Презентация на тему:

Управляющие операторы языков программирования C, C++, VBA, Pascal.

Выполнила студентка 1-го курса экономического ф-та группы менеджмент Зинович Наталья.




Содержание:

ВВЕДЕНИЕ

Оператор – это часть программы, которая может быть выполнена отдельно. Это означает, что оператор определяет некоторое действие.
Операторы строятся из специальных зарезервированных слов, логических выражений и других операторов.
В зависимости от языка программирования операторы слегка отличаются друг от друга, но в целом они очень похожи.
Существуют следующие группы операторов:
Условные операторы
Операторы цикла
Операторы безусловного перехода
Метки
В языке С к условным относятся операторы if и switch. Иногда их также называют *операторами условного перехода.* Операторы цикла – это while, for, do-while. К операторам безусловного перехода относятся break, continue, goto и return. К меткам относятся операторы case, default и собственно метки. Операторы- выражения – это операторы, состоящие из допустимых выражений. Блок представляет собой фрагмент текста программы, обрамленный фигурными скобками {}. Блок иногда называют *составным оператором.*

В СОДЕРЖАНИЕ

Оператор if… then… в Pascal, Delphi

Оператор if… then… называется условным и имеет вид:
If <условие-1> Then <команды-1>
где <условие> – это некое логическое выражение.
Логическое выражение принимает одно из двух возможных значений – true(истина) или false(ложь). Часто в роли логического выражения выступает какое-то условие, которое может выполняться либо нет. В первом случае его значение - «истина», а во втором – «ложь». Если логическое выражение <*условие-1*> принимает значение «истина», то выполняется оператор <*команды-1*>. В противном случае выполняется оператор, следующий за данным логическим оператором.

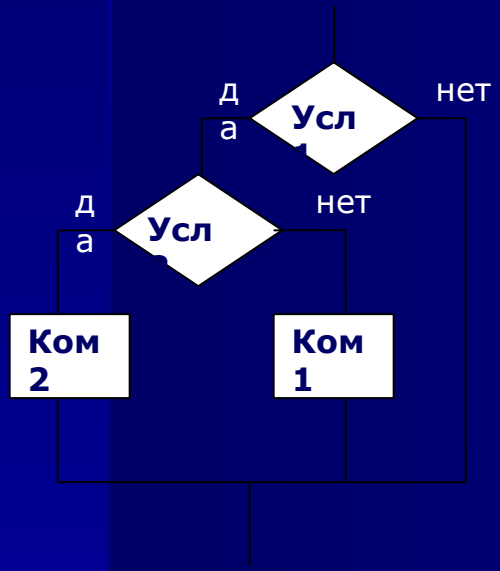
Операторы if… then… можно вкладывать друг в друга, так как конструкция
If <условие-2> Then <команды-2>;
также является оператором и может заместить оператор <*команды-1*>:
If <условие-1> Then If <условие-2> Then <команды-2>;
Пример условного оператора:
If Grade=0 Then Write('температура замерзания воды');

В СОДЕРЖАНИЕ

Оператор if… then… else… в Pascal, Delphi

Этот оператор является полной версией условного оператора и имеет вид:
If <условие-1> Then <команды-1> els<команды2>;
Выполняется данный оператор следующим образом: если выражение *<условие-1>* принимает значение «истина», то управление передается на оператор *<команды-1>*. В противном случае на оператор *<команды-2>*.
Следует учесть, что каждому else соответствует *ближайший* предшествующий if. Т.е. оператор

```
If <условие-1> Then
 If <условие-2> Then
   <команды-2>
        Else
   <команды-1>;
```



Равносилен оператору

```
If <условие-1> Then
        begin
 If <условие-2> Then
   <команды-2>
        Else
   <команды-1>;
```

Чтобы четко определить, что чему подчинено, необходимо использовать begin…end.
Пример условного оператора:

```
If Two=2 Then
  Writeln('два равно 2')
Else
   Writeln('Это не 2!')
```

В СОДЕРЖАНИЕ

Вложенные операторы if… then… else… в Pascal, Delphi

Как уже отмечалось, условные операторы можно вкладывать друг в друга, программируя таким образом сложные ветвления. Рассмотрим следующий оператор:

If <условие-1> Then
<команды-1>
Else If <условие-2> Then
<команды-2>
Else If <условие-3> Then
<команды-3>
….
Else If <условие-n> Then
<команды-n>

Вначале вычисляется значение логического выражения <условие-1>. Если оно истинно, выполняется оператор <команды-1>, если же это значение ложно, вычисляется значение выражения <условие-2>. В том случае, когда полученное значение истинно, будет выполняться оператор <команды-2>, при значении «ложь» будет вычисляться выражение <условие-3> и т. д.

Если выражения <условие-i> независимы, то есть вычисление их значений в любом порядке дает один и тот же результат для каждого из них, имеет смысл располагать их в таком порядке, чтобы выражение, с наибольшей вероятностью принимающее значение «истина», стояло на первом месте, выражение, принимающее значение «истина» с меньшей вероятностью, — на втором и т. д.. Это уменьшит время выполнения данного фрагмента программы, особенно если вложенный оператор появляется в цикле, который выполняется многократно.

Пример вложенных условных операторов:

```
if Two = 2 then
if One = 1 then
Writeln('Единица  равна  1')
else
Writeln('Единица не равна  1 ‘)
else
if Three = 3 then
Writeln(‘Три равно 3‘)
else
Writeln('Три не равно 3');
```

В СОДЕРЖАНИЕ

Оператор if…then… в С

В языках С, С++ оператор if…then будет выглядеть следующим образом:

If (условие) команда1; Else команда2;

Или, если операторы вложены друг в друга:

```
If (условие1)
    Команды1;
  Else
If (условие2)
  Команды2;
    Else
    Команды3;
```

В настоящее время большинство компиляторов допускают больше 100 уровней вложенности. Однако на практике необходимость в глубине вложенности, большей, чем несколько уровней, возникает довольно редко, так как увеличение глубины вложенности быстро запутывает программу и делает ее нечитаемой.

В программах часто используется конструкция, которую называют *лестницей if-else-if.* Общая форма лестницы имеет вид

```
if (выражение) оператор;
else if (выражение) опера тор;
else if (выражение) оператор;
     else оператор;
```

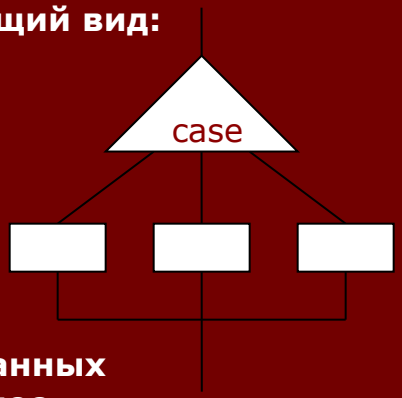
Работает эта конструкция следующим образом. Условные выражения операторов if вычисляются сверху вниз. После выполнения некоторого условия, т.е. когда встретится выражение, принимающее значение ИСТИНА, выполняется ассоциированный с этим выражением оператор, а оставшаяся часть лестницы пропускается. Если все условия ложны, то выполняется оператор в последней фразе else, а если последняя фраза else отсутствует, то в этом случае не выполняется ни один оператор.

В СОДЕРЖАНИЕ

Оператор case...of...end в Pascal, Delphi

Для ситуаций, где имеется несколько (три и более) альтернатив, больше подходит оператор case. Этот оператор называется *оператором выбора* и имеет следующий вид:

case <тестируемое выражение> of
<список-1> : <команды-1>;
< список-2> : <команды-2>;
< список-3> : <команды-3>;
...
< список -n> : <команды-n>;
Else <команды>;
End;



Рассмотрим элементы этой конструкции. Во-первых, это три зарезервированных слова: case, of и end. Между case и of находится выражение *<тестируемое выражение>,* принимающее значение, которое, возможно, имеется в одном из списков значений, находящихся слева от двоеточий. Данное выражение называется *селектором* оператора case. Каждый оператор, идущий за двоеточием, отделяется от следующего списка значений точкой с запятой. Ветвь else, отвечающая всем не перечисленным значениям выражения *<тестируемое выражение>*, необязательна. При выполнении данного оператора вначале вычисляется значение селектора. Затем выбирается тот список значений, которому принадлежит полученное значение, и выполняется соответствующий оператор.

В списках значений оператора case допустимыми являются типы переменных, называемые *скалярными* (к скалярным относятся целый, символьный, булев и пере числимые типы), включая целые и исключая вещественные типы. Любое заданное значение селектора может входить в список значений неоднократно, но выполняться будет лишь первая подходящая ветвь. Заметим, что «стилистически» такая конструкция выглядит не очень изящно. Если значение селектора отсутствует в списках значений, ни одна из альтернатив выполняться не будет. В этом случае выполняется ветвь else оператора case или (если эта ветвь отсутствует) следующий за case оператор.

В СОДЕРЖАНИЕ

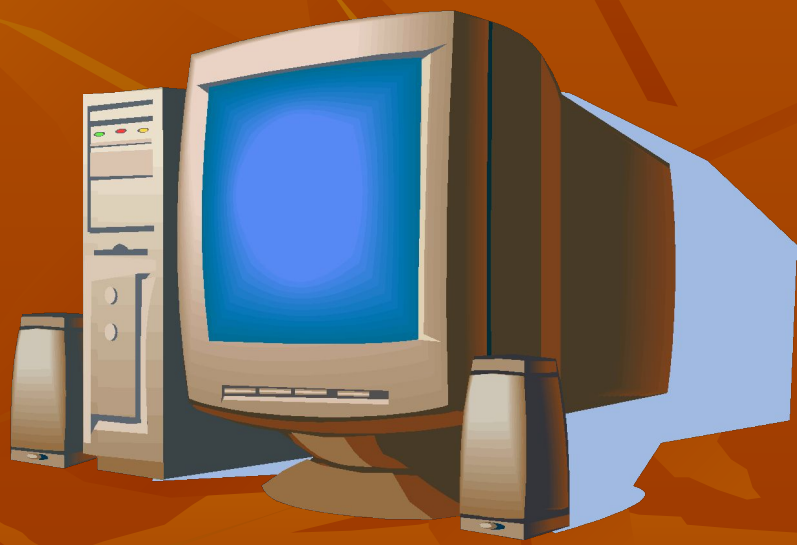
Оператор switch в С

Оператор выбора switch (часто его называют переключателем) предназначен для выбора ветви вычислительного процесса исходя из значения управляющего выражения. (При этом значение управляющего выражения сравнивается со значениями в списке целых или символьных констант. Если будет найдено совпадение, то выполнится ассоциированный с совпавшей константой оператор.)

Об операторе switch очень важно помнить следующее:

- Оператор switch отличается от if тем, что в нем управляющее выражение проверяется только на *равенство* с постоянными, в то время как в if проверяется любой вид отношения или логического выражения.
- В одном и том же операторе switch никакие два оператора case не могут иметь равных постоянных. Конечно, если один switch вложен в другой, в их операторах case могут быть совпадающие постоянные.
- Если в управляющем выражении оператора switch встречаются символьные константы, они автоматически преобразуются к целому типу по принятым в языке С правилам приведения типов.

Оператор switch часто используется для обработки команд с клавиатуры, например, при выборе пунктов меню.



В СОДЕРЖАНИЕ

Вложенные операторы switch в С

Оператор switch может находиться в теле внешнего по отношению к нему оператора switch. Операторы case внутреннего и внешнего switch могут иметь одинаковые константы, в этом случае они не конфликтуют между собой. Например, следующий фрагмент программы вполне работоспособен:

```
switch (x)
        {
case   1: switch(y)
        {
case   0:
printf ("Деление на нуль \n");
        break;
case   1:   process(x,y);
        break;
        }
        break;
case   2:
        ...
        }
```

В СОДЕРЖАНИЕ

Оператор цикла while…do... в Pascal, Delphi

Оператор цикла является важнейшим оператором и имеется в большинстве со временных языков программирования. Цикл позволяет многократно выполнить некоторое множество действий, задаваемых операторами, составляющими его тело. В Паскале имеется несколько разновидностей оператора цикла. Начнем с оператора *цикла с предусловием.* Данный оператор имеет вид:

While <условие> Do <команды>;

При выполнении этого оператора вначале вычисляется значение логического выражения <условие>. Если это значение истинно, выполняется оператор <команды>, затем значение выражения проверяется вновь и т. д., до тех пор, пока выражение не примет значение «ложь». Если выражение принимает значение «ложь» при первой же проверке, то оператор <команды> не выполняется вообще. Особо отметим частный случай:

while True do <команды>;

Здесь оператор <команды> будет выполняться бесконечно.

Пример оператора цикла с предусловием:

```
While Counter<10 Do begin
Write('Значение счетчика равно' ,Counter);
Writeln;
Counter:=Counter+2;
end;
```

В СОДЕРЖАНИЕ

Оператор цикла **repeat...until...** в **Pascal, Delphi**

Оператор *цикла с постусловием* имеет вид
repeat <команды> until <условие> ;
Здесь вначале выполняется оператор <команды>, а затем вычисляется значение логического выражения <условие>. Процесс повторяется, пока выражение <условие> принимает значение «ложь». Как только это значение станет истинным, выполнение цикла прекращается. Оператор <команды> может быть любым, в том числе и составным оператором:

```
begin
<команды-1>;
<команды-2>;
<команды-3>;
End;
```

В цикле repeat...until... операторные скобки begin...end могут быть опущены. Таким образом, в общем случае оператор repeat...until... имеет следующий вид:

```
repeat <команды-1>;
  <команды-2>;
  <команды-3>;
until <условие> ;
```

Точка с занятой перед зарезервированным словом until необязательна. В приведенном ниже частном случае

```
repeat <команды-1>;
  <команды-2>;
  <команды-3>;
Until False ;
```

цикл выполняется бесконечно. Еще раз обратим внимание на то, что если в операторе while...do... проверка выполняется в начале цикла, то в цикле repeat…until…. проверка выполняется в последнюю очередь, и тело цикла *в любом случае* выполняется *хотя бы один раз.*

Вот пример цикла с постусловием:

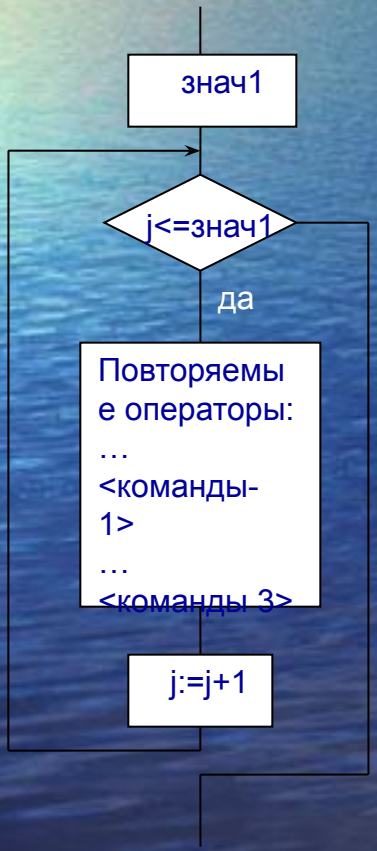
```
repeat
write('значение счетчика равно', Counter);
writeln;
Counter:=Counter+2;
until Counter=10 ;
```

В СОДЕРЖАНИЕ

Оператор цикла for...to...do... в Pascal, Delphi

Третий вариант оператора цикла — *цикл со счетчиком.* Можно считать, что есть две очень похожих друг на друга разновидности цикла со счетчиком. Первый из этих операторов имеет вид

For j:=<значение 1> To <значение 2> Do <команды>



Здесь переменная j, называемая управляющей переменной цикла for, является произвольным идентификатором, который объявляется как переменная любого скалярного типа (к скалярным относятся целый, символьный, булев и перечислимые типы).

При выполнении оператора for сначала вычисляется значение выражения <значение 1> затем вычисляется значение выражения <значение 2>, далее управляющая переменная цикла последовательно пробегает все значения от <значение 1> **до** <значение 2>. В том случае, когда значение <значение 1> оказывается больше значения <значение 2>, тело цикла не будет выполняться вовсе. Эти значения остаются неизменными в ходе выполнения всего цикла for.

СОДЕРЖАНИЕ

Оператор цикла for...downto...do... в Pascal, Delphi

Вариант for...downto...do... цикла for аналогичен циклу for…to...do... за исключением того, что в нем управляющая переменная на каждом шаге выполнения не увеличивается, а *уменьшается* на единицу:

for j := <значение 1> downto <значение 2> do <команды>;

Подводя итоги, **для применения циклов можно сформулировать следующие рекомендации:**

Используйте цикл for в том случае, когда точно знаете, сколько раз должно быть выполнено тело цикла. В противном случае обратитесь к циклам repeat или while

Используйте repeat, если необходимо, чтобы тело цикла выполнялось по крайней мере один раз.

Используйте while, если хотите, чтобы проверка была произведена, прежде, чем будет выполняться тело цикла.

Иногда бывает удобно проводить проверку на возможный выход из цикла где-нибудь в его середине, а не в начале или в конце. Такой выход из цикла обеспечивается процедурой break, которая прерывает выполнение самого внутреннего вложенного цикла, будь то for, while или repeat.

Пример:

```
While true do
Begin
Statement1;
If expression then break;
Statement2;
End;
```

Следует также упомянуть процедуру continue , которая прерывает выполнение тела самого внутреннего цикла for, while или repeat и передает управление на его заголовок, так что начинается выполнение очередной итерации цикла.

СОДЕРЖАНИЕ

Цикл for в С

Во всех процедурных языках программирования циклы for очень похожи, в С этот цикл особенно гибкий и мощный. Общая форма оператора for следующая:
for *(инициализация; условие; приращение) оператор;*
Цикл for может иметь большое количество вариаций. В наиболее общем виде принцип его работы следующий. *Инициализация* — это присваивание начального значения переменной, которая называется параметром цикла. *Условие* представляет собой выражение, определяющее, следует ли выполнять *оператор* цикла (часто его называют *телом цикла)* в очередной раз. Оператор *приращение* осуществляет изменение параметра цикла при каждой итерации. Эти три оператора (они называются также *секциями* оператора for) обязательно разделяются точкой с запятой. Цикл for выполняется, если выражение *условие* принимает значение ИСТИНА. Если оно хотя бы раз примет значение ЛОЖЬ, то программа выходит из цикла и выполняется оператор, следующий за телом цикла for.
В следующем примере в цикле for выводятся на экран числа от 1 до 100:

```
#include <stdio.h>
int main(void)
{
int  x;
for(x=1;  x  <= 100; x++)
printf("%d", x);
return  0;
}
```

В этом примере параметр цикла x инициализирован числом 1, а затем при каждой итерации сравнивается с числом 100. Пока переменная x меньше 100, вызывается функция printf () и цикл повторяется. При этом x увеличивается на 1 и опять проверяется условие цикла x <= 100. Процесс повторяется, пока переменная x не станет больше 100. После этого процесс выходит из цикла, а управление передается оператору, следующему за ним.

СОДЕРЖАНИЕ

Варианты цикла for в C

В предыдущем разделе рассмотрена наиболее общая форма цикла for. Однако в языке С допускаются некоторые его варианты, позволяющие во многих случаях увеличить мощность и гибкость программы. Один из распространенных способов усиления мощности цикла for — применение оператора "запятая" для создания двух параметров цикла. Оператор "запятая" связывает несколько выражений, заставляя их выполняться вместе. В следующем примере обе переменные (x и y) являются параметрами цикла for и обе инициализируются в этом цикле:

```
for(x=0,   y=0;   x+y  <   10;   ++x)
                    {
                  y =  getchar();
y =  y -   '0';   /*  Вычитание из  y   ASCII-кода  нуля  */
                  ……
                    }
```

Здесь запятая разделяет два оператора инициализации. При каждой итерации значение переменной x увеличивается, а значение y вводится с клавиатуры. Для выполнения итерации как x, так и y должны иметь определенное значение. Несмотря на то, что значение y вводится с клавиатуры, оно должно быть инициализировано таким образом, чтобы выполнилось условие цикла при первой итерации. Если y не инициализировать, то оно может случайно оказаться таким, что условие цикла примет значение ЛОЖЬ, тело цикла не будет выполнено ни разу.

СОДЕРЖАНИЕ

Бесконечный цикл в С

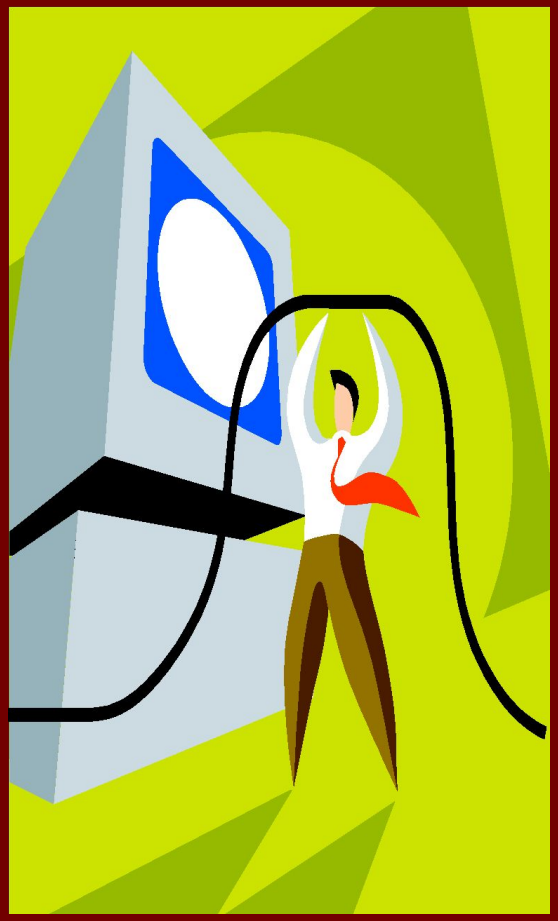
Для создания бесконечного цикла можно использовать любой оператор цикла, но чаше всего для этого выбирают оператор for. Так как в операторе for может отсутствовать любая секция, бесконечный цикл проще всего сделать, оставив пустыми все секции. Это хорошо показано в следующем примере:

for(; ;) printf(" Этот цикл крутится бесконечно ");

Если условие цикла for отсутствует, то предполагается, что его значение — ИСТИНА. В оператор for можно добавить выражения инициализации и приращения, хотя обычно для создания бесконечного цикла используют конструкцию for (; ;). Фактически конструкция for (; ;) не гарантирует бесконечность итераций, потому что в нем может встретиться оператор break, вызывающий немедленный выход из цикла. В этом случае выполнение программы продолжается с оператора, следующего за закрывающейся фигурной скобкой цикла for:

```
ch = ' \0';
for (; ;)
{
ch = getchar(); /* считывание символа */
if (ch == 'A') break; /* выход из цикла */
}
printf(" Вы напечатали 'A' ");
```

В данном примере цикл выполняется до тех пор, пока пользователь не введет с клавиатуры символ a.



СОДЕРЖАНИЕ

Цикл for без тела цикла в С

Следует учесть, что оператор может быть пустым. Это значит, что тело цикла for (или любого другого цикла) также может быть пустым. Такую особенность цикла for можно использовать для упрощения некоторых программ, а также в циклах, предна значенных для того, чтобы отложить выполнение последующей части программы на некоторое время.

Программисту иногда приходится решать задачу удаления пробелов из входного потока. Допустим, программа, работающая с базой данных, обрабатывает запрос "показать все балансы меньше 400". База данных требует представления каждого слова отдельно, без пробелов, т.е. обработчик распознает слово "показать", но не " показать". В следующем примере цикл for удаляет начальные пробелы в строке

*for(; *str==' '; str++);*

В этом примере указатель str переставляется на первый символ, не являющийся пробелом. Цикл не имеет тела, так как в нем нет необходимости.

Иногда возникает необходимость отложить выполнение последующей части программы на определенное время. Это можно сделать с помощью цикла for следующим образом:

for(t =0; t < SOME_VALUE; t++);

Единственное назначение этого цикла — задержка выполнения последующей части программы. Однако следует иметь в виду, что компилятор может оптимизировать объектный код таким образом, что пропустит этот цикл вообще, поскольку он не вы полняет никаких действий, тогда желаемой задержки выполнения последующей части программы не произойдет.

СОДЕРЖАНИЕ

Цикл while в С

Общая форма цикла while имеет следующий вид:
while *(условие) оператор;*
Здесь *оператор* (тело цикла) может быть пустым оператором, единственным оператором или блоком. *Условие* (управляющее выражение) может быть любым допустимым в языке выражением. *Условие* считается истинным, если значение выражения не равно нулю, а *оператор* выполняется, если условие принимает значение ИСТИНА. Если условие принимает значение ЛОЖЬ, программа выходит из цикла и выполняется следующий за циклом оператор.

Как и в цикле for, в цикле while условие проверяется перед началом итерации. Это значит, что если условие ложно, тело цикла не будет выполнено. Благодаря этому нет необходимости вводить в программу отдельное условие перед циклом.

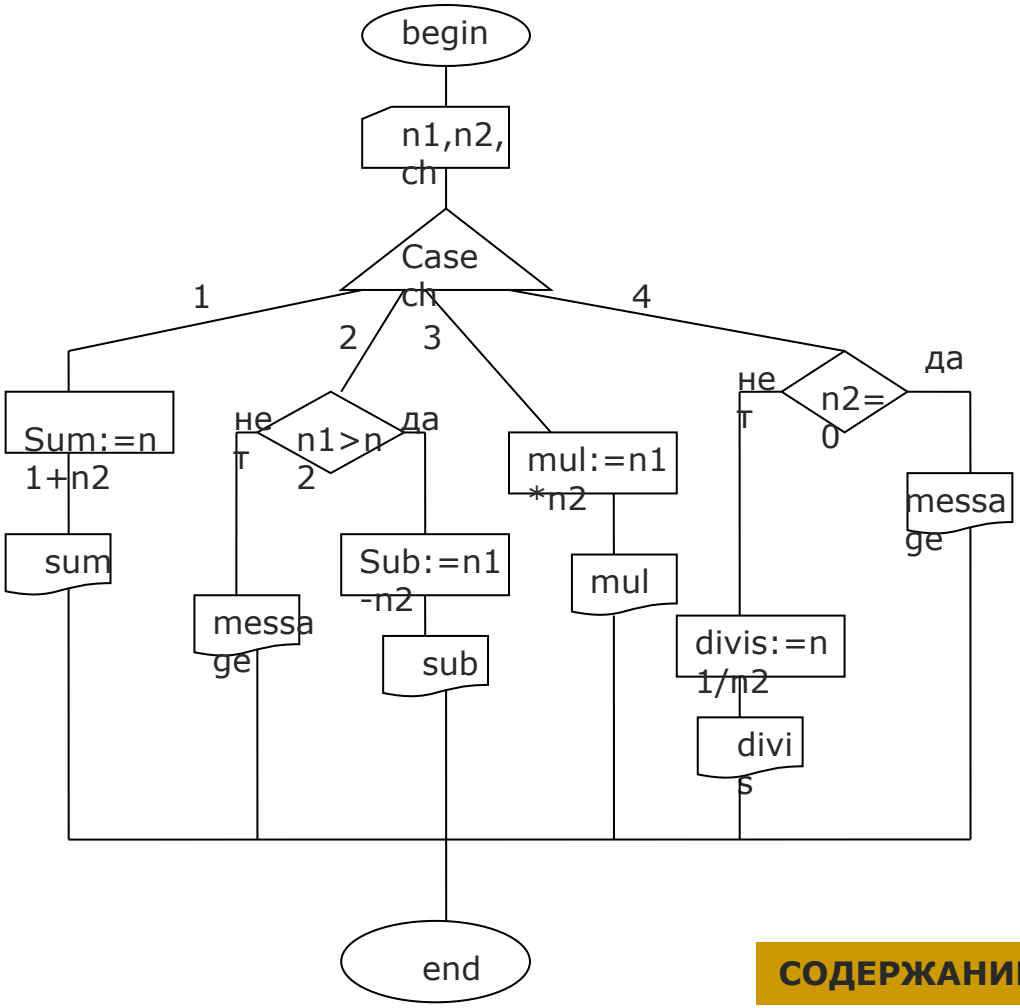
СОДЕРЖАНИЕ

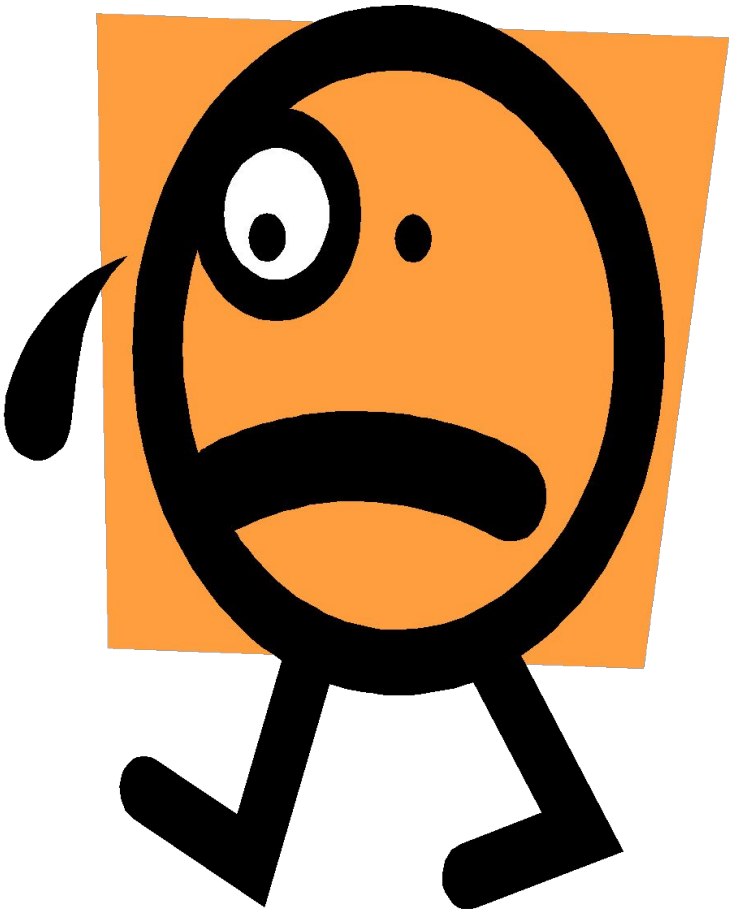
Пример решения задачи в описываемых языках

Условие: После запуска программы на экран выводятся меню:
1 – add;
2- substract
3-multiplie
4- divide
Затем на экран выводится просьба ввести первое число, а затем и второе.
После успешного ввода чисел выводится просьба ввести номер операции, которую вы хотите произвести.
Если вы ввели «1»- то компьютер выведет на экран сумму двух чисел
Если вы ввели «2»- то если первое введенное вами число больше, чем второе, то на экран выведется разность, иначе выведется сообщение о том, что первое число меньше второго и вычитание произвестись не может
Если вы ввели «3»- то выведет на экран произведение
Если вы ввели «4»- то если второе число не ноль, на экран выведется частное двух чисел (первое/второе), иначе на экран выведется надпись что знаменатель =0

СОДЕРЖАНИЕ

БЛОК - СХЕМА





СОДЕРЖАНИЕ

Для C++

```
#include <iostream>
using namespace std;
int main()
{
int ch, n1, n2,sum, sub, mul;
double divis;
cout<<"1- + \n";
cout<<"2- - \n";
cout<<"3- * \n";
cout<<"4- / \n";
cout<<"\n\n";
cout<<"enter the first number \n";
cin>>n1;
cout<<"enter the second number \n";
cin>>n2;
cout<<"enter the operation";
cin>>ch;
```

```
switch (ch)
{
case 1:  /*add numbers*/
sum=n1+n2;
cout<<"summ== "<<sum;
break;
case 2: /*subtraction*/
if (n1>n2)
{
sub=n1-n2;
cout<<"substraction= "<<sub;
}
else cout<<"the first number is less than the second!!! change
them!!\n ";
break;
case 3: /*multiplie*/
mul=n1*n2;
cout<<"multiplication= "<<mul;
break;
case 4: /*division*/
if (n2==0) cout<<"denominator==0!!!! error!!\n";
else
{
divis=n1/n2;
cout<<"division= "<<divis;
}
break;
}
return 0;
}
```

СОДЕРЖАНИЕ

Для Pascal, Delphi

```
program Project1;
{$APPTYPE CONSOLE}
uses
SysUtils;
var
ch, n1, n2,sum, sub, mul:integer;
divis:double;
begin
{ TODO -oUser -cConsole Main :
Insert code here }
writeln ('1- + ');
writeln ('2- - ');
writeln ('3- * ');
writeln ('4- / ');
writeln('enter the first number ');
readln(n1);
writeln('enter the second number
');
readln(n2);
writeln('enter the number of
operation ');
readln(ch);
case (ch) of
1:
begin
sum:=n1+n2;
writeln(sum);
end;
2:
begin
if (n1>n2) then
begin
sub:=n1-n2;
writeln(sub);
end
else writeln('the first number is less than the second!!!
change them!!');
end;
3: begin
mul:=n1*n2;
writeln(mul);
end;
4: begin
if (n2=0) then writeln('denominator==0!!!! error!!')
else begin
divis:=n1/n2;
writeln(divis);
end;
end;
end;
readln;
end.
```

СОДЕРЖАНИЕ

Для Visual basic

```
Imports System
Module MyModule
    Sub Main()
Dim ch As Integer
Dim n1,n2,sum,sub,mul As
    Integer
    Dim divis As double
  Console.WriteLine("1- +")
 Console.WriteLine("2- –")
 Console.WriteLine("3- *")
 Console.WriteLine("4- /")
 Console.WriteLine("enter the
    first number ")
 N1 = CInt(Console.ReadLine)
    Console.WriteLine()
        Console.WriteLine
("enter the second number ")
 N2 = CInt(Console.ReadLine)
    Console.WriteLine()
 Console.WriteLine("enter the
number of the operation”)
 ch = CInt(Console.ReadLine)
    Console.WriteLine()
```

СОДЕРЖАНИЕ

```
Select Case ch
    Case 1
        sum=n1+n2
Console.WriteLine ("Sum is: {0}",  sum)
    Case 2
  if (n1>n2) then
        sub=n1-n2
 Console.WriteLine ("substraction is:
{0}”,sub)
        end if
else Console.WriteLine ("the first number is less
than the second!!! change them!!")
    Case 3
        mul=n1*n2
Console.WriteLine ("multiplication is {0}",mul)
    Case 4
  if (n2==0) then
Console.WriteLine ("denominator ==0!!!! error!!")
        else
        divis=n1/n2
 Console.WriteLine ("division is
{0}",divis)

End Select
  End Sub
End Module
```

Вот и все!!!

ВЫХОД ЗДЕСЬ