**Операторы**

Оператор if

If – управляющий оператор, который позволяет изменить порядок выполнения инструкций к программе. Синтаксис:

**if (выражение)**

**{**

**оператор 1;**

**}**

**[ else**

**оператор 2; ]**

После *if (выражение*) ; не ставится иначе компилятор не замечает этой ошибки и стоящий далее оператор выполняется вне зависимости от (выражения). Выражение может быть любым логическим или выражением целого типа. Считается что если значение выражения ≠0 то это истина. При использовании оператораif нужно помнить, что else относится к ближайшему *if.*

Пример программы применения оператора *if* для выявления наибольшей величины*.*

#include <stdio.h>

int x,y;

int main()

{

printf(“\n Input x :”);

scanf(“ %d ”, & x);

printf(“\n Input y :”);

scanf(“ %d ”, & y);

if (x==y)

printf(“ x is equal to y \n ”);

else

if (x>y)

printf(“x is greater then y \n”);

else

printf(“y is greater then x \n”);

return 0;

}

Примечание: Функция scanf считывает данные с клавиатуры в соответствии с форматом (% формат) &- операция взятия адреса передаваемой числовой переменной.

Пример программы нахождения значения функции f(x) по заданному значению x

Sin x, x>0

F(x) =

cos x, x≤0

#include <stdio.h>

float x,y;

int main()

{

printf (“\n Input x :”);

scanf (“ %f ”, & x);

if (x>0) f=sin(x);

else f=cos(x);

printf(“F(%.2f)=%.2f \n”, x, y);

system (“PAUSE”);

return 0;

}

Пример программы нахождения корней квадратного уравнения

#include <stdio.h>

Int a, b, c;

float x1, x2, d;

int main()

{

printf(“\n Input a :”);

scanf(“ %d ”, & a);

printf(“\n Input b :”);

scanf(“ %d ”, & b);

printf(“\n Input c :”);

scanf(“ %d ”, & c);

d= b\*b-4\*a\*c;

if (d<0)

printf(“\n Korney net”);

else

{

If (d==0)

{ x1=x2=-b/(2\*a)};

Else

{

x1=(-b+sqrt(d))/(2\*a);

x2=(-b-sqrt(d))/(2\*a);}

printf(“\n x1=%f, x2=%f”, x1,x2);

}

System(“PAUSE”);

Return 0;

}

Переключатель SWITCH

Он аналогичен оператору выбора варианта в языке Паскаль. Синтаксис оператора

**switch (выражение)**

**{**

**case конст1: оператор 1;**

**case конст2: оператор 2;**

**……….**

**[default : операторы ;]**

**}**

Выражение должно принимать целочисленное значение типа char, long или int. Оператор вычисляет значение выражения и сравнивает значение с константами. Затем выполняется один из операторов. Если значение не совпадает ни с одной из констант, выполняется оператор после ключевого слова **default.** Оператор будет выполняться до тех пор пока не встретит **break.**

Пример программы использующей переключатель **switch** для организации структуры меню:

#include <stdio.h>

Int reply;

int main()

{

printf(“\n Enter a number 1 - 5 :”);

scanf(“ %d ”, & reply);

switch (reply);

{

case 0: break;

case 1:

{

puts(“You entered 1. \n”);

break;

}

case 2:

{

puts(“You entered 2. \n”);

break;

}

case 3:

{

puts(“You entered 3. \n”);

break;

}

case 4:

{

puts(“You entered 4. \n”);

break;

}

case 5:

{

puts(“You entered 5. \n”);

break;

}

default:

{

Puts(“Out of range. \n”);

}

} /\* конец switch\*/

return 0;

}

Оператор цикла while

Выполняет блок операторов до тех пор пока некоторое условие остаётся истинным.

Синтаксис:

**while (условие повторения)**

**{**

**тело цикла;**

**}**

Сначала проверяются значение выражения *УсловиеПовторения.* Если оно не равно нулю, т.е. условие истинно, то выполняются инструкции цикла (тело цикла). Затем снова проверяются значение выражения *УсловиеПовторения,* и если оно не равно нулю, инструкции цикла выполняются еще раз. И так до тех пор, пока значение выражения *УсловиеПовторения*  не станет равным нулю. Т.е. если условие ложно выполнение оператора while прекращается.

int x=0;

while (x<10)

{

printf(“\n X is %d”, x);

x++;

}

Вычисление y=n!

int y=1;

int k=1;

while (k<n)

{

y=y\*k;

k++;

}

Оператор цикла do while

Синтаксис:

**do**

**{**

**тело цикла;**

**}**

**while (условие повторения);**

Сначала выполняются инструкции цикла (тело цикла), затем проверяется значение выражения *УсловиеПовторения*, и если условие истинно, не равно нулю, то инструкции цикла выполняются еще раз. И так до тех пор, пока *УсловиеПовторения* не станет ложным, т.е. равным нулю.

Пример: отрывок программы которая вводит числа пока не встретится число большее 99.

int b;

do

{

scanf(“%d”, &b);

}

while (b<=99);

Оператор цикла for

Синтаксис:

**For (инициализация; условие выполнения; изменение)**

**{**

**операторы;**

**}**

*Инициализация* – инструкция инициализации счетчика циклов. Является оператором присваивания, который устанавливает значение некоторой переменной (вычисляется 1 раз в начале выполнения цикла)

*Условия выполнения* – выражение, значение которого определяет условие выполнения инструкций цикла. Инструкции цикла выполняются до тех пор, пока *УсловиеВыполнения* истинно, т.е. не равно нулю. Если условие истинно, то выполняется операторы в цикла, алее вычисляется выражение изменения (приращения) и выполняется снова вычисление условия.

*Изменение –* инструкция изменения параметра цикла. Как правило, эта инструкция изменяет значение переменной, которая водит в *УсловиеВыполнения .*

(n!) p=1; for (i=1; i<=n; i++) p=p\*i;

Любая из частей оператора может отсутствовать, но наличие ; обязательно

***for ( ; ; )***

Для преждевременного завершения оператора for можно использовать оператор break

Оператор for часто используется для отсчёта прогона циклов путём увеличентя переменной счетчика

for (a=1; a<=20; a++) printf(“%d\n”, a);

for (a=100; a>0; a--) printf(“%d\n”, a);

for (a=1; a<=20; a+=5) printf(“%d\n”, a);

Использование без операции инициализации

c=1;

for ( ; c<10; c++)

Использование без операции приращения

for (c=1; c<10; )

printf(“%d“, c++);

Оператор continue

Он может находиться только в теле цикла for, while, do… while. Выполнение оператора приводит к тому, что немедленно начинается следующая итерация цикла. Операторы, стоящие в теле цикла между continue и } игнорируются.

**while(…)**

**{**

**…**

**continue;**

**…**

**break;**

**…**

**}**

Оператор безусловного перехода goto

Переход осуществляется к оператору отмеченному меткой. Метка может стоять как в отдельной строке, так и в начале строки содержащей оператор. Метка дролжна быть уникальна и не может повторяться. Оператор goto и метка должны находиться в одной функции. Метка отделяется от оператора двоеточием.

Выполнение команд операционной системы

System() – выполняет команды операционной системы прямо из работающей программы. Для использования этой функции программа должна включать файл stdlib.h.

Формат команды:

**System(команда);**

Команда может быть строковой константой или указателем на строку

system(“dir c:”)

или

сhar \*command=”dir c:”;

system (command);

После выполнения команды ОС, управление передается в программу в то же место, откуда был сделан вызов.