УВАЖАЕМЫЕ СТУДЕНТЫ!

ВАМ НЕОБХОДИМО ВЫПОЛНИТЬ СЛЕДУЮЩЕЕ:

1. Ознакомиться с теорией и законспектировать, составить и ответить на вопросы.

2. Предоставит отчет, прислать в виде скриншото в течении трех дней .

3. Отправить преподавателю на почту **v.vika2014@mail.ru** и указать свою Ф.И.О, группу, и название дисциплины тел 072-17-44-9-22

Тема: Основные понятия структурного программирования.

Структурное программирование

Структурное программирование – идеология создания программ, основанная на **теореме о структурировании**: любая программа может быть написана с использованием трех видов базовых управляющих структур – следования, циклов и развилок. Структуры могут иерархически вкладываться друг в друга.

В соответствии с данной методологией

1. Любая программа представляет собой структуру, построенную из трёх типов базовых конструкций:

- **последовательное исполнение** — однократное выполнение операций в том порядке, в котором они записаны в тексте программы;
- **ветвление** — однократное выполнение одной из двух или более операций, в зависимости от выполнения некоторого заданного условия;
- **цикл** — многократное исполнение одной и той же операции до тех пор, пока выполняется некоторое заданное условие (условие продолжения цикла). В программе базовые конструкции могут быть вложены друг в друга произвольным образом, но никаких других средств управления последовательностью выполнения операций не предусматривается.

2. Разработка программы ведётся пошагово, методом «сверху вниз».

Идентификатор

Ид-р - просто ИМЯ некоей сущности в программе (переменная, константа, функция, тип и т.п.).

Правила создания **идентификаторов**.

- идентификаторы могут состоять из символов латинских букв, арабских цифр 0-9 и символа подчеркивания (_);
- идентификаторы могут начинаться с символа подчеркивания или с буквы;
- длина идентификатора ограничивается 255 символами (учитываются первые 31);
- идентификаторы должны быть информативными, т.е. соответствовать смыслу соответствующей именованной сущности (переменной, константы, типа, процедуры и т.п.);
- регистр букв в идентификаторах учитывается (в языке C/C++);
- зарезервированные слова нельзя использовать в качестве идентификаторов.

Идентификатор создается на этапе объявления переменной, функции, структуры и т.п. после этого его можно использовать в последующих инструкциях разрабатываемой программы.

Следует отметить важную особенность при выборе идентификатора: идентификатор не должен совпадать с ключевыми словами, с зарезервированными словами и именами функций библиотеки компилятора языка СИ.

Константы и переменные

1. Константы

Константа – именованная область оперативной памяти, выделенная для хранения значения. Константа получает свое значение при объявлении. Не допускается его изменения. Константы определяют в начале программы. После определения они могут использоваться во всей программе.

CONST Имя_константы = Выражение [, Имя_константы "Выражение]...

Выражение - Значение, которое должно храниться в константе. Это значение может быть просто числом, другой константой, текстом или арифметическим выражением.

2. Переменные

Переменная – область оперативной памяти, используемая в программе для хранения значения определенного типа. Для обращения к значению переменной может использоваться ее *имя* либо различные способы *адресации*. Значение переменной может быть сколько угодно раз изменено в процессе выполнения программы.

В отличие от констант, значения переменных могут меняться в процессе выполнения программы.

Примечание: Следите за тем, чтобы имена переменных и констант соответствовали их содержанию. С переменными height!, telefon$, length! сразу **ассоциируется их применение.**

Тип данных

Тип данных — фундаментальное понятие теории программирования. Тип данных определяет множество значений, набор операций, которые можно применять к таким значениям и, возможно, способ реализации хранения значений и выполнения операций. Любые данные, которыми оперируют программы, относятся к определённым типам.

Тип (сорт) — относительно устойчивая и независимая совокупность элементов, которую можно выделить во всём рассматриваемом множестве (предметной области). Полиморфный тип — представление набора типов как единственного типа.

Типы данных бывают следующие:

1. Простые.

- Перечислимый тип. Может хранить только те значения, которые прямо указаны в его описании.
- Числовые. Хранятся числа. Могут применяться обычные арифметические операции.

- Целочисленные: со знаком, то есть могут принимать как положительные, так и отрицательные значения; и без знака, то есть могут принимать только неотрицательные значения.
- Вещественные.
- Символьный тип. Хранит один символ. Могут использоваться различные кодировки.
- Логический тип. Имеет два значения: истина и ложь, при троичной логике может иметь и третье значение — «не определено» (или «неизвестно»). Могут применяться логические операции. Используется в операторах ветвления и циклах. В некоторых языках является подтипом числового типа, при этом ложь=0, истина=1.

2. Составные (сложные). Множество. В основном совпадает с обычным математическим понятием множества. Допустимы стандартные операции с множествами и проверка на принадлежность элемента множеству. В некоторых языках рассматривается как составной тип.

Массив. Является индексированным набором элементов одного типа. Одномерный массив — вектор, двумерный массив — матрица.

- Строковый тип. Хранит строку символов. Аналогом сложения в строковой алгебре является конкатенация (прибавление одной строки в конец другой строки). В языках, близких к бинарному представлению данных, чаще рассматривается как массив символов, в языках более высокой абстракции зачастую выделяется в качестве простого.

Запись (структура). Набор различных элементов (полей записи), хранимый как единое целое. Возможен доступ к отдельным полям записи. Например, struct в C или record в Pascal.

Файловый тип. Хранит только однотипные значения, доступ к которым осуществляется только последовательно (файл с произвольным доступом, включённый в некоторые системы программирования, фактически является неявным массивом).

Класс.

Указатель. Хранит адрес в памяти компьютера, указывающий на какую-либо информацию, как правило — указатель на переменную.

Выражения

Выражение – конструкция языка, описывающая последовательность операций над данными, результатом выполнения которых является значение некоторого типа. Выражения могут быть арифметическими, логическими, символьными и другими в зависимости от типа результата. В общем случае выражения состоят из *операций*, *операндов* и *круглых скобок*. Операндами могут быть литералы, константы, переменные, вызовы функций, выражения. Последовательность выполнения операций определяется их приоритетом, ассоциативностью и расположением круглых скобок.

Привести пример выражения

Инструкция присваивания

Инструкция присваивания является основной вычислительной инструкцией, подразумевающей вычисление выражения справа от оператора присваивания и сохранение его результата в переменной, имя которой указано слева от оператора присваивания.

В общем виде инструкция присваивания выглядит так:

ИмяПеременной:= Выражение;

где:

Имя - переменная, значение которой изменяется в результате выполнения инструкции присваивания;

:= - символ инструкции присваивания.

выражение - выражение, значение которого присваивается переменной, имя которой указано слева от символа инструкции присваивания. Тип переменной и тип результата выражения должны быть совместимы.

Пример:

*Summa := Cena * Kol;*

Skidka := 10;

Found := False;

Базовые управляющие структуры

Управляющие структуры - предназначены для управления последовательностью выполнения инструкций в программе, а именно, для организации ветвлений и повторений в алгоритмах.

Базовый управляющие Структуры:

1. условные операторы
2. циклы
3. блоки (следование, составная инструкция).