**Программирование на С++**

**Литература:**

* Лафоре Роберт «Объектно-ориентированное программирование в С++» СПБ Издательство Питер 2003г.
* Павловская Т.А. «С/С++ программирование на языке высокого уровня» СПБ Питер 2003г.
* Страуструн Б. «Язык программирования С++» Билсом(?) СПБ 2002

**История и эволюция С++**

С 1972г. – лаборатория Bell. Цель – системное программирование

С++ 1983 – Страуструн, лаборатория Bell. Цель – реализация методологии ООП

Java – 1994, корпорация Sun. Цель – переносимость и надёжность

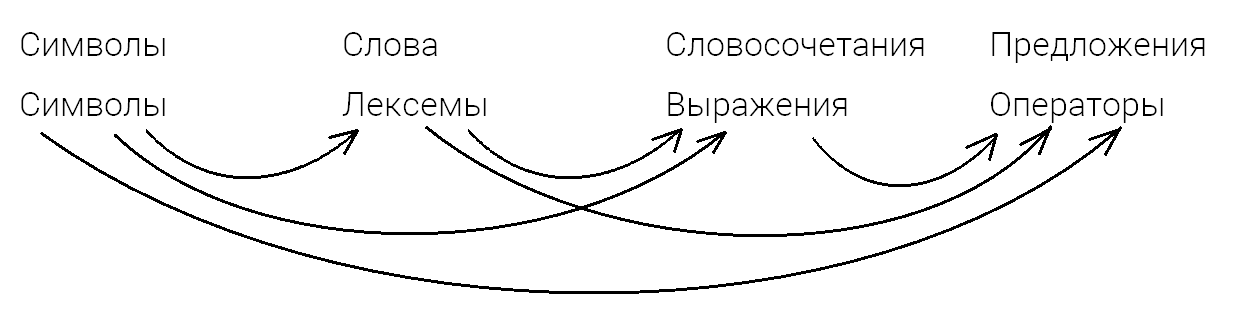
C# – 2000, Microsoft. Цель – удобство и надёжность

PHP; Perl. Цель – простота использования

**Алфавит С++**

* Прописные (заглавные) и строчные латинские буквы(регистр важен)
* Знак подчёркивания
* Цифры от 0 до 9
* Специальные знаки ( { } ; | ; [ ] ; ( ) ; + ; - ; \* ; / ; % ; \ ; ; ; ‘ ; : ; ? ; < ; = ; > ; ! ; & ; ~ ; ^ ; . ; # )
* Разделители (пробел, табуляция, конец строки)
* Все остальные символы можно использовать только в символьных строках, в символьных константах и в комментариях.

**Состоит из:**



Лексема – минимальная единица языка, имеющая самостоятельный смысл.

Выражение задаёт правило вычисления некоторого значения.

Оператор задаёт законченное описание некоторого действия.

Примеры лексем:

* Имена (идентификаторы; имена переменных, массивов и т.д.)  
  Не рекомендуется начинать с подчёркивания
* Ключевые слова (зарезервированные разработчиками слова, части операторов по типу If, else и т.д.)
* Знаки операции (одно и двухсимвольные)
* Константы
* Разделители (пробел, табуляция, конец строки – enter)
* Комментарии (однострочные «//» и многострочные «/\*….\*/» )

**Примеры неименованных констант**

Целые:

* Десятичные 123 0 98
* Восьмиричные 01 015
* Шестнадцатиричные 0хА1 0х00FF (икс может быть и большим и маленьким)

Вещественные:

* Scientific - Сайнтифик (научный) 5.8 2e-3

Символьные (пишутся в апострофах):

* Односимвольные ‘x’
* Двусимвольные ‘xy’ ‘\n’

Строковые (пишутся в кавычках):

* “привет” “1 \n 2”

**Типы в языке С++**

1. Числовые (целые и вещественные)
2. Символьные и строковые значения
3. Булевские (логические)

Типы делятся на основные (базовые) и составные.

В С++ 6 базовых типов (представляют целые, вещественные, символьные и логические величины)

На основе базовых строятся составные типы (программист может самостоятельно конструировать составные типы) Например: массивы, структуры, объединения и классы, перечисления, функции, указатели и ссылки.

Основные типы (иногда их называют арифметическими):

|  |  |  |
| --- | --- | --- |
| Тип даных | Значение | Размер (в байтах) |
| int | Целый | 2 или 4 |
| Char | Символьный | 1 |
| Wchar\_t | Расширенный символьный | 2 |
| Bool | Логический | 1 |
| Float | Вещественный | 4 |
| Double | Вещественный с двойной точностью | 8 |

Существуют 4 спецификатора типа, уточняющих внутреннее представление и диапазон значений стандартных типов:

* Short (короткий) 2 байта
* Long (длинный) 4 байта
* Signed (знаковый)
* Unsigned (беззнаковый)

По умолчанию все целые типы считаются знаковыми, т.е. спецификатор signed можно опускать.

Short int (всегда 2 байта); long int(всегда 4 байта); signed int; unsigned. Такие типы можно использовать без int.

Если переменная имеет тип int, то её размер зависит от реализации (от системы) и будет составлять 2 или 4 байта.

Bool так же зависит от реализации.

Bool относится к специальным типам.

К специальным типам те же относится void. Её используют для обозначения функций, не возвращающих значение и для не типизированных указателей.

**Переименование типов**

Typedef тип новое имя[ «размерность» ] – позволяет задать типу новое имя.

Пример: typedef integer int IUNT или typedef char Msy[100] или

или typedef strruct {  
 char fio[30];  
 int date code;  
 double salary;} Worker

**Перечисления (enum)**

При написании программ часто возникает потребность определить несколько именованных констант, для которых требуется, что бы все они имели разные значения. Но при этом конкретные значения могут быть не важны. В этом случае можно использовать перечисляемый тип данных, в котором все возможные значения перечисляются списком целочисленных констант.

Пример: enum [ имя типа ] { список констант };   
(имя типа может быть, а может и не быть; всё, что находится в квадратных скобках не обязательно)

Enum Err (Err\_Read, Err\_Write, Err Convert);   
Err error;  
 Switch (error)  
 {case Err\_Read; /x…..x/ Break;  
 case Err\_Write;/x…….x/ Break;  
 case Err\_Convert; /x…….x/ Break }

Имя типа задаётся в случае, если будем определять переменную этого типа. Такие переменные могут принимать значения только из списка констант.