Лекция №6. Элементы языка программирования С++. Решение задач с помощъю линейных алгоритмов.

План:

1. **Элементы языка программирования С++.**
2. **Решение задач с помощъю линейных алгоритмов.**

1. Элементы языка программирования С++.

Язык программирования С ++ был разработан в 1979 году Бьярном Страуструпом из исследовательского центра AT & T Bell Laboratories в Нью-Джерси, США.

Рисунок 1.1. Бярн Страуструп - программист, создавший язык программирования С ++.

Название языка программирования С++ происходит от оператора ++ (инкремент), который увеличивает на 1 значения переменной в языке программирования С.

Dev-C++ (Dev-Cpp) - одна из программ, обрабатывающих программы на языке программирования С ++.

Окно программы Dev-C ++ выглядит так:

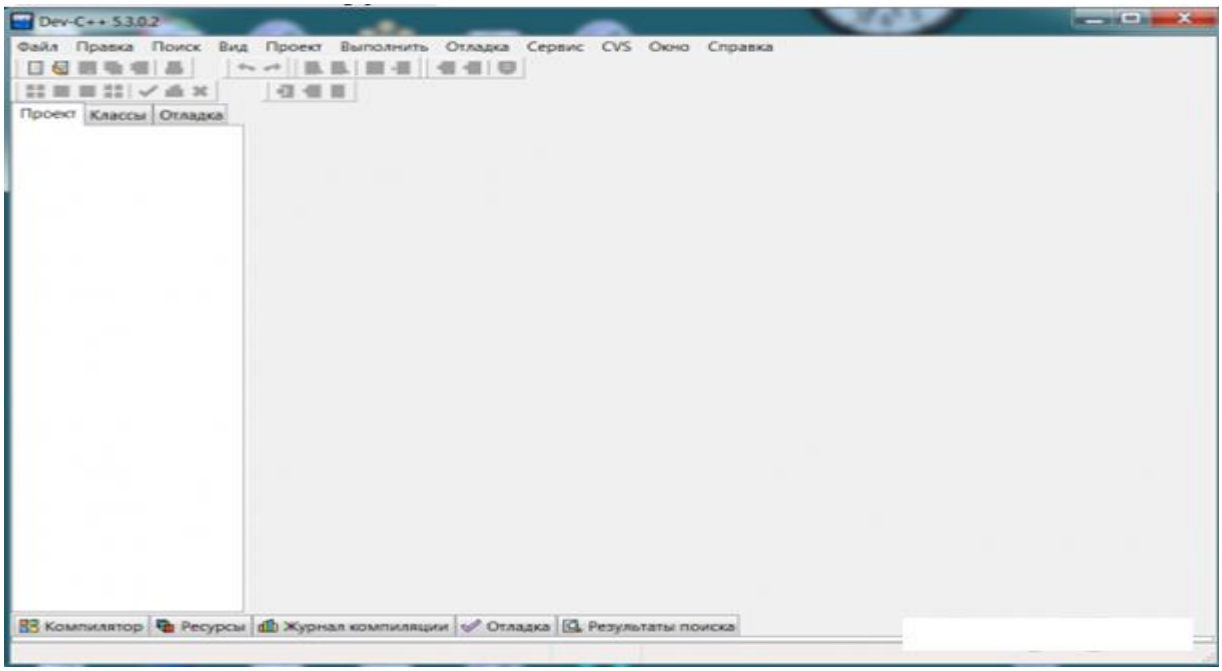

Рисунок 1.2. Структура окна программы Dev-C ++.

В Dev-C ++ можете выполнять следующее операции:

Ctrl + O - отобразить файл в памяти;

Прежде чем вы сможете запустить программу на языке программирования C ++, вам нужно скомпилировать ее. Для этого выполните следующие действия:

- Скомпилировать (F9) - простая компиляция программы. На этом этапе компилятор проверяет наличие ошибок в программе. Если все правильно, программа отправит код в файл в формате * .exe. Если есть ошибки, то компилятор перестанет работать, и код ошибки будет отображен в окне «Компилятор». Этот код может быть использован для исправления ошибки.

- Выполнить (F10) - эта команда позволяет запустить программу без компиляции.

- Скомпилировать и выполнить (F11) - компилирует программу и запускает ее немедленно.

6.2. Алфавит, идентификатор, служебные слова.

Алфавит. Алфавит C ++ включает в себя следующие символы:

• Прописные и строчные латинские буквы (A, B, .., Z, a, b,…, z)

• Числа: 0,1,2,3,4,5,6,7,8,9

• Специальные символы: «, {} | [] () + - /% \; «. :? <=> _! & * # ~ ^

• Невидимые символы («пробелы»).

Другие символы, такие как русские буквы, могут использоваться в комментариях, строках и символьных переменных.

Идентификаторы. Идентификаторы состоят из последовательности латинских букв, символов подчеркивания и цифр. Идентификатор должен начинаться с латинской буквы или подчеркивания.

Например:

A1, _MAX, _01, RIM, rim.

Прописные и строчные буквы различаются, поэтому последние два идентификатора различны.

Служебные слова. Идентификаторы, которые не могут использоваться переменными в качестве имен переменных, называются служебными словами.

Следующие служебные слова доступны в C ++:

int, extern, else, char, register, for, float, тип, def, do, double, static, while, struct, goto, switch, union, return, case, long, sizeof, default, short, break, entry, без знака, продолжить, авто, если и так далее.

Alfavit, identifikator.

6.3. Переменные.

Одна из основных понятий в C ++ - это поименованная часть памяти - объект. Один из разновидностей объекта - это переменная. Когда переменной присваивается значение, код значения записывается в часть памяти, выделенной для нее. На значение переменной можно ссылаться по имени, а на часть памяти можно ссылаться только по адресу. Имя переменной является идентификатором. Служебные слова не могут использоваться в качестве имен переменных.

Существуют следующие типы переменных:

char - один символ;

long char - длинный символ;

int - целое число;

short - короткое целое число;

длинный - длинный целый конец;

float - реальное число;

double - двойное действительное число;

long double - длинное двойное вещественное число;

Переменные могут быть определены или переопределены в любой части программы.

Например:

Int a, b1, ac;

Int a;

int b1;

int ac;

Когда переменные определены, их значения могут быть не определены. Но при определении переменных можно их инициализировать, то есть дать начальные значения.

Например:

Int I = 0;
Char c = 'k';

Константы.

Константа - значение которого не изменяется во время работы программы. В С++ можно использовать пять типов констант: целочисленные, вещественные, символьные переменные и нулевой указатель. Целые числа могут быть в десятичном, восьмеричном или шестнадцатеричном виде. В десятичной системе целые числа состоят из последовательности чисел 0-9, первое число не должно быть 0. В восьмеричной системе счисления целые числа состоят из чисел 0-7, начинается с 0. В шестнадцатеричной системе счисления целое число представляет собой последовательность чисел 0-9, начиная с 0x или 0X и букв A-F. Например, десятичные числа 15 и 22 представлены как 017 и 026 в восьмеричной и 0xF и 0x16 в шестнадцатеричном.

Константы метки.

Один или два символа заключется в апостроф. Например, 'x', '*', '2', '\0', '\n' являются односимвольными переменными; Двухсимвольные переменные: 'dd', '\n\t ','\x07 \x07 '.

Таблица символов ESC (Escape):

Написание	Внутренный код	Символ (имя)	Значение
\a	0x07	bel (audible bell)	Звуковой сигнал
\b	0x08	Bs (bascspase)	Вернуть один шаг
\f	0x0C	Ff (form feed)	Вернуться на страницу
\n	0x0A	lf (line feed)	Пропустить строку
\r	0x0D	Cr (carriage return)	возврат каретки
\t	0x09	Ht (horizontal tab)	Горизонтальная табуляция
\v	0x0B	Vt (vertical tab)	Вертикальная табуляция
\\	0x5C	\ (bacslash)	обратная линия
\'	0x27	' (single out)	Апостроф
\"	0x22	" (double quote)	Кавычки
\?	0x3F	? (question mark)	Знак вопроса

Строковые константы.

Строковые костанты не включены в список констант С ++, но являются отдельным типом. Строковая костанты - это последовательность произвольных символов, заключенная в кавычки. Например, «Строкавая константа». Строковые символы помещаются в память последовательно и в конце каждой строковой переменной компилятор автоматически добавит

символ «\ 0». Размер такой строки в памяти равен на количеству символов + 1 байт.

Строки, которые являются последовательными и разделены пробелами, табуляторами или переносами строк, преобразуются в одну строку во время компиляции. Например:

Строки «Привет» «Ташкент» рассматриваются как одна строка: «Привет, Ташкент».

Строки, написанные в несколько строк, также подпадают под это правило. Например :

"Весна"
"пришла"
"в Узбекистан"
"Весна пришла в Узбекистан".

Именованные константы.

Значения этих констант не могут быть изменены в программе. Имена констант могут быть идентификаторами, введенными программистом, и отличаться от служебных слов. Именованные константы вводятся в следующей форме:

Константный тип constant_name = constant_value.

Например:

Const double EULER = 2,718282;
Const long M = 99999999;
Const R = 765;

В последнем примере тип переменной не указан. Тип констант определяется по их значениям и принадледит типу int.

Нулевой указатель.

NULL - неарифметическая переменная. Нулевой указатель может быть представлен 0 или именованной константой NULL. В следующей таблице показаны границы переменных и соответствующих типов:

Типы данных	Объем, бит	Диапазон значений	Функции типа
Unsigned char	8	0…255	Маленькие целые и символы коды
Char	8	-128…127	Маленькие целые числа и коды ASII
Enum	16	-32768…32767	Последовательность целых чисел
Unsigned int	16	0…65535	Большие целые числа
Short int	16	-32768…32767	Маленькие целые числа, циклы управление
Int	16	-32768…32767	Kichik butun sonlar, sikllarni boshqarish

Unsigned long	32	0…4294967295	Астрономические расстояния
Long	32	-147483648… …2147483647	Большие числа
Float	32	3.4E-32…3.4E+38	Научные расчеты (7 цифр)
Double	64	1.7E-308…1.7E+308	Научные расчеты (15 цифр)
Long double	80	3.4E-4932… 1.1E+4932	Финансовая отчетность (19 цифр)

Операции в С ++

Арифметические операции. Многие программы выполняют арифметические операции во время исполнения. Операции в С ++ приведены в таблице ниже. Они были использованы с двумя операндами.

Операция	Арифметический оператор	Арифметическое выражение	Выражение в С++
Сложение	+	h+19	h+19
Вычитание	-	f-u	f-u
Умножение	*	sl	s*l
Деление	/	v/d	v/d
Деление по модули	%	k mod 4	k%4

Когда делитель и делимое являются целыми числами, результатом является целое число. Дробная часть отбрасывается, а целая часть остается прежней.

Выражение x%y дает остаток при делении x на y. Итак, результат выражения 7% 4 равен 3. Оператор% работает только с целыми числами. Для работы с запятыми (действительными) числами используйте функцию fmod в библиотеке math.h. В С ++ значение круглых скобок такое же, как в математике. Кроме того, последовательность других алгебраических выражений также как обычно. Операторы умножения, деления и извлечения модулей выполняются первыми. Если в строке более одного оператора, они выполняются слева направо. После этих операторов выполняются сложение и вычитание. Например, k = m * 5 + 7% n / (9 + x);

Первым выполняется м * 5, потом выполняется 7%n и полученное остаток делится на (9 + x). Результат добавляется в ответ m * 5.

Таблица операций

Арифметические операции	Операции с разрядами	Операции осравнений	Логические опрации
+ сложение	& и	= = равно	&& и
- вычитание	| или	!= не равно	|| или
* умножение	^ исключающий или	> больше	! отрицание
/ деление	<< сдвиг налево	>= больше или равно	
% деление по модулю	>> сдвиг направо	< меньше	
- унарный минус	~ отрицание	<= меньше или равно	
+ унарный плюс			
++ oshirish			
-- kamaytirish			

Таблица операций

Операций	Операций	Операций	Операций
() – круглая скобка	= - присваивание значений	(tip) – изменение типа	& - определение адреса
[] – квадратные скобки	op= - составное присваивание значений	sizeof- измерение объема	* - определение значение по адресу
, - вергул	? – условная операция		

Арифметические операции. Операции обычно делятся на унарный, то есть операции, которые применяются к одному операнду, и двоичные, то есть операции, которые применяются к двум операндам.

Двоичные операции делятся на аддитивные операции, т.е. + сложение и вычитание, и мультипликативные операции, такие как * умножение, / деление и % деления по модули. Преобладание аддитивных действий ниже, чем преобладание мультипликативных действий.

Когда целое число делится на целый, результат округляется до целого числа. Например, 20/3 = 6; (-20) / 3 = -6; 20 / (- 3) = - 6.

Операция модуля равна остатку от деления целого числа на целое. Если операция модуля применяется к положительным операндам, результат также

является положительным, в противном случае знак результата зависит от компилятора.

При выполнении двоичных арифметических операций типы задаются в соответствии со следующими правилами: типы short и char сводятся к типу int;

Если один из операндов имеет тип long, второй операнд также имеет тип long, а результат имеет тип long;

Если один из операндов принадлежит типу с плавающей точкой, другой операнд также принадлежит типу с плавающей точкой и результат также принадлежит типу с плавающей точкой;

Если один из операндов имеет тип double, второй операнд также имеет тип double, а результат имеет тип double;

Если один из операндов имеет тип long double, другой операнд также имеет тип long double, а результат имеет тип long double;

Унарные операции включают унар минус - и унар +, которые меняют знак. Опреации ++ и – тоже являются унарными.

Операция ++ представляет увеличение значения на 1. Операциия в виде i++ указывает, что сначало используется значение i, потом увеличивается на 1(постфиксный инкремент). Операциия в виде ++ Операциия в виде ++i указывает, что сначало увеличивается на 1 значение i, потом используется(префиксный инкремент). указывает, что сначало используется значение i, потом увеличивается на 1(префиксный инкремент).. Например, если значение i равно 2, то значение выражения 3 + (++ i) равно 6, а значение выражения 3 + i ++ равно 5. В обоих случаях значение i равно 3.

Операция -- означает уменьшение значения на 1. Эта функция также может использоваться в виде префиксов и постфиксов. Эти две операции могут применяться только к переменным. Приоритет унар действий выше, чем бинарных.

Битовые операции. Результатом битовой операции является применение логических операций, соответствующих каждому биту двоичного представления целых чисел. Например, код 5 равен 101, а код 6 равен 110. Значение 6 & 5 равно 4, что составляет 100. Значение 6 | 5 равно 7, что равно 111. Значение 6 ^ 5 равно 3, что составляет 011.

В этих примерах приоритет действий дается в порядке возрастания.

В дополнение к этим операциям M << N перемещается влево, а M >> N перемещается вправо. Сдвиг применяется побитовой форме целого числа M. N указывает, сколько позиций нужно переместить. Перемешение M на позиции N влево означает умножение на 2^N . Например 5 << 2 = 20. Это операция побитовой форме имеет следующий вид: 101 << 2 = 10100.

Операция отношений. Значения операций отношения равны 1, если отношение выполняется, 0 если наоборот. Операции отношения применяются к операндам или указателям арифметического типа.

Примеры:

Значение 1! = 0 равно 1;

Значение 1 == 0 равно 0;

Значение 3> = 3 равно 1;

Значение 3> 3 равно 0;

Значение 2 <= 2 равно 1;

Значение 2 <2 равно 0;

Приоритет операций: больше>, меньше <, больше или равно>=, меньше или равно <=. Преобладание равных и неравных действий одинаково и ниже, чем у остальных.

Логические операции. Результаты этих действий определяются следующим образом:

Мы хотим проверить, что точка (x, y) расположена в 1-й четверти координатной плоскости. Операция x || y равна 1, если выполняется хотя бы одно из условий x> 0 и y> 0, и 0 если не выполняется. x && y равно 1, если x> 0 и y> 0, и 0 если невыполняется. Вэтих примерах приоритет действий дается в порядке возрастания. В дополнение к этим действиям также доступны следующие действия:

Операция присваивания. Операция присваивания = - это двоичная операция, в которой левый операнд обычно является переменной, а правый операнд обычно является выражением. Например, Z = 4,7 + 3,34. Значение этого выражением равно к 8,04. Это значение передается в переменную Z.

В конце этого выражения ставится точка с запятой;

Z = 4,7 + 3,34;

В одном выражении могут быть использованы несколько значений. Например: C = y = f = 4,2 + 2,8;

Существует также сложная операция присваивания в C++, общая форма которой: Variable_name operation = expression;

Например:

Выражение x + = 4 эквивалентно выражению x = x + 4;

Выражение x * = a эквивалентно выражению x = x * a;

Выражение x / = a + b эквивалентно выражению x = x / (a + b);

Выражение x >> = 4 эквивалентно выражению x = x >> 4;

Структура программы.

Программа состоит из команд препроцессора и нескольких функций. Среди этих функций должны быть функция с , называнием main.

Давайте напишем следующую программу:

```
 # include <iostream>
использование пространства имен std;
 пустая функция ()
   # include <iostream>
  using namespace std;
   void main ()
   {
```

```
    cout << "Hello \ n";
}
```

Команда #Include указывает компилятору использовать стандартные потоки ввода и вывода в файле iostream. Без этих описаний операция cout << "Hello \ n" не будет работать. ***namespace std-*** указывают на использование пространства имен. Если эта команда не указана, вам придется ввести слово std в операторы ввода и вывода данных. void main () - это основная функция в программе, называемая main, которая должна быть записана в программе. Каждая программа должна иметь функцию с именем main, и программа запускается с этой функцией. void - указывает тип основной функции, указывая, что функция не возвращает значение. cout << "Hello \ n" - это оператор вывода данных, который печатает слово Hello. Символ \n обозначает переход на новую строку.

Чтобы получить результат программы, написанной на С ++, вам нужно скомпилировать ее. Текст программы читается и анализируется, если ошибок не обнаружено, проверяются имена и действия, используемые в программе (в нашем случае это cout и <<). Если возможно, программа заполнит отсутствующие определения в библиотеке.

Следующая программа попросит вас ввести длину в дюймах. После этого он переводит дюймы в сантиметр.

```
#include< iostream>
using namespace std;
int main()
{
    int inch = 0;        // inch - дюйм
    cout << "dyum";
    cin >> inch;
    cout << inch;
    cout << " in = ";
    cout << inch*2.54;
    cout << " sm\n";
}
```

Первая строка функции main () определяет переменную целого типа *inch* в дюймах. Его значение считывается с помощью операции >> стандартного потока ввода данных *cin*. После запуска программы результат может выглядеть так:

```
dyum
 12
 in = 30,48 см
```

Последние четыре команды печати могут быть записаны одним оператором:

```
cout << inch << "in =" << inch * 2.54 << "sm \ n";
```

Комментарии. Текст, который используется программой только для интерпретации программы и не учитывается компилятором, называется комментарием. В С ++ комментарий пишется двумя способами. В первом методе комментарий начинается с / * и заканчивается * /. Комментарий такого типа используется для многострочных комментариев. // символы используются для однострочных комментариев.

Каждое имя и каждое выражение имеет тип, который определяет операции, которые могут быть выполнены в этом выражение. Например,

int inch;

Здесь определяется, что типом дюйма является ***int***, то есть дюйм является целочисленной переменной. Описание - это оператор, который вводит имя в программу. Описание определяет тип этого имени. Тип определяет правильное использование имени или выражения. Для уелых чисел могут быть выполнены операции, как +, -, * и /.

Основные типы:

byte , short, int, long, float, double,boolean, char

Первые четыре типы используются для представления целых чисел, следующие два используются для представления действительных чисел с плавающей запятой. Тип char используется для представления одного символа, а тип ***boolean*** определяет логический тип. Диапазон целых чисел, которые могут быть представлены с помощью типа, зависит от его размера. Взаимозависимость основных видов может быть записана следующим образом:

1 = sizeof(char) <= sizeof(short) <= sizeof(int) <= sizeof(long)

sizeof(float) <= sizeof(double)

Символ внутри апостроф - это размер символа. Арифметические операции могут применяться к любой комбинации этих типов:

+ (сложение, унарный и бинарный плюс)

- (вычитание, унарный и бинарный минус)

* (умножение)

/ (деление)

Также операции сравнения:

== (равно)

! = (не равно)

<(меньше)

> (больше)

<= (меньше и равно)

> = (больше и равно)

Деление целого числа дает целый результат: 7/2 = 3. Над целыми числами можно выполнять деление с остатком: 7% 2 = 1.

Имеются множество действий в С ++. Например:

~ (дополнительно)

& (и)
^ (эксклюзив или)
| (или)
<< (смещение влево)
>> (смещение вправо)
Эти операции применяются к целым числам.

Результат операции зависит от количества операндов; & унарная операция- операция определяет адреса переменных; & бинарная операция - означает логическое сложение. Результат операции зависит от типа операндов: в выражениях a + b, если тип операндов являются действительным числом, то результат тоже принимает действительное значение, если операнды являются целыми числами, то тип результата тоже принимает действительное значение. C ++ имеет оператор присваивание =. Например, при выражении a = b = c значением c присваивается переменной b, затем переменной a. Операция x * = 4 равносильно с операцией x = x * 4.

Большинство программ на C ++ широко используют указатели-*. * - это унарная операция, *p - объект, который указывается с помощью p. Этот процесс также называется косвенной адресацией. Например, если char * p, p-это символ, обозначенный с помощью p. Часто операции ++ и -- полезна при работе с указателями.

В C ++ некоторые общие математические функции определены в файле <cmath>. Например, нахождение корня, возведение в степень, sin (), cos () и так далее.

Таблица 1 - математические функции в C ++.

Функция	**Tavsifi**	**Misol**
abs(a)	**a** ning moduli yoki absolyut qiymati	abs(-3.0)=3.0 abs(5.0)= 5.0
sqrt(a)	nomanfiy **a** kattalikdan olingan kvadrat ildiz	sqrt(9.0)=3.0
pow(a, b)	**a** ni **b** darajaga oshirish	pow(2,3)=8
ceil(a)	**a** ni **a** dan kichik bo`lmagan eng kichik butun songacha yaxlitlash	ceil(2.3)=3.0 ceil(-2.3)=-2.0
floor(a)	**a** ni **a** dan katta bo`lmagan eng katta butun songacha yaxlitlash	floor(12.4)=12 floor(-2.9)=-3
fmod(a, b)	a/b ifoda qoldig`ini xisoblash	fmod(4.4, 7.5) = 4.4 fmod(7.5, 4.4) = 3.1

Функция	**Tavsifi**	**Misol**
exp(a)	Вычисление экспоненты $\mathbf{e^a}$	exp(0)=1
sin(a)	Вычисление синуса от **a** в радианах	
cos(a)	Вычисление косинуса от **a** в радианах	
log(a)	Вычисление натурального логарифма от **a**	log(1.0)=0.0
log10(a)	Вычисление десятичного логарифма от **a**	Log10(10)=1
asin(a)	Вычисление арксинуса от **a**, **-1.0 < a < 1.0**	asin(1)=1.5708
acos(a)	Вычисление арккосинуса от **a**, **-1.0 < a < 1.0**	
atan(a)	Вычисление арктангенса от **a**	
sinh(a)	Вычисление гиперболического синуса от **a**	
cosh(a)	Вычисление гиперболического косинуса от **a**	
tan(a)	Вычисление тангенса от **a**	
tanh(a)	Вычисление гиперболического тангенса от **a**	

Операнды этих функций всегда должны принимать действительное значение.

Пример программирования для линейного алгоритма.

Вычислить значение выражения $Z = \frac{\cos^2(3x+a)}{tg(bx^2+a)}$, где a = -3,15; b = 4,33; x - произвольное число.

```
#include <iostream>
#include <math.h>
using namespace std;
int  main( )
{ double a,b,x,z;
cout<< " ввести значения переменных  a,b,x:\n";
cin>>a>>b>>x;
z=pow(cos(3*x+a),2)/tan(b*x*x+a);
cout <<"z="<<z;
}
```