1) Оператор условной передачи управления в программе используют для

программирования ветвлений, т. е. таких вычислений, когда при определенных условиях возникает необходимость выбора одного из возможных вариантов вычислений.

2)Простыми называются операторы условной передачи управления с проверкой одного условия. Сложными – операторы с проверкой двух (нескольких) условий. Простые операторы условной передачи, в свою очередь, по сложности проверяемого условия можно разделить на элементарные и составные.

3) Опера́тор ветвле́ния (усло́вная инстру́кция, усло́вный опера́тор) — оператор, конструкция языка программирования, обеспечивающая выполнение определённой команды (набора команд) только при условии истинности некоторого логического выражения, либо выполнение одной из нескольких команд (наборов команд) в зависимости от значения некоторого выражения.

4) Ветвящимся называется такой вычислительный процесс, в котором выбор направления обработки информации зависит от исходных или промежуточных данных (от результатов проверки выполнения какого-либо логического условия).

5) оператор:

if операция:

if(a >b)

c=a+b;

Оператор - то, что исполняет. операция - то, что исполняется.

6.Что такое вложенный условный оператор? Как определяется соответствие между операторами if и else во вложенных условных операторов?

расставлением скобок типа {} в Си или ключевых слов begin end в паскале (и соответственно делфи) . Один блок имеет структуру

if (условие)

then

begin

действия

end

else

begin

действия

end;

В Си, вместо begin ставится открывающаяся скобка { , вместо end закрывающая }. Соответственно, при наличии вложенных условий, вложенный условия вместе со своими скобками или begin-end олжны полностью располагаться внутри бока begin-end внешнего условия.

7. Что такое условная операция и чем она отличается от условного оператора?

### Условные операторы

Условные *операторы* изменяют ход выполнения программы в зависимости от некоего условия. Существуют конструкции if…then, условная операция ( ?) и *переключатель* switch. При разработке программ, использующих условные *операторы*, нужно учитывать, что как и в С++, при проверке равенства двух величин нужно писать a==b, а не a=b, т.к. *выражение* a=b в условном операторе будет расценено как *оператор присваивания*. Второе. При разработке составных условий *выражение* a>3 && b<5 обозначает операцию конъюнкции, т.е. должны выполняться оба условия. *Выражение* a>3 || b<5 обозначает дизъюнкцию, т.е. для истинности выражения достаточно выполнение хотя бы одного условия.

#### Оператор if..then

Синтаксис оператора if…then следующий:

If (Условие) {Действие 1} else {Действие 2}

В случае истинности условия выполняется первое действие, в противном случае - второе действие.

В предлагаемом примере сравниваются два числа, и находится наименьшее.

var a, b, c, str;

a = 1;

b = 300;

if (a < b) { c = a; }

else { c = b; }

str = "Минимальное значение из двух чисел равно " + c;

document.write(str);

8. Какие операции позволяют формировать условие в условном операторе и условной операции?

В языке Си имеется короткий способ записи одного из видов оператора **if-else**. Он называется "условным выражением" и использует операцию условия - **?:**. Эта операция состоит из двух частей и содержит три операнда. Ниже приводится пример оператора с помощью которого находится абсолютное значение числа:

*x = (y < 0 )? -y : y;*

Все, что находится между знаком **=** и символом "точка с занятой" представляет собой **условное выражение**. Смысл этого оператора заключается в следующем: если **у** меньше 0, то **х = - у**; в противном случае **х = у**. В терминах оператора **if-else** данный оператор мог выглядеть так:

*if(у < 0)         x = (y < 0 )? -y : y;*

*х = -у;*

*else*

*х = у;*

В общем виде условное выражение можно записать следующим образом:

*выражение1* ? *выражение2* : *выражение3*

Если *выражение1* истинно (больше нуля), то значением всего условного выражения является величина *выражения2*;

если *выражение1* ложно (равно 0), то значение всего условного выражения - величина *выражения3*.

     Условное выражение удобно использовать в тех случаях, когда имеется некоторая переменная, которой можно присвоить одно из двух возможных значений. Типичным примером является присваивание переменной значения большей из двух величин:

*mах = (а > b)? а : b;*

9) Составной оператор — конструкция языка программирования, состоящая из нескольких команд (операторов) языка программирования, но участвующая в программе в качестве единого оператора.

Составной оператор предназначен для объединения нескольких действий. Он имеет вид:begin

операторы

end;

В условном операторе составной оператор используется для описания нескольких действий: если истина - то составной оператор, если ложь то составной оператор.

10. Из каких компонентов состоит оператор множественного ветвления?

Оператор служит для выбора одного из помеченных вариантов действия (операторов), в зависимости от значения "параметра". Оператор имеет вид:

Case n Of список помеченных операторовElse "оператор";End;

Здесь n - выражение или переменная порядкового типа.

Из "списка помеченных операторов" выполняется оператор с меткой, включающей значение "параметра", иначе оператор после слова Else.

Конструкция Else "оператор" может отсутствовать. Напомним, что "оператор" может иметь вид: Begin "операторы" end;

Пример операторов для определения порядка целого числа N от 0 до 999:

case N of 0..9 : writeln('однозначное'); 10..99 : writeln('двузначное'); 100..999 : writeln('трехзначное') else writeln('Число "N" не входит в указанный диапазон') end;

11)Ключевое слово default введено в C++ 11. Его использование указывает компилятору самостоятельно генерировать соответствующую функцию класса, если таковая не объявлена в классе. Компилятор автоматически генерирует ряд конструкторов класса и деструктор.

Если это значение не совпадает ни с одним из перечисленных значений

и оператор default присутствует, выполняется следующий за ним оператор.

Если не нашлось соответствующего значения выражения выбора и оператор

default отсутствует, то действие оператора switch аналогично пустому оператору.

12) Как выполняется отладка программ, содержащих ветвления вычислительного процесса?

13) Формально к операторам передачи управления относятся:

-оператор безусловного перехода goto;

-оператор перехода к следующему шагу (итерации) цикла continue;

-выход из цикла или оператора switch - break;

-оператор возврата из функции return.

Рассмотрим их более подробно. В языке Си предусмотрен оператор goto, хотя в большинстве случаев можно обойтись без него. Общий вид оператора

1.1)goto метка;

Он предназначен для передачи управления на оператор, помеченный меткой. Метка представляет собой идентификатор, оформленный по всем правилам идентификации переменных с символом «двоеточие» после него, например, пустой помеченный оператор:

m1 : ;

Область действия метки - функция, где эта метка определена. Программа с goto может быть написана без него за счет повторения

некоторых проверок и введения дополнительных переменных.

Наиболее характерный случай использования оператора goto - выполнение прерывания (выхода) во вложенной структуре при возникновении грубых неисправимых ошибок во входных данных. И в этом случае необходимо выйти из двух (или более) циклов, где нельзя использовать непосредственно оператор break, т.к. он прерывает только самый внутренний цикл:

for (...)

for (...)

{

...

}

if ( ошибка ) goto Еrror;

...

Еrror : - операторы для устранения ошибки;

Если программа обработки ошибок сложная, а ошибки могут возникать в нескольких местах, то такая организация оказывается удобной.

1.2. Оператор continue

Этот оператор может использоваться во всех типах циклов, но не в операторах switch. Наличие оператора continue вызывает пропуск "оставшейся" части итерации и переход к началу следующей, т.е. досрочное завершение текущего шага и переход к следующему шагу.

В циклах while и do это означает непосредственный переход к проверочной части. В цикле for управление передается на шаг коррекции, т.е. модификации выражения.

Фрагмент программы обработки только положительных элементов массива a, отрицательные значения пропускаются:

for ( i = 0; i<n; i++) {

if( a[i]<0) continue;

обработка положительных элементов;

}

1.3. Оператор break

Оператор break производит экстренный выход из самого внутреннего цикла или оператора switch, к которому он принадлежит, и передает управление первому оператору, следующему за текущим оператором.

1.4. Оператор return

Оператор return; производит досрочный выход из текущей функции. Он также возвращает значение результата функции:

return выражение;

В функциях, не возвращающих результат, он неявно присутствует после последнего оператора. Значение выражения при необходимости преобразуется к типу возвращаемого функцией значения.

14)Оператор перехода

В языке С определены четыре оператора перехода: return, goto, break и continue. Операторы return и goto можно использовать в любом месте внутри функции. Операторы break и continue можно использовать в любом из операторов цикла. Как указывалось ранее в этой главе, break можно также использовать в операторе switch.

15) Метка (англ. label) — символьное имя, идентификатор для более удобного указания данных и кода в языках программирования. Позволяет программисту обходиться без вычисления и пересчёта адресов и смещений внутри программы (эти действия за него выполняет компилятор). Несмотря на то что большинство языков программирования высокого уровня поддерживают операции с метками, их использование крайне нежелательно, так как программный код становится плохочитаемым, и отладка такой программы занимает большее время.

При компиляции каждой метке сопоставляется определённый адрес (число), который заносится в исполняемый модуль. Таким образом, в случае необходимости компилятор может вычислить смещение между метками (разность адресов).

16) Метки независимы от имен, используемых для переменных и других идентификаторов. Следовательно, у метки может быть тот же идентификатор, что и у другой сущности в программе, не вступая в конфликт с другим одноименным идентификатором. Оператор goto и помеченный оператор, на который он передает управление, должны находиться в той же функции

17) Линейный алгоритм

Линейный алгоритм – описание действий, которые выполняются однократно в заданном порядке. Исполнитель выполняет действия последовательно, одно за другим в том порядке в котором они следуют.

18) В теории программирования доказано, что программу для решения задачи любой сложности можно составить только из трех структур, называемых следованием, ветвлением и циклом. Следованием называется конструкция, представляющая собой последовательное выполнение двух или более операторов (простых или составных). Ветвление задает выполнение либо одного, либо другого оператора в зависимости от выполнения какого-либо условия. Цикл задает многократное выполнение оператора ( рис. 2.1).

Рис. 2.1. Базовые конструкции структурного программирования

Следование, ветвление и цикл называют базовыми конструкциями структурного программирования. Их особенностью является то, что любая из них имеет только один вход и один выход, поэтому они могут вкладываться друг в друга. Например, цикл может содержать следование из нескольких ветвлений, каждое из которых включает вложенные циклы.

Целью использования базовых конструкций является получение программы простой структуры. Такую программу легко читать, отлаживать и при необходимости модифицировать. Язык Паскаль способствует созданию хорошо структурированных программ, поскольку базовые конструкции реализуются в нем непосредственно с помощью соответствующих операторов.

Операторы ветвления

Операторы ветвления if и варианта case применяются для того чтобы в зависимости от конкретных значений исходных данных обеспечить выполнение разных последовательностей операторов. Оператор if обеспечивает передачу управления на одну из двух ветвей вычислений, а оператор case — на одну из произвольного числа ветвей.

Условный оператор if

Условный оператор if используется для разветвления процесса вычислений на два направления. Структурная схема оператора приведена на рис. 2.2. Формат оператора:

if выражение then оператор\_1 [else оператор\_2;]

Сначала вычисляется выражение, которое должно иметь логический тип. Если оно имеет значение true, выполняется первый оператор, иначе — второй. После этого управление передается на оператор, следующий за условным.

Oператоры, входящие в состав условного оператора, могут быть простыми или составными. Составной оператор ( блок ) обрамляется ключевыми словами begin и end. Блок применяют в том случае, когда по какой-либо ветви требуется выполнить несколько операторов: ведь иначе компилятор не сможет понять, где заканчивается ветвь и начинается следующая часть программы. Одна из ветвей может отсутствовать.

19) Какие логические операции используются в C/C++?

В языке программирования C/C++ используются следующие логические операции:

&& – логическое «И»;

|| – логическое «ИЛИ»;

! – логическое «НЕТ».

Результатом логических операций есть значение false или true. В языке C++ принято, что значение false считается равным 0, а значение true считается равным 1.

23. Дайте характеристику свойств и методов объекта RadioButton

Элемент управления RadioButton

Элемент управления RadioButton, подобно CheckBox используется для фиксирования выбора пользователя. Однако отличается от CheckBox тем, что обычно применяется в ситуациях, когда пользователь должен выбрать единственный вариант из ряда предлагаемых опций.

Один из способов управления набором опций с возможностью единственного выбора - задать элементы управления RadioButton, соответствующие опциям, как элементы одного уровня в контейнере.

Кроме того, опции можно сгруппировать, задавая для каждого RadioButton свойство GroupName (Имя группы). В предыдущем примере в StackPanel располагались опции 1 -4, и Silverlight разрешал пользователю приложения выбирать только одну из них.

Если потребовалось бы разделить этот набор опций на две группы, чтобы пользователь мог выбирать одну из опций 1 и 2, и одну из опций 3 и 4, вместо контейнера можно было бы использовать группы RadioButton.

. Дайте характеристику свойств и методов объекта RadioGroup

Радиокнопки образуют группы взаимосвязанных индикаторов, из которых обычно может быть выбран только один. Они используются для выбора пользователем одной из нескольких взаимоисключающих альтернатив, например, отдела, в котором работает сотрудник, или пола сотрудника. Впрочем, радиокнопки могут использоваться и для отображения аналогичных данных. В этом случае управление кнопками осуществляется программно. Несколько примеров организации размещения радиокнопок вы можете увидеть на рис. 7. в разделе выше.

Начнем рассмотрение радиокнопок с компонента RadioGroup — панели группы радиокнопок. Это панель, которая может содержать регулярно расположенные столбцами и строками радиокнопки. Надпись в левом верхнем углу панели. определяется свойством Caption. А надписи кнопок и их количество определяются свойством Items, имеющим тип TStrings. Щелкнув на кнопке с многоточием около этого свойства в окне Инспектора Объектов, вы попадете в редактор списков строк, который уже рассматривался нами в разделе выше . В нем вы можете занести надписи, которые хотите видеть около кнопок, по одной в строке. Сколько строчек вы запишете — столько и будет кнопок. Например, для компонента RadioGroup в верхней правой части формы свойство Items имеет вид:

Кнопки, появившиеся в панели после задания значений Items, можно разместить в несколько столбцов (не более 17), задав свойство Columns. По умолчанию Columns = 1, т.е. кнопки размещаются друг под другом.

Определить, какую из кнопок выбрал пользователь, можно по свойству ItemIndex, которое показывает индекс выбранной кнопки. Индексы, как всегда в Delphi, начинаются с 0. По умолчанию ItemIndex = -1, что означает отсутствие выбранной кнопки. Если вы хотите, чтобы в момент начала выполнения приложения какая-то из кнопок была выбрана (это практически всегда необходимо), то надо установить соответствующее значение ItemIndex во время проектирования. Если вы используете радиокнопки не для ввода, а для отображения данных, устанавливать значение ItemIndex можно программно во время выполнения приложения.

Компонент RadioGroup очень удобен, но не свободен от некоторых недостатков. Его хорошо использовать, если надписи кнопок имеют примерно одинаковую длину и если число кнопок в каждом столбце (при размещении их в нескольких столбцах) одинаково. Посмотрите, например. из раздела выше. Группа радиокнопок в нижнем левом углу формы имеет нормальный вид. А группа аналогичных радиокнопок в верхней правой части формы выглядит плохо: она занимает слишком много места, которое пропадает в пустую. Связано это с тем, что длина надписей у кнопок первого столбца превышает длину надписей у остальных кнопок. A RadioGroup при размещении кнопок ориентируется на надпись максимальной длины. Еще хуже выглядела бы эта группа, если бы число кнопок было, например, равно 5.

В подобных случаях желательно нерегулярное расположение кнопок. Такую возможность дают компоненты RadioButton, сгруппированные панелью GroupBox. Панель GroupBox выглядит на форме так же, как RadioGroup, и надпись в ее верхнем левом углу также определяется свойством Caption. Эта панель сама по себе пустая. Ее назначение — служить контейнером для других управляющих элементов, в частности, для радиокнопок RadioButton. Отдельная радиокнопка RadioButton особого смысла не имеет, хотя и может служить индикатором, включаемым и выключаемым пользователем. Но в качестве индикаторов обычно используются другие компоненты — CheckBox. А радиокнопки имеют смысл, когда они взаимодействуют друг с другом в группе. Эта группа и объединяется единым контейнером, обычно панелью GroupBox.

25. Дайте характеристику свойств и методов объекта GroupBox

Компоненты GroupBox и CheckBox

GroupBox во многом похож на обычную панель. Он предназначен для группировки схожих по смыслу компонентов. Текст из свойства Caption выходит не посреди панели, а вверху, создавая таким образом заглавие этой панели. При перемещении GroupBox, будут перемещаться и все компоненты, расположенные на нем.

Создайте новое приложение. Мы сделаем полезную утилиту, меняющую вид формы в зависимости от настроек. На форму установите компонент GroupBox, присвоив его свойствам Left и Top значение 6, чтобы прижать его к верхнему левому краю окна. В свойстве Caption этого компонента напишите текст "BorderIcons". Поскольку обрамление компонента вплотную подходит к тексту – заголовку, с дизайнерской точки зрения будет нелишним добавить пару пробелов перед текстом, и столько же после него.

Мы уже знаем, что BorderIcons – это свойство формы, программирующее ее внешний вид. Это свойство имеет 4 значения, каждое из которых можно включить, либо выключить. Таким образом, нам нужно 4 классических "флажка", в которые можно будет поставить "галочку" или убрать ее. Роль таких флажков выполняет компонент CheckBox.