**Лабораторная работа Программирование с использованием ветвлений**

**Цель работы:** изучение операторов организации ветвления, приемов создания программ, обеспечивающих выполнение альтернативных действий

**Объем работы:**2 часа

**Теоретическая часть**

*Ветвлением* в программировании называют конструкцию обеспечивающую выполнение различных действий в зависимости от результатов проверки некоторого условия.

Для программирования ветвлений, т. е. ситуаций, когда возникает необходимость при выполнении условия реализовывать одни действия, а при нарушении – другие, используют оператор условной передачи управления. Условие записывают в виде выражения, заключенного в круглые скобки. В зависимости от результата этого выражения осуществляется выбор одной из ветвей: если результат выражения «не ноль», то выполняется оператор, следующий за условным выражением, иначе – оператор, следующий за служебным словом else.

Синтаксис оператора ветвления:

**if (<Выражение>) <Оператор;> [ else <Оператор;>]**

где:

**Оператор**– любой оператор С++.

**Выражение** – любое выражение, соответствующее правилам составления выражений С++.

В каждой ветви допускается запись одного оператора, в том числе и другого условного оператора.

Если по алгоритму решения задачи необходимо в ветвях размещать более одного оператора, используют составные операторы.

*Составным оператором* в С++ называют последовательность операторов, заключенную в операторные скобки ( {……}). Операторы последовательности отделяют друг от друга точкой с запятой. Перед закрывающей скобкой (}) блока, точку с запятой также обязательно ставить. Так как точка с запятой является неотъемлемой частью оператора, то перед else точка с запятой обязательна. Исключением является составной оператор, так как в нем точка с запятой уходит внутрь блока.

В соответствии с синтаксисом оператора ветвления, допускается использовать оператор условной передачи управления с пропущенной (пустой) ветвью else. Такой оператор называется укороченным.

Так как в качестве оператора ветви по синтаксису может быть использован условный оператор, как в полной, так и укороченной форме, то при реализации такой конструкции могут возникнуть неоднозначности. Для этого необходимо использовать правило вложения. Это правило гласит:**ветвь else всегда относится к ближайшему if.**

Правило вложения продемонстрируем на примере нижеприведенных фрагментов алгоритмов:



**Рисунок 1 –**Варианты алгоритмов неоднозначных ветвлений

Следуя правилу вложения, для реализации варианта *а* подойдет следующий фрагмент программы на языке С++:

**if** <Условие1> **then**

**if** <Условие2> **then** <Действие1>

**else** <Действие 2>

Для реализации варианта *б* следует использовать составной оператор {…}:

**if** <Условие1>

**{**

**if** <Условие2> <Действие1>

}

**else** <Действие 2>

В этом случае укороченный вложенный **if**войдет внутрь составного оператора, а **else**,по правилам вложения, будет отнесен к внешнему**if**, так как он в этом случае будет ближайшим.

**Пример 1.** Разработать программу, которая вычисляет значение функции, заданной следующим образом:

ì |x|, при |x| £ 1;

y = í x2, при 1 < |x| £ 2;

î 4, иначе.

Программа должна начинаться с ввода значения аргумента. Затем проверяем введенное значение, и в зависимости от того, в какой интервал оно попадает, вычисляем значение функции по одному из заданных выражений. Для того чтобы проверить попадание в три указанных диапазона необходимо выполнить две проверки.

Алгоритм решения данной задачи представлен на рис. 2.

Рисунок 2 - Схема алгоритма программы вычисления функции, заданной на отрезках, в заданной точке

Текст программы имеет следующий вид:

#include "stdafx.h"

#include <stdio.h>

#include <math.h>

int main()

{

float x,y;

puts("Input x: ");

scanf("%f",&x);

if(fabs(x)<=1)

{

y=fabs(x);

puts("y=|x|");

}

else

if(fabs(x)<=2)

{

y=x\*x;

puts("y=x^2");

}

else

{

y=4;

puts("y=4");

}

printf("For x=%6.2f y=%7.2f\n",x,y);

return 0;

}

**Пример 2.** Разработать программу, которая определяет можно ли из трех отрезков, представленных действительными числами x,y,z, вводимыми с клавиатуры, построить треугольник. Как известно, основные правила существования треугольника следующие: ни одна сторона не должна быть равна нулю, а суммы двух сторон должна быть больше третьей. Таким образом, нам нужно проверить шесть условий:

x, y, z не равны 0;

x+y>z; y+z>x; x+z>y.

Для реализации проверок следует использовать операторы условной передачи управления.

Текст программы приведен ниже.

#include "stdafx.h"

#include <stdio.h>

#include <math.h>

int main()

{float x,y,z;

puts("Input x,y,z: ");

scanf("%f %f %f",&x,&y,&z);

printf(" x=%6.2f y=%7.2f z=%7.2f\n",x,y,z);

if((x!=0)&&(y!=0)&&(z!=0))

if(((x+y)>z)&&((y+z)>x)&&((z+x)>y))

puts("Treangle is exzist");

else

{

puts("Treangle no exzist, one side > summy other side");

printf(" x=%6.2f y=%7.2f z=%7.2f\n",x,y,z);

}

else

{ puts("Treangle no exzist, is 0 side");

printf(" x=%6.2f y=%7.2f z=%7.2f\n",x,y,z);

}

return 0;

}

Другие примеры программ, содержащих ветвления, приведены в [1].

**Порядок выполнения работы**

1. Прочитать и проанализировать задание в соответствии со своим вариантом.

2. Разработать схему алгоритма решения задачи.

3. Написать программу.

4. Вызвать среду программирования Visual Studio 2008 , создать новый проект консольного приложения и ввести текст программы в редактор среды программирования.

5. Подобрать тестовые данные (не менее 3-х вариантов).

6. Отладить программу на выбранных тестовых данных.

7. Продемонстрировать работу программы преподавателю.

8. Составить отчет по лабораторной работе.

9. Защитить лабораторную работу преподавателю, ответив на контрольные вопросы.

**Требования к отчету**

Отчет должен быть выполнен на бумаге формата А4 или А5 в том числе в тетрадях или на тетрадных листах. Если отчет выполняется на отдельных тетрадных листах, то они должны быть аккуратно обрезаны по линии подшивки и скреплены. Неаккуратно выполненные, оборванные или грязные отчеты не принимаются.

Все записи в отчете должны быть либо напечатаны на принтере, либо разборчиво выполнены от руки синей или черной ручкой (карандаш – не допускается). Схемы также должны быть напечатаны при помощи компьютера или нарисованы с использованием чертежных инструментов, в том числе карандаша.

Каждый отчет должен иметь титульный лист, на котором указывается:

а) наименование факультета и кафедры;

б) название дисциплины;

в) номер и тема лабораторной работы;

г) фамилия преподавателя, ведущего занятия;

д) фамилия, имя и номер группы студента;

е) номер варианта задания.

Кроме того, отчет по лабораторной работе должен содержать**:**

1) схему алгоритма, выполненную вручную или в соответствующем пакете;

2) текст программы;

3) результаты тестирования, которые должны быть оформлены в виде таблицы вида:

|  |  |  |
| --- | --- | --- |
| Исходные данные | Ожидаемый результат | Полученный результат    |
|  |  |  |

Задание каждому – ориентируясь на примеры составить свои простейшие программы. В отчете вариант не указывать, темы лабораторных работ согласно учебного плана:

1. **Составление простейших программ на языке Си++**
2. **Составление программ с использованием ветвлений**

В теоретической части в примерах довольно много материалов. Если возникнут вопросы по программному обеспечению – пишите мне на почту vas2498@yandex.ru - ,будем думать, как решить.