ассоциати-вность	Действие
<	6	→	Меньше
>		→	Больше
<=		→	Меньше или равно
>=		→	Больше или равно
==	7	→	Равно
!=		→	Не равно
&&	11	→	Конъюнкция (И), результат 0 (ложь) или 1 (истина)
	12	→	Дизъюнкция (ИЛИ), результат 0 (ложь) или 1 (истина).

Бинарные операции

=	14	←	Простое присваивание
oper=	14	←	Составное присваивание, где oper – знак (* / % + - <<>> & ^) X oper = Y равнозначно X =X oper Y
.(точка)	1	→	Прямой выбор компонента структурированного объекта
->	1	→	Косвенный выбор компонента структурированного объекта
()	1	→	Обращения к функции
[]	1	→	Индексирование массива
,	15	→	Выражения, разделенные запятыми вычисляются слева направо. Результат - тип и значение правого выражения.
?:	15	←	Вычисляется выражение перед '?', если оно истинно, то результат - выражение после '?', иначе после ':'

Пример программы

Выполнение программы начинается с главной функции, имеющей имя main

```
int main()
{
//операторы
return 0;
}
```

```
void main()
{
//операторы
}
```

Задача

Вычисление суммы двух чисел