Федеральное государственное автономное
образовательное учреждение
высшего профессионального образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра информатики

КУРСОВОЙ ПРОЕКТ

Система мгновенного обмена сообщениями для локальных компьютерных сетей

Руководитель _____ _____ Кузнецов.А.С.

подпись, дата должность, ученая степень инициалы, фамилия

Студент КИ11-17Б 031102909 _____ Прозоров Н.Р.

номер группы номер зачетной книжки подпись, дата инициалы, фамилия

Студент КИ11-17Б 031100518 _____ Цирюльников А.М.

номер группы номер зачетной книжки подпись, дата инициалы, фамилия

Студент КИ11-17Б 031100044 _____ Шатров С.Н.

номер группы номер зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2015

СОДЕРЖАНИЕ

1. Описание языка

1.1. Обзор языка

Язык «Крабик (V)_ii_(V)» разрабатывается, как язык с Си-подобным синтаксисом, который позволит детям школьного возраста с легкостью овладеть основными методами программирования, основам логики и основам структур данных. Данный язык предназначен для разработки простых программ. Обеспечивает абстракцию от низкоуровневого инструментария, не позволяет ”выстрелить в ногу”.

«Крабик (V)_ii_(V)» является языком высокого уровня с блочной структурой, включающий арифметические выражения и подвыражения в скобках с операциями сложения, вычитания, умножения, деления, унарными минусом и плюсом и другие; логические выражения и подвыражения в скобках (логические И, ИЛИ, НЕ и другие):

- Выражения, конструкции, которым может быть поставлено в соответствие некоторое значение
- Присваивания в виде операций либо операторов
- Условия с необязательной частью else
- Циклы с параметром, с предусловие и постусловием
- Множественный выбор (switch-case)
- Безусловные переходы (goto, break и continue)
- Операции сравнения (>, =, <, !=, <=, =>)
- Функции и/или процедуры с операторами вызова и возврата из них (return)
- Простые средства ввода, вывода в виде операторов или встроенных функций
- Минимальные средства комментирования кода

Язык будет поддерживать данные встроенных (базовых) и конструируемых (пользовательских) типов. Базовые типы – целые и вещественные числа, литерные, логические, строковые. Количество базовых типов – не менее трех. Две или более разновидности пользовательских типов – классы/объекты, структуры, объединения, кортежи и другие со средствами описания структуры типа и операторами обращения к составным частям объектов пользовательских типов.

Язык будет поддерживать операторы объявления программных объектов, включая скаляры и массивы встроенных и пользовательских типов.

Области видимости переменных – блок, внутри которого они объявлены.

Встроенные типы: целочисленные Integer, с плавающей точкой Float, логический Boolean, строковый String.

Особенностью данного языка является некоторая ограниченность разработчика. К примеру, в языке не предусмотрены операции непосредственного доступа к памяти и упрощённая грамматика (относительно языка C).

1.2. Элементы языка

Набор символов “Крабик (V)_ii_(V)” содержит прописные и строчные буквы латинского алфавита, 10 десятичных цифр арабской системы исчисления и символ подчеркивания (_).

- Прописные английские буквы:
- A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
- Строчные английские буквы:
- a b c d e f g h i j k l m n o p q r s t u v w x y z
- Десятичные цифры:
- 0 1 2 3 4 5 6 7 8 9
- Символ подчеркивания (_)

Используются для формирования ключевых слов, констант и идентификаторов.

1.2.1. Разделительные символы

Пробел, смена строки, возврат каретки и символ новой строки. Эти символы отделяют задаваемые пользователем элементы (например, константы и идентификаторы, от других элементов программы).

Компилятор языка “Крабик (V)_ii_(V)” игнорирует разделительные символы, если они не служат для целей разделения или не являются компонентами символьных констант или строковых литералов. Компилятор рассматривает комментарии, как разделительные символы.

1.2.2. Знаки пунктуации и специальные символы

Знаки пунктуации и специальные символы из набора символов языка “Крабик (V)_ii_(V)” . Список знаков пунктуации и специальных символов из набора символов “Крабик (V)_ii_(V)” :

,	запятая
.	точка
;	точка с запятой
:	двоеточие
?	знак вопроса
'	одинарная цитатная скобка
"	двойная цитатная скобка
(	левая круглая скобка
)	правая круглая скобка
[	левая прямоугольная скобка
]	правая прямоугольная скобка
{	левая фигурная скобка
}	правая фигурная скобка
<	левая угловая скобка
>	правая угловая скобка
!	восклицательный знак
|	вертикальная черта

/ знак деления

\ знак обратного деления

~ тильда

+ плюс

номер

% процент

& амперсанд

^ крышечка

* звездочка

- минус

= равно

1.2.3. Управляющие последовательности

Строки и символьные константы могут содержать "управляющие последовательности". Управляющие последовательности это комбинации символов, состоящие из разделительного символа и неграфических символов. Управляющая последовательность состоит из знака обратного деления (\) за которым следует буква или комбинация цифр.

Список управляющих последовательностей:

\n новая строка

\t горизонтальная табуляция

\v вертикальная табуляция

\b забой

\r возврат каретки

\f смена листа

\a зуммер

\' одинарная цитатная скобка

\" двойная цитатная скобка

\\ обратное деление

\ddd восьмеричное значение символа ASCII

\xddd шестнадцатеричное значение символа ASCII

1.2.4. Операторы

"Операторы" это символы (состоящие из одного символа или комбинации символов), которые задают манипуляции над значениями. Каждый символ интерпретируется как отдельный элемент, называемый "лексемой".

1.2.5. Константы

Константа это число, символ или строка символов, которая в программе используется, как значение. Значение константы нельзя изменить.

В языке “Крабик (V)_ii_(V)” есть четыре вида констант: целые, с плавающей точкой, символьные и строковые литералы, логические.

1.2.6. Целые константы

Синтаксис: цифры

0цифры

0хцифры

0Хцифры

· "Десятичная константа" имеет форму одной или нескольких цифр (от 0 до 9), первая из которых не ноль.

· "Восьмеричная константа" имеет форму 0цифры, где цифры это одно или нескольких восьмеричных цифр (от 0 до 7), первая из которых ноль.

· "Шестнадцатеричная константа" имеет форму 0хцифры или 0Хцифры, где цифры это одно или нескольких шестнадцатеричных цифр (от 0 до 9 и прописные или строчные буквы от a до f), указание 0x или 0X обязательно.

Целые константы всегда имеют положительные значения. Если нужно использовать отрицательное значение, то поместите знак минус (-) перед константой для формирования выражения с отрицательным значением. Каждой целой константе на основании ее значения присваивается соответствующий тип. Тип константы определяет, какие преобразования будут выполнены при использовании константы в выражении или использовании знака минус (-).

Десятичные константы рассматриваются как величины со знаками и имеют тип int.

Восьмеричные и шестнадцатеричные имеют тип int.

1.2.7. Константы с плавающей точкой

Синтаксис: [цифры][.цифры][E|e-[+]цифры]

Константы с плавающей точкой всегда имеют положительные значения. Если нужно использовать отрицательное значение, то поместите знак минус (-) перед константой для формирования выражения с отрицательным значением. В данном случае знак минус интерпретируется как арифметический оператор.

Все константы с плавающей точкой имеют тип float.

Строковые литералы

Синтаксис: "символы”

"This is a crab (V)_ii_(V)"

Строковые литералы имеют тип String.

1.2.8. Идентификаторы

Синтаксис: буква|_[буква|цифра|_]...

Идентификатор создается его заданием в объявлении переменной, типа или функции. После этого его можно использовать в операторах программы для ссылки на связанный с ним элемент. Хотя метки и являются специальным видом идентификаторов и имеют собственный класс, их создание аналогично созданию идентификаторов для переменных и функций

Компилятор “Крабик (V)_ii_(V)” рассматривает прописные и строчные буквы как разные символы.

Идентификаторы не могут иметь произношение и написание, совпадающее с ключевыми словами языка. Ключевые слова описаны в разделе "Ключевые слова".

Приведем ряд примеров для идентификаторов:

Qwe

Crab123

1.2.9. Ключевые слова

"Ключевые слова" это заранее определенные идентификаторы, которые имеют для компилятора “Крабик (V)_ii_(V)” специальное значение. Их можно использовать только так, как они определены. Имя элемента программы не может совпадать по произношению и написанию с ключевым словом.

В языке “Крабик (V)_ii_(V)”имеются следующие ключевые слова:

int struct break else

switch return union

float continue for while

goto do if

1.2.10. Комментарии

Синтаксис: /* символы */

Комментарии могут появляться везде, где допустимо появление разделительных символов. Комментарий рассматривается компилятором как единичный разделительный символ, поэтому нельзя включать комментарии с лексемами. Однако, из-за того, что компилятор игнорирует символы комментария, в его текст можно включать ключевые слова без появления ошибок.

1.3. Структура программы

1.3.1. Исходная программа

Исходная программа языка "Крабик" это набор любого числа объявлений и функций. Все переменные в языке должны быть использованы только после объявления.

"Объявления функций" (или "прототипы") устанавливают имя функции, ее возвращаемый тип и, возможно, ее формальные параметры. В определении функции присутствуют те же элементы, что и в ее прототипе, плюс тело функции.

Объявления функций и переменных могут появляться как внутри так и вне определения функции. Любое объявление внутри определения функции считается "внутренним" (локальным). Объявление вне всех определений функций считается "внешним" (глобальным).

Определения переменных, как и их объявления, могут происходить как на внутреннем, так и на внешнем уровне. Определения функций всегда происходят на внешнем уровне.

1.3.2. Выполнение функций и программ

Каждая программа должна иметь главную функцию, которая должна иметь им main. Функция main является стартовой точкой выполнения программы. Она обычно управляет выполнением программы, вызывая в нужный момент другие функции. Программа обычно заканчивает свое выполнение в конце функции main, хотя ее выполнение может быть прервано и в других точках программы в зависимости от условий ее выполнения.

Исходная программа обычно состоит из нескольких функций, каждая из которых выполняет одну или несколько конкретных задач. Функция main может вызывать эти функции для выполнения соответствующих им задач. Когда main вызывает другую функцию, она передает выполнение этой функции и ее выполнение начинается с выполнения первого оператора функции. Функция возвращает управление после выполнение оператора return или после достижения конца функции.

1.3.3. Время и сфера действия

Время жизни и сфера действия переменных нераздельно связана с понятием "блок". Под блоком будем понимать последовательность определений и операторов, которая заключена в фигурные скобки. Блок может содержать в себе другие блоки за тем исключением, что блок не может содержать в себе другие блоки.

1.3.4. Время действия

"Время действия" это период выполнения программы, в котором существует переменная или функция. Все функции программы существуют на всем протяжении выполнения программы.

Время действия переменной может быть внутренним (локальным) или внешним (глобальным). Элемент с локальным временем действия ("локальный элемент") хранит и определяет значение только в блоке, в котором он был определен или объявлен. Локальному элементу всегда выделяется новое место в памяти каждый раз, когда программа входит в

блок, и хранимая величина теряется, когда программа выходит из блока. Если время действия переменной глобально (глобальный элемент), то он хранит и определяет значение на всем протяжении выполнения программы.

Следующие правила определяют, имеет переменная глобальное или локальное время действия:

- Переменные, которые объявлены на внутреннем уровне (в блоке) обычно имеют локальное время действия.

- Переменные, которые объявлены на внешнем уровне (вне всех блоков программы) всегда имеют глобальное время действия.

1.3.5. Сфера действия

"Сфера действия" элемента определяет ту часть программы, в которой на него можно сослаться по имени. Элемент доступен только в своей сфере действия, которая может быть ограничена (в порядке усиления ограничений) файлом, функцией, блоком или прототипом функции, в котором она появляется.

В случае меток имя метки имеет сферой своего действия функцию.

1.4. Объявления

Язык «Крабик» включает в себя пять типов данных:

*string - строковые значения

*float - числа с плавающей точкой

*integer - целочисленные значения в диапозоне от -32768 до 32,768

*bool - логические переменные

*func - для объявления функции, которая не возвращает значения

Объявление какой-либо переменной имеет следующий вид: [тип] [имя переменной] ([знак присваивания] [выражение])*. Пользователь может создавать свои собственные типы данных, объявление которых основано на уже имеющихся типах данных. Можно объявлять массивы, структуры данных и функции.

Объявление массива выглядит следующим образом: {тип данных} {имя массива}[(кол-во эллементов)*]([знак присваивания] { [выражение] })*.

"Объявление структуры" задает имя структурной переменной и последовательность значений переменной (называемых "компонентами" структуры), которые могут иметь различные типы. Переменная этого типа структуры содержит определенные этим типом последовательности. Имеет вид: struct [имя структуры] {[тело структуры]}

1.5. Выражения

1.5.1. Операнды

Операндами являются константы, идентификаторы, строки, вызовы функций, выражения индексов, выражения выбора компонент или более сложные выражения, которые формируются комбинированием операндов с операторами или заключением операндов в скобки.

Каждый операнд имеет тип. Далее будет рассматривается тип значения каждого вида, представляемый операндом. Операнд может быть "приведен" (или временно преобразован) из своего первоначального типа в другой тип, с помощью операции "приведения типа". Выражения приведения типа также формируют операнд выражения.

1.5.2. Константы

Постоянный операнд имеет то значение и тип, которое совпадает со значением, которое он представляет. Целочисленная константа имеет тип integer, вещественное число - float, true/false - boolean, а строка - string.

1.5.3. Идентификаторы

"Идентификатор" задает имя переменной или функции. Каждый идентификатор имеет тип, который устанавливается при его объявлении.

1.5.4. Строки

"Строковый литерал" это символ или последовательность символов, заключенная в двойные цитатные скобки.

1.5.5. Вызовы функций

Синтаксис: выражение([список-выражений])

"Вызов функции" состоит из выражения, за которым стоит необязательный элемент - список выражений в скобках, где выражение задает ее идентификатор и список выражений (разделенных запятыми), значения которых ("действительные аргументы") передаются в функцию. Аргумент "список выражений" может быть пустым.

К каждому выражению из списка выражений можно добавить &, это будет указывать на то, что требуется передать значение «по ссылке».

Выражение вызова функции имеет значение и тип возвращаемого функцией значения.

1.5.6. Доступ к массивам

Синтаксис: выражение1 [выражение2]

Выражение1 представляем собой имя массива, а выражение2 - номер элемента массива. Нумерация элементов массива начинается с нуля.

1.5.7. Выражения выбора компоненты

Синтаксис: выражение.идентификатор

"Выражение выбора компоненты" указывает компоненту структуры и объединения. Такие выражения имеют значение и тип выбранный компоненты.

1.5.8. Выражения с операторами

Выражения с операторами могут быть "унарными" и "бинарными". Унарное выражение состоит либо из унарного оператора, который применяется к операнду, за которым следует выражение. Это выражение может быть либо именем переменной

унарный-оператор операнд

Бинарные выражения состоят из двух операндов, объединенных бинарным оператором:

операнд бинарный-оператор операнд

1.5.9. Выражения в скобках

Можно заключать любой операнд в скобки без изменения типа или значения выражения

(10 + 5) / 5

Результатом (10+5)/5 будет 3. Без скобок, 10+5/5 даст своим результатом 11.

1.5.10. Операторы

Операторы языка С обрабатывают один операнд (унарные операторы), два операнда (бинарные операторы) или три операнда (тернарный оператор). Операторы присвоения включают и унарные и бинарные операторы. Операторы присвоения рассматриваются отдельно.

Унарные операторы ставятся до их операндов и ассоциируются справа налево. В языке С имеются следующие унарные операторы:

- ~ ! отрицание и дополнение

++ инкремент

-- декремент

Бинарные операторы ассоциируются слева направо. В языке имеются следующие бинарные операторы:

* / % мультипликативные

+ - аддитивные

< > <= >= == != отношения

& | ^ битовые

&& || логические

1.5.11. Приоритет и порядок проведения операций

Приоритет и ассоциативность выполнения операторов языка влияет на группирование и вычисление операндов в выражениях. Приоритет выполнения операторов имеет смысл только в том случае, если имеются другие операторы с большим или меньшим приоритетом. Сначала всегда вычисляются выражения с большим приоритетом операторов.

В Таблице 1 показаны приоритеты и ассоциативность операторов языка. Они приводятся в списке в порядке убывания их приоритета. Если на

одной строке указано несколько операторов, то они имеют равный приоритет и вычисляются по правилам их ассоциативности.

Таблица 1

Символ	Тип операции	Ассоциативность
() [] .	выражения	слева направо
- ~ !	унарные	справа налево
++ --	унарные	справа налево
* / %	мультипликативные	слева направо
+ -	аддитивные	слева направо
< > <= >=	отношения (неравенство)	слева направо
== !=	отношения (равенство)	слева направо
&	битовое-И	слева направо
^	битовое включающее-ИЛИ	слева направо
|	битовое исключающее-ИЛИ	слева направо
&&	логическое-И	слева направо
||	логическое-ИЛИ	слева направо

1.6. Операторы

Операторы программы на языке "Крабик" управляют процессом ее выполнения. В языке имеется ряд операторов, с помощью которых можно выполнять циклы, указывать другие операторы для выполнения и передавать управление на другой участок программы.

В операторах языка "Крабик" появляются следующие ключевые слова:

- if [else]*

Синтаксис: if (выражение)

оператор 1

[else

оператор2]

Выполнение:

Тело оператора if выполняется выборочно, в зависимости от значения выражения, по следующей схеме:

1) Вычисляется значение выражения.
2) Если значение выражения "истина" (не ноль), то выполняется оператор1.
3) Если значение выражения "ложь", то выполняется оператор2.
4) Если значение выражения "ложь" и не задана статья else, то оператор1 игнорируется.

Управление передается от оператора if на следующий оператор программы.

- for

Синтаксис: for([начальное-выражение];

[условное-выражение];

[выражение-цикла])

оператор

Тело оператора for может выполниться несколько раз, а может не выполниться ни разу, пока значением необязательного условного-выражения не станет "ложь". Можно использовать необязательные начальное-выражение и выражение-цикла для инициализации и смены значений при выполнении операторов for.

Выполнение оператора for происходит следующим образом:

1) Вычисляется начальное-выражение, если оно есть.
2) Вычисляется условное-выражение, если оно есть. Возможны три результата:
3) Если значение условного-выражения "истина" (ненулевое), то выполняется оператор. Затем вычисляется выражение-цикла, если оно есть. Процесс начинается снова с вычисления условного-выражения.
4) Если условное-выражение опущено, то считается, что его значение "истина" и процесс выполнения протекает так, как это описано в первом случае. Оператор for без аргумента условного-выражения прекращает

свое выполнение только в случае выполнения оператора break или return в теле оператора, или при выполнении оператора goto (который передаст управление на оператор с меткой вне тела оператора for).

5) Если значение условного-выражения "ложь", то выполнение оператора for прекращается и управление передается к следующему оператору программы.

Оператор for также прекращает свое выполнение при выполнении в теле оператора оператора break, goto или return.

- do

Синтаксис: do

оператор

while (выражение);

Тело оператора do выполняется один или несколько раз до тех пор, пока значением выражения не станет "ложь" (0). Выполнение происходит следующим образом:

1) Выполняется тело оператора.

2) Вычисляется выражение. Если его значение "ложь", то выполнение оператора do заканчивается и управление передается следующему оператору программы. Если его значение "истина" (ненулевое значение), то процесс повторяется, начиная с шага 1.

Выполнение оператора do может быть прервано выполнением оператора break, goto или return в теле оператора do.

- while

Синтаксис: while (выражение)

оператор

Выполнение

Тело оператора while выполнится ноль или более раз до тех пор, пока значением выражения не станет "ложь" (0). Процесс выполнения протекает следующим образом:

1) Вычисляется значение выражения.
2) Если значение выражения есть "ложь", то тело оператора while не выполняется, и управление передается на следующий за оператором while оператор программы.
3) Если значение выражения есть "истина" (не ноль), то выполняется тело оператора и процесс повторяется с шага 1.

- switch case

Синтаксис: switch (выражение) {

[объявление]

.

.

.

[case постоянное-выражение:]

.

.

.

[оператор]

.

.

.

[default:

[оператор]]

}

Оператор switch передает управление на оператор в своем теле. Управление будет передано тому оператору, значение case постоянное-выражение которого совпадает с выражением switch. Оператор switch может содержать любое число элементов case. Выполнение тела оператора начинается в выбранном операторе и заканчивается в конце тела или в тот момент, когда оператор передаст управление вне тела.

Оператор default выполняется в том случае, если ни одно постоянное-выражение case которого не совпадет с выражением switch. Если оператор default не задан и ни одно совпадение с case не обнаружено, то ни один из операторов тела switch не будет выполнен. Располагать оператор default в конце не обязательно, он может появиться в произвольном месте тела оператора switch.

Выражение switch должно иметь интегральный тип, но результирующее значение будет преобразовано в int. Затем каждое постоянное-выражение case будет преобразовано с использованием обычных арифметических преобразований. Значения всех постоянных-выражений case должны быть разными в теле оператора. если тип выражения switch больше int, то появится диагностическое сообщение.

Метки case и default тела оператора switch действуют только при первоначальной проверке, определяющей начало выполнения тела цикла. Все операторы от начала выполнения и до конца тела выполняются независимо от их меток, кроме случая, когда управление передается в часть программы вне тела оператора.

- goto

Синтаксис: goto имя;

.

.

.

имя: оператор

Оператор goto передает управление непосредственно на оператор, который имеет своей меткой "имя". Оператор сметкой будет выполнен сразу после выполнения оператора goto. Оператор с заданной меткой должен находиться в той же самой функции и заданная метка может помечать только один оператор в данной функции.

Метка оператора имеет смысл только для оператора goto. В любом другом контексте оператор с меткой выполняется так, как если бы ее вообще не было.

- break

Синтаксис: break;

Оператор break прекращает выполнение вложенного оператора do, for, switch или while, в котором он появляется. Управление передается тому оператору, который непосредственно следует за прерванным оператором. Оператор break может появиться только в операторах do, for, switch или while.

- Continue

Синтаксис: continue;

Оператор continue передает управление на следующую итерацию оператора do, for или while, в котором он появляется передавая все оставшиеся операторы тела оператора do, for или while. Следующая итерация оператора do, for или while понимается следующим образом:

1) В операторе do или while следующая итерация начинается вычислением выражения оператора do или while.
2) В операторе for следующая итерация начинается вычислением выражения цикла оператора for. После вычисления условного выражения в зависимости от его результатов происходит либо прекращение выполнения оператора либо выполнение его тела.

- return

Синтаксис: return [выражение]

Выполнение

Оператор return прекращает выполнение функции, в которой он появляется и передает управление на вызов функции. Выполнение программы продолжается непосредственно с той точки, откуда был произведен вызов функции. Значение выражения, если оно есть, передается на вызов функции. Если выражение не задано, то возвращаемое функцией значение не определено.

1.7. Особенности Языка (вероятны доработки)

Данный язык будет языком программирования с С подобным синтаксисом, содержащий некоторые особенности,

которые нацелены на облегчение понимания, увеличения простоты написания программ, и читаемость кода.

Список особенностей:

- Одно выражение в одной строке

Пример:

```
int q;
int z;
q = 0;
z = 0;
q = z + 12;
z = q + 5;
```

- Низкоуровневые операции выполняются с помощью встроенных функций(операции работы со строками, математические, операции работы с массивами)

- Отсутствие указателей

В языке «Крабик (V)_ii_(V)» не поддерживаются следующие конструкции(и им подобные)

```
q + z; z - 12;

int q = z - 5 + 22;

for(int i = 0; i < 4; i++ for(;;));;
```

Приложение 1.

Примеры программ на языке “Крабик (V)_ii_(V)”.

```
Var MakeMeCoffee (integer wishes)
{
	Integer Cofe = 26 + 4 / 5;
	Return Cofe;
}
Var AndBucketOfCola (integer wishes)
{
	Integer Cola = 2 + 3 – 4;
	Return Cola;
}

Integer main(integer wishes)
{
	MakeMeCoffee(wishes&);
	AndBucketOfCola(wishes&);
}
```