Учебный Центр компьютерного обучения «СПЕЦИАЛИСТ» при МГТУ им. Н.Э.Баумана

Программирование на языке Си++

Модуль 1.

ТИПЫ ДАННЫХ, ОПЕРАЦИИ И ФУНКЦИИ В Си++

- Ссылочный тип данных
- Операции расширения контекста, `new`, `delete`
- Встроенные inline-функции
- Перегрузка функций. Аргументы по умолчанию

Немного истории

- **1979** — Сотрудник AT&T Bell Labs Бьярн Страуструп (Bjarne Stroustrup) приступает к созданию надмножества языка Си под названием «Си с классами». Задача — обогатить Си возможностями в стиле языка Simula, необходимыми для ведения крупномасштабных проектов разработки ПО
- **1983** — Новый язык получает современное название C++ (Си++) *(приписывается Рику Маскутти (Rick Mascutti))*
- **1985** — В США опубликована книга Б. Страуструпа *The C++ Programming Language*, де-факто ставшая неформальным стандартом на язык Си++
- **1998** — Международная организация по стандартизации (ISO) принимает ныне действующий стандарт ISO/IEC 14882:1998 "Standard for the C++ Programming Language"
- **2003** — Выходят в свет технические поправки в стандарт под номером ISO/IEC 14882:2003

Алфавит и комментарии в языке Си++

- **Алфавит**
  - буквы: A, B, C, …, Z, a, b, c, …, z
  - цифры: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
  - специальные символы: + – / % . ? ! " < > | \ ' _ & ~ ^
  - знаки пунктуации языка: [ ] ( ) { } , ; : … * = #
  - пробельные символы: ␣ (пробел), ↳ (символ табуляции), ↲ (символ перевода строки)
  - прочие символы — только в комментариях к тексту программы и строковых литералах
- **Комментарии**
  - многострочные комментарии:

    /* [<*произвольный текст*>] */
  - однострочные комментарии (правый ограничитель — символ конца строки):

    // [<*произвольный текст*>]

Лексемы в языке Си++. Правила выбора идентификаторов

- **Лексемы** — идентификаторы, ключевые слова, константы, операции, разделители
  - единицы текста программы, которые при компиляции воспринимаются как единое целое и по смыслу не могут быть разделены на более мелкие элементы [Под04]
- **Идентификатор** — любая последовательность букв A, B, C, …, Z, a, b, c, …, z, цифр 0, 1, …, 9 и символов подчеркивания _ , не начинающаяся с цифры, ограничения на длину которой накладываются каждым из компиляторов. Строчные и прописные буквы в идентификаторах различаются

Ключевые слова

- **Ключевое слово** — одно из слов языка, входящих в следующий список:
  - спецификаторы типов: `char`, `class`, `double`, `enum`, `float`, `int`, `long`, `short`, `struct`, `signed`, `union`, `unsigned`, `void`, `typedef`, `typeid`
  - квалификаторы объектов и типов: `const`, `friend`, `inline`, `virtual`, `volatile`
  - спецификаторы доступа: `private`, `protected`, `public`
  - квалификаторы классов памяти: `auto`, `extern`, `register`, `static`
  - операторы языка и идентификаторы специального назначения: `break`, `catch`, `continue`, `delete`, `do`, `for`, `goto`, `if`, `new`, `return`, `switch`, `throw`, `try`, `while`; `asm`, `default`, `case`, `else`, `operator`, `sizeof`, `template`, `this`
  - модификаторы и псевдопеременные: конкретный набор зависит от компилятора

Константные значения (начало)

- **Константа** — неизменяемое арифметическое значение целого, вещественного, символьного или перечислимого типа, нулевой указатель либо строковый литерал:
  - целые — записываются в системах счисления по основаниям 10, 8, 16:
    - (целочисленный) нуль в любой системе счисления — `0`
    - десятичные — последовательность десятичных цифр, не начинающаяся с нуля
    - восьмеричные — последовательность восьмеричных цифр, начинающаяся с нуля
    - шестнадцатеричные — последовательность шестнадцатеричных цифр, начинающаяся с 0x или 0X
  - вещественные — записываются в десятичной системе в следующих форматах:
    - [+|-]<*целая часть*>.[<*дробная часть*>]
    - [+|-]<*целая часть*>{e|E}[+|-]<*порядок*>
    - .[<*дробная часть*>][{e|E}[+|-]<*порядок*>]

    где<*целая часть*> есть целая часть абсолютной величины десятичной мантиссы,
    <*дробная часть*> — дробная часть абсолютной величины десятичной мантиссы,
    <*порядок*> — абсолютная величина десятичного порядка (экспоненциальной части числа)

Константные значения (окончание)

- символьные — записываются естественным образом* или посредством escape-последовательностей**, *** согласно следующим правилам:
  - * символы, имеющие экранное представление — любой входящий или не входящий в алфавит языка единичный символ в обрамлении апострофов (');
  - ** *ряд* символов, лишенных экранного представления — одна из следующих управляющих последовательностей: `'\n'` — перевод строки; `'\t'` — горизонтальная табуляция; `'\r'` — возврат каретки; `'\\'` — обратная косая черта; `'\''` — апостроф; `'\"'` — двойная кавычка; `'\0'` — нулевой символ; `'\a'` — звонок; `'\b'` — возврат на одну позицию; `'\f'` — перевод страницы; `'\v'` — вертикальная табуляция; `'\?'` — знак вопроса;
  - *** любой символ — собственный восьмеричный код в виде '\ooo', где o — цифра от 0 до 7, либо шестнадцатеричный код в виде '\xhh' или '\Xhh', где h — цифра от 0 до F;
- перечислимые — задаются в определении программистом собственного типа-перечисления;
- нулевой указатель — единственная неарифметическая константа, представимая различными компиляторами как `0`, `0L` или `NULL` (значение NULL может не совпадать с нулем (`0`) и (или) нулевым символом (`'\0'`));
- строковый литерал — заключенная в двойные кавычки (") последовательность символов, записанных по правилам для символьных констант *, **, *** без обрамляющих апострофов

Операции и разделители

- **Операция** — любая из операций, закрепленных в стандарте ANSI на язык Си, либо одна из следующих вновь добавленных операций:

```
::    .*    ->*    new    delete    typeid
```

- **Разделитель** — парный или одиночный знак пунктуации, входящий в следующий список:

```
[ ]    ( )    { }    ,    ;    :    …    *    =    #    &
```

Знаки и приоритет операций (начало)

Приоритет операций	Знаки операций	Порядок выполнения операций с равным приоритетом
1	( ) [ ] -> :: .	слева направо
2	! ~ + – ++ –– & * (<*имя типа*>) sizeof new delete <*имя типа*>()	справа налево
3	.* ->*	слева направо
4	* / %	слева направо
5	+ –	слева направо
6	<< >>	слева направо
7	< <= >= >	слева направо
8	== !=	слева направо
9	&	слева направо

Знаки и приоритет операций (окончание)

Приоритет операций	Знаки операций	Порядок выполнения операций с равным приоритетом
10	^	слева направо
11	|	слева направо
12	&&	слева направо
13	||	слева направо
14	?:	справа налево
15	= *= /= %= += −= &= ^= |= <<= >>=	справа налево
16	,	слева направо

Основные типы данных

Имя типа	Размер области памяти (бит)	Диапазон значений (для вещественных типов — по абсолютной величине)
unsigned char	8	0 ... 255, '\x00' ... '\xFF'
char, singed char	8	-128 ... 127
enum	16	-32768 ... 32767
unsigned, unsigned int	16	0 ... 65535
int, signed int	16	-32768 ... 32767
unsigned long	32	0 ... 4294967295
long, singed long	32	-2147483648 ... 2147483647
float	32	3.4E-38 ... 3.4E38
double	64	1.7E-308 ... 1.7E308
long double	80	3.4E-4932 ... 1.1E4932

Понятие ссылки. Определение ссылок (начало)

- **Ссылка** — синонимичное обозначение (псевдоним, «другое имя») существующего объекта, равноправное с основным именем. Ссылка должна быть инициализирована в момент определения таковой. Для инициализации ссылок *обычно* служат леводопустимые выражения. Значение ссылки после определения — адрес существующего объекта. Изменить значение ссылки после инициализации невозможно
- **Обращение по ссылке** не требует ее разыменования. Операции с операндом – ссылочной переменной действуют не на ссылку, а на объект, к которому та относится
- **Определение ссылки на переменную конкретного типа данных**

  <имя типа> & *<идентификатор>* *<инициализирующее выражение>*;

  Если *<инициализирующее выражение>* является праводопустимой константой, то ссылка инициализируется адресом временного объекта, созданного для размещения этой константы в памяти

Понятие ссылки. Определение ссылок (окончание)

- **Определение ссылки на указатель**

  <имя типа> &<идентификатор> <инициализирующее выражение>;*

- **Ограничения в использовании ссылок**
  - над ссылками не определены операции;
  - ссылка не может иметь тип void и не может быть создана с использованием операции new;
  - не существует конструкций типа «ссылка на ссылку», «указатель на ссылку» и «массив ссылок»

- **Определение ссылки на функцию с конкретной спецификацией параметров**

  <тип результата> (&*<идентификатор>*)
  ([*<спецификация формальных параметров>*])
  <инициализирующее выражение>;

Операции расширения контекста. Потоковый ввод-вывод в языке Си++

- **Расширение действия (перегрузка)** — возможность распространения действия стандартных операций на операнды других, «нестандартных» типов. Пример — перегрузка операций >> и << для организации потокового ввода-вывода данных базовых типов
- **Для ввода-вывода данных базовых типов** наряду с со стандартной библиотекой ANSI-функций stdio.h могут использоваться следующие объекты стандартизованных классов:
  - cin (с перегруженной операцией >>) — для получения данных базовых типов из входного потока (обычно с клавиатуры);
  - cout (с перегруженной операцией <<) — для выдачи данных базовых типов в выходной поток (обычно на экран монитора);
  - cerr (с перегруженной операцией <<) — для выдачи данных базовых типов в выходной поток сообщений об ошибках (обычно на экран монитора)

Операции `new` и `delete`

- **Операции `new` и `delete`** служат для динамического распределения памяти и являются альтернативой стандартных ANSI-функций языка Си `malloc()` и `free()`
  - выражение с операцией `new` имеет вид:

    `new` <*имя типа*> [<*инициализирующее выражение*>]
  - выражение с операцией `delete` имеет вид:

    `delete` <*указатель*>

    В случае невозможности выделить памяти выражение с операцией `new` получает значение NULL. Применение операции `delete` к указателю со значением NULL разрешено, хотя практически не имеет смысла
- **В применении к массивам** операции `new` и `delete` имеют следующий синтаксис (здесь скобки [] — элемент грамматики языка):

  `new` <*имя типа*> [<*размер массива*>]

  `delete` [] <*указатель*>

Встраиваемые функции

- **Встраиваемые (подставляемые, открыто подставляемые) функции** при компиляции внедряются непосредственно в код программы, что увеличивает объем объектного модуля, но повышает быстродействие кода
- Функция не может быть встраиваемой, если она
  - слишком велика для подстановки в объектный код;
  - является рекурсивной;
  - определена ниже по тексту программы точки обращения к ней;
  - вызывается в выражении два и более раза;
  - содержит цикл, переключатель или оператор перехода

  При невозможности подстановки функция считается статической.
- **Определение встраиваемой функции**

  `inline` <*тип результата*> <*идентификатор*>
  (`[`<*спецификация формальных параметров*>`]`)
  `{` `[`<*тело функции*>`]` `}`

Перегрузка функций. Аргументы по умолчанию. Неименованные параметры

- **Перегрузка функций** — определение ряда одноименных функций с разными по типам и/или количеству формальными (а значит, и фактическими) параметрами (иными словами — с разными **сигнатурами**). Распознавание перегруженных функций при вызове происходит по сигнатурам
- Задание **аргументов по умолчанию** расширяет стандартное ANSI-определение функций

  <тип результата> <идентификатор>
  (*[<спецификация формальных параметров>]*)
  { *[<тело функции>]* }

  так, что спецификация единичного формального параметра приобретает вид

  <имя типа> <идентификатор> = <инициализирующее выражение>
- Неиспользуемые в теле функции формальные параметры могут оставаться **неименованными**

Список литературы

- [КР92] Керниган Б., Ритчи Д. Язык программирования Си / Пер. с англ. — М.: Финансы и статистика, 1992. — 272 с.
- [КР06] Керниган Б., Ритчи Д. Язык программирования С / Пер. с англ. — М.: Вильямс, 2006. — 304 с.
- [Под03] Подбельский В.В. Язык Си++: Учеб. пособие. — 5-е изд. — М.: Финансы и статистика, 2003. – 560 с., ил.
- [Под04] Подбельский В.В., Фомин С.С. Программирование на языке Си. – 2-е доп. изд. – М., Финансы и статистика, 2004. – 600 с.