Уральский федеральный университет

СТРУКТУРЫ ДАННЫХ

Лектор
Спиричева Наталия Рахматулловна

Ст. преподаватель каф. ИТ
Ауд. Р-246

Структуры данных

Составитель курса лекций:
Спиричева Наталия Рахматулловна,
ст. преподаватель каф. Информационных технологий

Структуры данных

Целью лекции является приобретение студентами следующих компетенций :

- знать основные типы данных
- уметь правильно и оптимально их использовать при программировании на ЯВУ

ФУНДАМЕНТАЛЬНЫЕ ТИПЫ ДАННЫХ

Основные темы лекции:

- Числовые типы
- Битовые типы
- Логический тип
- Символьный тип
- Интервальный тип
- Указатели

Иерархия типов данных С

- char - символьный;
- int - целый;
- float - вещественный;
- double - вещественный двойной точности;
- void - не имеющий значения.

Типы данных С

Объект некоторого базового типа может быть модифицирован. С этой целью используются специальные ключевые слова, называемые модификаторами. В стандарте ANSI языка Си имеются следующие модификаторы типа:

- unsigned
- signed
- short
- long

Типы данных C

Тип	Размер в байтах (битах)	Интервал изменения
char	1 (8)	от -128 до 127
unsigned char	1 (8)	от 0 до 255
signed char	1 (8)	от -128 до 127
int	2 (16)	от -32768 до 32767
unsigned int	2 (16)	от 0 до 65535
signed int	2 (16)	от -32768 до 32767
short int	2 (16)	от -32768 до 32767
unsigned short int	2 (16)	от 0 до 65535
signed short int	2 (16)	от -32768 до 32767
long int	4 (32)	от -2147483648 до 2147483647
unsigned long int	4 (32)	от 0 до 4294967295
signed long int	4 (32)	от -2147483648 до 2147483647
float	4 (32)	от 3.4E-38 до 3.4E+38
double	8 (64)	от 1.7E-308 до 1.7E+308
long double	10 (80)	от 3.4E-4932 до 3.4E+4932

ФУНДАМЕНТАЛЬНЫЕ ТИПЫ ДАННЫХ

Простые структуры данных называют также базовыми структурами или фундаментальными типами данных. Они служат основой для построения более сложных структур. В языках программирования простые структуры описываются простыми (базовыми) типами. К простым типам относятся порядковые и вещественные типы.

ФУНДАМЕНТАЛЬНЫЕ ТИПЫ ДАННЫХ

Порядковые типы

Каждый из них имеет конечное число возможных значений. Эти значения можно определенным образом упорядочить (отсюда – название типов) и, следовательно, с каждым из них можно сопоставить некоторое целое число – порядковый номер значения. К порядковым типам относятся целые, логический, символьный, перечисляемый типы.

К порядковым типам кроме логического можно также применять функции:

- --X – возвращает предыдущее значение порядкового типа;
- ++X - возвращает следующее значение порядкового типа.

ФУНДАМЕНТАЛЬНЫЕ ТИПЫ ДАННЫХ

Вещественные типы

Вещественные типы, строго говоря, тоже имеют конечное число значений, которое определяется форматом внутреннего представления вещественного числа. Однако количество возможных значений вещественных типов настолько велико, что сопоставить с каждым из них целое число (его номер) не представляется возможным.

ФУНДАМЕНТАЛЬНЫЕ ТИПЫ ДАННЫХ

Числовые типы

Числовые типы

Целые типы

С помощью целых чисел может быть представлено количество объектов, являющихся дискретными по своей природе (т.е. счетное число объектов).

Числовые типы

Целые типы

ПРЕДСТАВЛЕНИЕ В ПАМЯТИ

Для представления чисел со знаком в ряде компьютеров был использован метод, называемый методом знака и значения. Обычно для знака отводится первый (или самый левый) бит двоичного числа, затем следует запись самого числа.

Например: +10 и -15 в двоичном виде можно представить так:

- +10 0 0001010
- - 15 1 0001111

- 0 - используется для представления знака плюс
- 1 - для минуса.

Числовые типы

Например, сложение чисел +6 и -7 на самом деле подразумевает операцию вычитания, а вычитание -6 из +7 операцию сложения. Для анализа знакового бита требуется особая схема, и, кроме того, при представлении числа в виде знака и величины необходимы отдельные устройства для сложения и вычитания, т.е. если положительное и отрицательные числа представлены в прямом коде, операции над кодами знаков выполняются раздельно. Поэтому представление чисел в виде знака и значения не нашло широкого применения.

В то же время при помощи обратного и дополнительного кодов, используемых для представления отрицательных чисел, операция вычитания (алгебраического сложения) сводится к операции простого арифметического сложения. При этом операция сложения распространяется и на разряды знаков, рассматриваемых как разряды целой части числа. Именно поэтому для представления целых чисел со знаком применяется дополнительный код.

Целочисленные типы C++

Тип	Количество битов	Диапазон значений
byte	8	От 0 до 255
sbyte	8	От -128 до 127
short	16	От -32768 до 32767
ushort	16	От 0 до 65535
int	32	От -2147483648 до 2147483647
uint	32	От 0 до 4294967295
long	64	От -9223372036854775808 до 9223372036854775807
ulong	64	От 0 до 18446774073709551615

Целочисленные типы C

Тип	Диапазон значений	Машинное представление
char	-128 . . . 127	8 бит, со знаком
short, int(ms_DOS)	-32768 . . . 32767	16 бит, со знаком
long, int(windows)	-2147483648..2147483647	32 бита, со знаком
unsigned char	0 . . . 255	8 бит, без знаком
unsigned short	0 . . . 65535	16 бит, без знаком
long long	-2^{63} . . . $2^{63}-1$	64 бита, со знаком

Числовые типы

Дополнительный код отрицательного числа формируется по следующим правилам:

1. модуль отрицательного числа записать в прямом коде, в неиспользуемые старшие биты записать нули;
2. сформировать обратный код числа, для этого нуль заменить единицей, а единицу заменить нулем;
3. к обратному коду числа прибавить единицу.

Например: для числа -33 в формате short

```
1000000000100001      прямой код .
0111111111011110      обратный код
+                    1
-------------------------
 1111111111011111     дополнительный код
```

Для положительных чисел прямой, обратный и дополнительный коды одинаковы.

Числовые типы

МАШИННОЕ ПРЕДСТАВЛЕНИЕ БЕЗЗНАКОВЫХ ТИПОВ

К беззнаковым типам в C++ относятся целые типы с приставкой unsigned (беззнаковый).

Формат машинного представления чисел типа unsigned char такое же, как и у char.

Например:

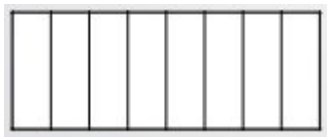
1). Машинное представление числа 45:

45=25+23+22+20 = 00101101

2). Машинное представление границ диапазона допустимых значений чисел 0 и 255:

0: 00000000; 255: 11111111.

тип unsigned char



Числовые типы

Формат машинного представления чисел типа unsigned short

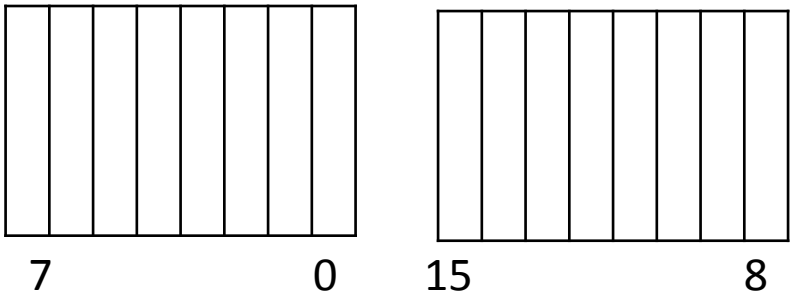
1). Машинное представление числа 258:

257=28+21 = 00000001 00000010.

2). Машинное представление границ:

0: 00000000 00000000; 65535: 11111111 11111111

тип unsigned short



Числовые типы

Для представления чисел со знаком определены следующие типы CHAR, SHORT, INT, LONG, LONG LONG. В приведенных типах числа хранятся в дополнительном коде.

машинное представление чисел в формате CHAR:

1). 0: 00000000;
2). +127: 01111111;
3). - 128: 10000000.

машинное представление чисел в формате sHORT

1). +32765: 11111101 0111111
2). - 32765: 00000011 10000000;
3). -47:11010001

Перевод чисел из одной системы счисления в другую

При переводе целого числа (целой части числа) из одной системы счисления в другую исходное число (или целую часть) надо разделить на основание системы счисления, в которую выполняется перевод. Деление выполнять, пока частное не станет меньше основания новой системы счисления. Результат перевода определяется остатками от деления: первый остаток дает младшую цифру результирующего числа, последнее частное от деления дает старшую цифру.

При переводе правильной дроби из одной системы счисления в другую систему счисления дробь следует умножать на основание системы счисления, в которую выполняется перевод. Полученная после первого умножения целая часть является старшим разрядом результирующего числа. Умножение вести до тех пор пока произведение станет равным нулю или будет получено

Перевод чисел из одной системы счисления в другую

Например

1). Перевести дробное число 0.243 из десятичной системы счисления в двоичную.

$\mathbf{0.243_{10}}$ $\mathbf{0.0011111_{2}}$.

Проверка:

0.0011111 = 0*2-1+0*2-2+1*2-3+1*2-4+1*2-5+1*2-6+1*2-7= 0,2421875

2). Перевести целое число 164 из десятичной системы счисления в двоичную систему.

$\mathbf{164_{10}}$ $\mathbf{10100100_{2}}$

Проверка:

10100100=1*27+0*26+1*25+0*24+0*23+1*22+0*21+0*20=128+32+4=164.

При переводе смешанных чисел целая и дробная части числа переводятся отдельно.

Вещественные типы

ПРЕДСТАВЛЕНИЕ ВЕЩЕСТВЕННЫХ ЧИСЕЛ В ПАМЯТИ.

В некоторых областях вычислений требуются очень большие или весьма малые действительные числа. Для получения большей точности применяют запись чисел с плавающей точкой. Запись числа в формате с плавающей точкой является весьма эффективным средством представления очень больших и весьма малых вещественных чисел при условии, что они содержат ограниченное число значащих цифр, и, следовательно, не все вещественные числа могут быть представлены в памяти. Обычно число используемых при вычислениях значащих цифр таково, что для большинства задач ошибки округления пренебрежимо

Вещественные типы

Формат для представления чисел с плавающей точкой содержит одно или два поля фиксированной длины для знаков. Количество позиций для значащих цифр различно в разных ЭВМ, но существует, тем не менее, общий формат. В соответствии с этой записью формат вещественного числа содержит в общем случае поля мантиссы, порядка и знаков мантиссы и порядка.

Знак числа	Порядок	Знак порядка	Мантисса

Вещественные типы

Однако чаще вместо порядка используется характеристика, получающаяся прибавлением к порядку такого смещения, чтобы характеристика была всегда положительной.

Знак числа	Характеристика	Мантисса

Вещественные типы

Введение характеристики избавляет от необходимости выделять один бит для знака порядка и упрощает выполнение операций сравнения (<,>,<=,>=) и арифметических операций над вещественными числами. Так, при сложении или вычитании чисел с плавающей точкой для того, чтобы выровнять операнды, требуется сдвиг влево или вправо мантиссы числа. Сдвиг можно осуществить с помощью единственного счетчика, в который сначала заносится положительное число, уменьшающееся затем до тех пор, пока не будет выполнено требуемое число сдвигов.

Таким образом, для представления вещественных чисел в памяти ЭВМ порядок p вещественного числа представляется в виде характеристики путем добавления смещения (старшего бита порядка):

- X = 2n-1 + k + p, где n - число бит, отведенных для характеристики,
- p - порядок числа,
- k - поправочный коэффициент фирмы IBM.

Вещественные типы

Следующим компонентом представляемого в машине числа с плавающей точкой является мантисса. Для увеличения количества значащих цифр в представлении числа и исключения переполнения при умножении мантиссу обычно подвергают нормализации. Нормализация означает, что мантисса (назовем ее F), кроме случая, когда F = 0, должна находиться в интервале R-1 <= F < 1.

Для двоичной системы счисления R = 2. Тогда в связи с тем, что 2-1 <= F < 1, ненулевая мантисса любого хранимого числа с плавающей точкой должна начинаться с двоичной единицы. В этом и заключается одно из достоинств двоичной формы представления числа с плавающей точкой. Поскольку процесс нормализации создает дробь, первый бит которой равен 1, в структуре некоторых машин эта единица учитывается, однако не записывается в мантиссу. Эту единицу часто называют скрытой единицей, а получающийся дополнительный бит используют для увеличения точности представления чисел

Вещественные типы

АЛГОРИТМ ФОРМИРОВАНИЯ МАШИННОГО ПРЕДСТАВЛЕНИЯ ВЕЩЕСТВЕННОГО ЧИСЛА В ПАМЯТИ ЭВМ:

1. Число представляется в двоичном коде.

2. Двоичное число нормализуется. При этом для чисел, больших единицы, плавающая точка переносится влево, определяя положительный порядок. Для чисел, меньших единицы, точка переносится вправо, определяя отрицательный порядок.

3. Определяется характеристика, с учетом типа вещественного числа.

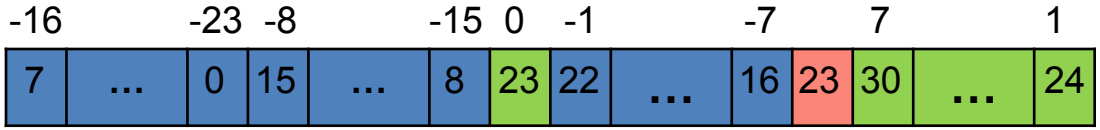
Вещественные типы

4. В отведенное в памяти поле в соответствии с типом числа записываются мантисса, характеристика и знак числа. При этом необходимо отметить следующее:

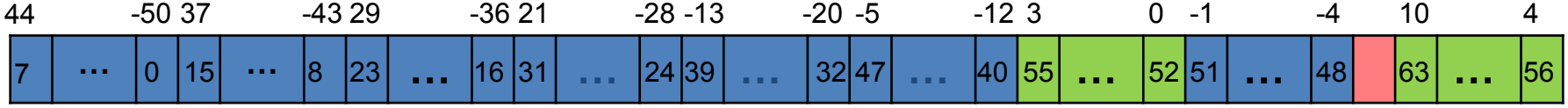
- для чисел типа float, double характеристика хранится в младшем байте памяти;
- знак числа находится всегда в старшем бите старшего байта;
- мантисса всегда хранится в прямом коде;
- целая часть мантиссы (для нормализованного числа всегда 1) для чисел типа float, double.

Вещественные типы

МАШИННОЕ ПРЕДСТАВЛЕНИЕ ДАННЫХ ТИПА FLOAT:



МАШИННОЕ ПРЕДСТАВЛЕНИЕ ДАННЫХ ТИПА DOUBLE:



- знаковый разряд
- характеристика
- нормализованная мантисса

Десятичные типы

Эти типы применяются для внутримашинного представления таких данных, которые в первую очередь должны храниться в вычислительной системе и выдаваться пользователю по требованию и лишь во вторую очередь - обрабатываться (служить операндами вычислительных операций)

Архитектура некоторых вычислительных систем (например, IBM System/390) предусматривает команды, работающие с десятичным представлением чисел, хотя эти команды и выполняются гораздо медленнее, чем команды двоичной арифметики. В других архитектурах операции с десятичными числами моделируются программно.

Десятичные типы

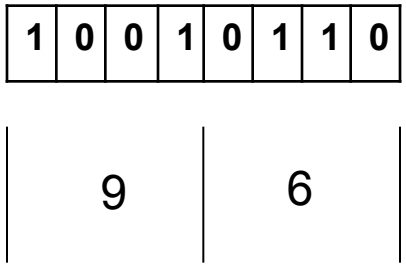
ДЕСЯТИЧНЫЙ ТИП С ФИКСИРОВАННОЙ ТОЧКОЙ.

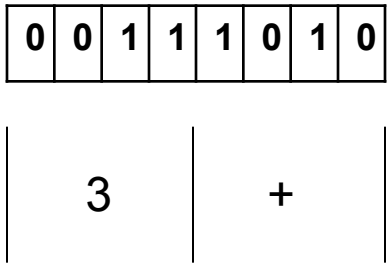
В языке PL/1 десятичный тип с фиксированной точкой описывается в программе, как:

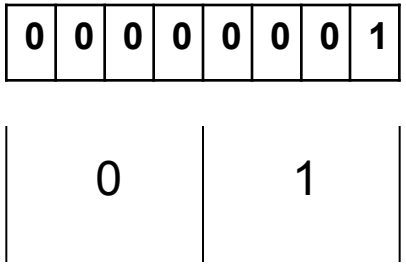
DECIMAL FIXED (m.d) или DECIMAL FIXED (m). Первое описание означает, что данное представляется в виде числа, состоящего из m десятичных цифр, из которых d цифр расположены после десятичной точки. Второе - целое число из m десятичных цифр. Следует подчеркнуть, что в любом случае число десятичных цифр в числе фиксировано. Внутримашинное представление целых чисел и чисел с дробной частью одинаково. Для последних положение десятичной точки запоминается компилятором и учитывается им при трансляции операций, в которых участвуют десятичные числа с фиксированной точкой.

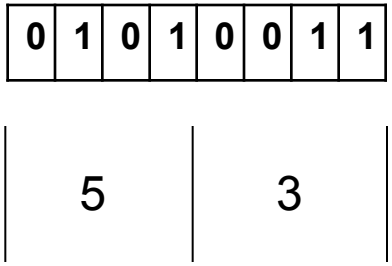
Десятичные типы

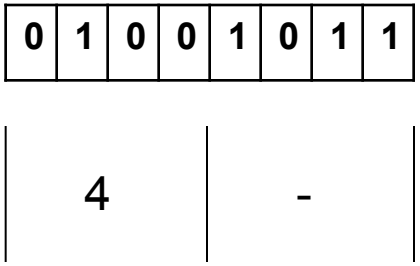
Примеры:











Десятичные типы

Каждая десятичная цифра числа занимает полбайта (4 двоичных разряда) и представляется в этом полубайте ее двоичным кодом. Еще полбайта занимает знак числа, который представляется двоичным кодом 1010 - знак "+" или 1011 - знак "-". Представление занимает целое число байт и при необходимости дополняется ведущим нулем.

Тип decimal C#

Возможно, наиболее интересным числовым типом данных в C# является тип decimal, предназначенный для использования в денежных вычислениях. В типе decimal для представления значений, находящихся в диапазоне от 1E-28 до 7.9E+28, используется 128 битов. Помним, что в обычных арифметических вычислениях, производимых над числами с плавающей точкой, неоднократные округления значений приводят к неточному результату. Тип данных decimal устраняет ошибки, возникающие при округлении, и может представлять числа с точностью до 28 десятичных разрядов (а в некоторых случаях и до 29 разрядов). Эта способность представлять десятичные значения без ошибок округления особенно полезна, когда рассчитываются финансы.

Операции над числовыми типами

Над числовыми типами, как и над всеми другими, возможны прежде всего четыре основных операции: *создание, уничтожение, выбор, обновление*. Специфические операции над числовыми типами - хорошо известные всем арифметические операции: сложение, вычитание, умножение, деление. Операция возведения в степень в некоторых языках также является базовой и обозначается специальным символом или комбинацией символов (^ - в BASIC, ** - в PL/1), в других - выполняется встроенными функциями (pow в C). В языке Pascal возведение в степень выполняется с помощью функций Exp и Ln.

Операции над числовыми типами

Обратим внимание на то, что операция деления по-разному выполняется для целых и вещественных чисел. При делении целых чисел дробная часть результата отбрасывается, как бы близка к 1 она ни была. В связи с этим в языке PASCAL имеются даже разные обозначения для деления вещественных и целых чисел - операции "/" и "div" соответственно. В других языках оба вида деления обозначаются одинаково, а тип деления определяется типом операндов. Для целых операндов возможна еще одна операция - остаток от деления - ("mod" - в PASCAL, "%" - в C). Еще одна группа операций над числовыми типами - операции сравнения: "равно", "не равно", "больше", "меньше" и т.п. Существенно, что хотя операндами этих операций являются данные числовых типов, результат их имеет логический тип - "истина" или "ложь".

Говоря об операциях сравнения, следует обратить внимание на особенность выполнения сравнений на равенство/неравенство вещественных чисел. Поскольку эти числа представляются в памяти с некоторой (не абсолютной)

Фундаментальные типы данных

Битовые типы

Битовые типы

ПРЕДСТАВЛЕНИЕ БИТОВЫХ ТИПОВ

В ряде задач может потребоваться работа с отдельными двоичными разрядами данных. Чаще всего такие задачи возникают в системном программировании, когда, например, отдельный разряд связан с состоянием отдельного аппаратного переключателя или отдельной шины передачи данных и т.п. Данные такого типа представляются в виде набора битов, упакованных в байты или слова и не связанных друг с другом. Операции над такими данными обеспечивают доступ к выбранному биту данного. В языке PASCAL роль битовых типов выполняют беззнаковые целые типы byte и word. Над этими типами помимо операций, характерных для числовых типов, допускаются и побитовые операции. Аналогичным образом роль битовых типов играют беззнаковые целые и в языке C.

В языке PL/1 существует специальный тип данных - строка битов, объявляемый в программе, как: BIT(n).

Данные этого типа представляют собой последовательность бит длиною n. Строка битов занимает целое число байт в памяти и при необходимости дополняется справа нулями.

Битовые типы

ОПЕРАЦИИ НАД БИТОВЫМИ ТИПАМИ.

Над битовыми типами возможны три группы специфических операций:

- операции булевой алгебры
- операции сдвигов
- операции сравнения

Операции булевой алгебры - НЕ (!), ИЛИ (|), И (&), исключающее ИЛИ (^).

Эти операции и по названию, и по смыслу похожи на операции над логическими операндами, но отличие в их применении к битовым операндам состоит в том, что операции выполняются над отдельными разрядами операндов.

Битовые типы

Примеры выполнения побитовых логических операций:

а). x= 01101100
not x = 10010011
x & y = 01001100

б). x = 01101100
y = 11001110
x | y = 11101110

в). x = 01101100
y = 11001110

г). x = 01101100
y = 11001110
x ^ y = 10100010

Битовые типы

Операции сдвигов выполняют смещение двоичного кода на заданное количество разрядов влево или вправо. Из трех возможных типов сдвига (арифметический, логический, циклический) в языках программирования обычно реализуется только логический (например, операциями << и >> в языке C).

В операциях сравнения битовые данные интерпретируются как целые без знака, и сравнение выполняется как сравнение целых чисел. Битовые строки в языке PL/1 - более общий тип данных, к которому применимы также операции над строковыми данными.

ПРОСТЫЕ СТРУКТУРЫ ДАННЫХ

Тема 3:
Логический тип

Логический тип

Значениями логического типа BOOL может быть одна из предварительно объявленных констант false (ложь) или true (истина).

Данные логического типа занимают один байт памяти. При этом значению false соответствует нулевое значение байта, а значению true соответствует любое ненулевое значение байта.
Например:

- *false* всегда в машинном представлении: 00000000;
- *true* может выглядеть таким образом: 00000001 или 00010001 или 10000000.

Логический тип

Над логическими типами возможны операции булевой алгебры - НЕ (!), ИЛИ (||), И (&&), исключающее ИЛИ (^). В этих операциях операнды логического типа рассматриваются как единое целое - вне зависимости от битового состава их внутреннего представления.

Кроме того, следует помнить, что результаты логического типа получаются при сравнении данных любых типов.

ПРОСТЫЕ СТРУКТУРЫ ДАННЫХ

Тема 4:
Символьный тип

Символьный тип

Значением символьного типа *char* являются символы из некоторого предопределенного множества. В большинстве современных персональных ЭВМ этим множеством является ASCII (American Standard Code for Information Intechange - американский стандартный код для обмена информацией). Это множество состоит из 256 разных символов, упорядоченных определенным образом, и содержит символы заглавных и строчных букв, цифр и других символов, включая специальные управляющие символы. Допускаются некоторые отклонения от стандарта ASCII, в частности, при наличии соответствующей системной поддержки это множество может содержать буквы русского алфавита. Порядковые номера (кодировку) можно узнать в соответствующих разделах технических описаний.

Значение символьного типа char занимает в памяти 1 байт. Код от 0 до 255 в этом байте задает один из 256 возможных символов ASCII таблицы. Например: символ "1" имеет ASCII код 49, следовательно, машинное представление будет выглядеть следующим образом: 00110001.

Символьный тип

ASCII, однако, не является единственно возможным множеством. Другим достаточно широко используемым множеством является код EBCDIC (Extended Binary Coded Decimal Interchange Code - расширенный двоично-кодированный десятичный код обмена), применяемый в системах IBM средней и большой мощности. В EBCDIC код символа также занимает один байт, но с иной кодировкой, чем в ASCII.

И ASCII, и EBCDIC включают в себя буквенные символы только латинского алфавита. Символы национальных алфавитов занимают "свободные места" в таблицах кодов, и, таким образом, одна таблица может поддерживать только один национальный алфавит. Этот недостаток преодолен во множестве UNICODE, которое находит все большее распространение прежде всего в UNIX-ориентированных системах. В UNICODE каждый символ кодируется *двумя байтами*, что обеспечивает более 64 тыс. (216) возможных кодовых комбинаций и дает возможность иметь единую таблицу кодов, включающую в себя все национальные алфавиты. UNICODE, безусловно, является перспективным, однако, повсеместный переход к двухбайтным кодам символов может вызвать необходимость переделки значительной части существующего программного обеспечения.

Символьный тип

Специфические операции над символьными типами - только операции сравнения. При сравнении коды символов рассматриваются как целые числа без знака. Кодовые таблицы строятся так, что результаты сравнения подчиняются лексикографическим правилам: символы, занимающие в алфавите места с меньшими номерами, имеют меньшие коды, чем символы, занимающие места с большими номерами. В основном символьный тип данных используется как базовый для построения интегрированного типа "строка символов".

ПРОСТЫЕ СТРУКТУРЫ ДАННЫХ

Тема 5:
Перечислимый тип

Перечислимый тип

ЛОГИЧЕСКАЯ СТРУКТУРА

Перечислимый тип представляет собой упорядоченный тип данных, определяемый программистом, т.е. программист перечисляет все значения, которые может принимать переменная этого типа. Значения являются неповторяющимися в пределах программы идентификаторами, количество которых не может быть больше 2 147 483 647.

Например:

- *enum color{red,blue,green};*
- *enum work_day{mo,tu,we,th,fr};*
- *enum winter_day{december,january,february};*

Перечислимый тип

МАШИННОЕ ПРЕДСТАВЛЕНИЕ

Для переменной перечислимого типа в C++ выделяется 4-е байта, в который записывается порядковый номер присваиваемого значения. Порядковый номер определяется из описанного типа, причём нумерация начинается с 0. Имена из списка перечислимого типа являются константами

Например:

- *color B,C;*
- *B:=blue; (* B=1 *)*
- *C:=green; (* C=2 *)*
- *printf("B- %d C- %d", B, C); - выдаст на экран: B- 1 C- 2*

После выполнения данного фрагмента программы на экран будут выданы цифры 1 и 2. Содержимое памяти для переменных B И C при этом следующее: B - 00000001;

C – 00000010

Перечислимый тип

ОПЕРАЦИИ

На физическом уровне над переменными перечислимого типа определены операции создания, уничтожения, выбора, обновления. При этом выполняется определение порядкового номера идентификатора по его значению и, наоборот, по номеру идентификатора определяется его значение.

На логическом уровне переменные перечислимого типа могут быть использованы только в выражениях булевского типа и в операциях сравнения; при этом сравниваются порядковые номера значений.

ПРОСТЫЕ СТРУКТУРЫ ДАННЫХ

Тема 6:
Интервальный тип языка PASCAL

Интервальный тип языка PASCAL

ЛОГИЧЕСКАЯ СТРУКТУРА

Один из способов образования новых типов из уже существующих - ограничение допустимого диапазона значений некоторого стандартного скалярного типа или рамок описанного перечислимого типа. Это ограничение определяется заданием минимального и максимального значений диапазона. При этом изменяется диапазон допустимых значений по отношению к базовому типу, но представление в памяти полностью соответствует базовому типу

Интервальный тип языка PASCAL

МАШИННОЕ ПРЕДСТАВЛЕНИЕ

Данные интервального типа могут храниться в зависимости от верхней и нижней границ интервала независимо от входящего в этот предел количества значений в виде. Для данных интервального типа требуется память размером один, два или четыре байта

Например.

- *var A: 220..250; (* Занимает 1 байт *)*
- *B: 2221..2226; (* Занимает 2 байта *)*
- *C: 'A'..'K'; (* Занимает 1 байт *)*
- *begin A:=240; C:='C'; B:=2222; end.*

После выполнения данной программы содержимое памяти будет следующим: A - 11110000;

C - 01000011;

B - 10101110 00001000.

Интервальный тип языка PASCAL

ОПЕРАЦИИ

На физическом уровне над переменными интервального типа определены операции создания, уничтожения, выбора, обновления. Дополнительные операции определены базовым типом элементов интервального типа.

А). Интервальный тип от символьного: определение кода символа и, наоборот, символа по его коду.

Пусть задана переменная типа tz:'d'..'h'. Данной переменной присвоено значение 'e'. Байт памяти, отведенный под эту переменную, будет хранить ASCII-код буквы 'e', т.е. 01100101 (в 10-м представлении 101).

Б). Интервальный тип от перечислимого: определение порядкового номера идентификатора по его значению и, наоборот, по номеру идентификатора - его значение.

На логическом уровне все операции, разрешенные для данных базового типа, возможны и для данных соответствующих интервальных типов.

ПРОСТЫЕ СТРУКТУРЫ ДАННЫХ

Тема 7:
Указатели

Указатели

Оперативная память компьютера представляет собой совокупность элементарных ячеек для хранения информации – байтов, каждый их которых имеет собственный номер. Эти номера называются *адресами,* они позволяют обращаться к любому байту памяти.

Указатель – это переменная, которая в качестве своего значения содержит адрес юайта памяти.

Тип указателя представляет собой адрес ячейки памяти (в подавляющем большинстве современных вычислительных систем размер ячейки - минимальной адресуемой единицы памяти - составляет один байт).

Указатели

При решении прикладных задач с использованием языков высокого уровня наиболее частые случаи, когда программисту могут понадобиться указатели, следующие:

1) при необходимости представить одну и ту же область памяти, а следовательно, одни и те же физические данные, как данные разной логической структуры. В этом случае в программе вводятся два или более указателей, которые содержат адрес одной и той же области памяти, но имеют разный тип. Обращаясь к этой области памяти по тому или иному указателю, программист обрабатывает ее содержимое как данные того или иного типа;

2) при работе с динамическими структурами данных, что более важно. Память под такие структуры выделяется в ходе выполнения программы, стандартные функции выделения памяти возвращают адрес выделенной области памяти - указатель на нее. К содержимому динамически выделенной области памяти программист может обращаться только через такой указатель.

Указатели

Представление указателей в языках программирования

В программе на языке высокого уровня указатели могут быть типизированными и нетипизированными.

При объявлении типизированного указателя определяется и тип объекта в памяти, адресуемого этим указателем.

Так, например, объявления в языке PASCAL:

- *Var ipt : ^integer; cpt : ^char;*

  или в языке C:
- *int* ipt; char* cpt*;

означают, что переменная ipt представляет собой адрес области памяти, в которой хранится целое число, а cpt - адрес области памяти, в которой хранится символ.

Указатели

Хотя физическая структура адреса не зависит от типа и значения данных, хранящихся по этому адресу, компилятор считает указатели ipt и cpt имеющими разный тип, и в Pascal оператор

cpt := ipt;

будет расценен компилятором как ошибочный (компилятор C для аналогичного оператора присваивания, в случае явного приведения типов, ограничится предупреждением).

Таким образом, когда речь идет об указателях типизированных, правильнее говорить не о едином типе данных "указатель", а о целом семействе типов: "указатель на целое", "указатель на символ" и т.д. Могут быть указатели и на более сложные, интегрированные структуры данных, и указатели на указатели.

Указатели

Нетипизированный указатель, тип pointer в Pascal или void * в C, служит для представления адреса, по которому содержатся данные неизвестного типа. С их помощью удобно динамически размещать данные, структура и тип которых меняются в ходе работы программы. Работа с нетипизированными указателями существенно ограничена, они могут использоваться только для сохранения адреса, обращение по адресу, задаваемому нетипизированным указателем, невозможно.

Указатели

Операции над указателями

Основными операциями, в которых участвуют указатели, являются присваивание, получение адреса и разыменование.

Присваивание является двухместной операцией, оба операнда которой - указатели. Как и для других типов, операция присваивания копирует значение одного указателя в другой, в результате оба указателя будут содержать один и тот же адрес памяти. Если оба указателя, участвующие в операции присваивания, типизированные, то оба они должны указывать на объекты одного и того же типа.

Операция получения адреса - одноместная, ее операнд может иметь любой тип, результатом является типизированный (в соответствии с типом операнда) указатель, содержащий адрес объекта-операнда.

Операция разыменования - одноместная, ее операндом является типизированный (обязательно!) указатель, результат - данные, выбранные из памяти по адресу, заданному операндом. Тип результата определяется типом указателя-операнда.

Указатели

В языке C доступны также операции адресной арифметики

К указателю можно прибавить целое число или вычесть из него целое число. Поскольку память имеет линейную структуру, прибавление к адресу числа даст адрес памяти, смещенный на это число байт (или других единиц измерения) относительно исходного адреса. Результат операций "указатель + целое", "указатель - целое" имеет тип "указатель". Можно вычесть один указатель из другого (оба указателя-операнда при этом должны иметь одинаковый тип). Результат такого вычитания будет иметь тип целого числа со знаком. Его значение показывает, на сколько байт (или других единиц измерения) один адрес отстоит от другого в памяти.

Указатели

Операции адресной арифметики выполняются только над типизированными указателями. Единицей измерения в адресной арифметике является размер объекта, который указателем адресуется. Так, если переменная ipt определена как указатель на целое число (int *ipt), то выражение ipt+1 даст адрес, больший не на 1, а на количество байт в целом числе (в MS DOS - 2). Вычитание указателей также дает в результате не количество байт, а количество объектов данного типа, помещающихся в памяти между двумя адресами. Это справедливо как для указателей на простые типы, так и для указателей на сложные объекты, размеры которых составляют десятки, сотни и более байт.

КОНТРОЛЬНЫЕ ВОПРОСЫ

- Каковы особенности порядковых типов?
- Алгоритм перевода десятичного числа в двоичное?
- Как представляются целые отрицательные числа в памяти компьютера?
- Каковы особенности представления вещественных чисел?
- Какие основные операции выполняются над фундаментальными типами данных?

Спасибо за внимание!