Лабораторная работа №1.
"Знакомство с Си++. Выполнение программы простой структуры"

Цель: Знакомство со средой программирования, создание, отладка и выполнение простой программы, содержащей ввод/вывод информации и простейшие вычисления.

1. Краткие теоретические сведения

Язык Си создан в 1972 г. Деннисом Ритчи при разработке ОС Unix. Он проектировался как инструмент системного программирования с ориентацией на разработку хорошо структурированных программ. Таким образом он сочетает в себе, с одной стороны, средства языка программирования высокого уровня: описание типов данных, операторы for, while, if и т. д. , а , с другой стороны, содержит средства языка типа Ассемблер : регистровые переменные, адресную арифметику, возможность работы с полями бит и т. д.

1.1. Структура программы

Программа на языке Си имеет следующую структуру:

```
#директивы препроцессора
. . . . . .
#директивы препроцессора
функция а ( )
     операторы
функция в ( )
     операторы
void main ( )        //функция, с которой начинается выполнение программы
     операторы
          описания
          присваивания
          функция
          пустой оператор
               составной
               выбора
               циклов
               перехода
```

Директивы препроцессора - управляют преобразованием текста программы до ее компиляции. Исходная программа, подготовленная на языке Си в виде текстового файла проходит 3 этапа обработки:

1) препроцессорное преобразование текста;
2) компиляция;
3) компоновка (редактирование связей или сборка).

После этих 3 этапов формируется исполняемый машинный код программы.

Задача препроцессора - преобразование текста программы до ее компиляции. Правила препроцессорной обработки определяет программист с помощью директив препроцессора. Директива начинается с #. Например,

1) #define - указывает правила замены в тексте.

#define ZERO 0.0

Означает , что каждое использование в программе имени ZERO будет заменяться на 0.0.

2) #include< имя заголовочного файла> - предназначена для включения в текст программы текста из каталога «Заголовочных файлов», поставляемых вместе со стандартными библиотеками. Каждая библиотечная функция Си имеет соответствующее описание в одном из заголовочных файлов. Список заголовочных файлов определен стандартом языка. Употребление директивы include не подключает соответствующую стандартную библиотеку, а только позволяют вставить в текст программы описания из указанного заголовочного файла. Подключение кодов библиотеки осуществляется на этапе компоновки, т. е. после компиляции. Хотя в заголовочных файлах содержатся все описания стандартных функций, в код программы включаются только те функции, которые используются в программе.

После выполнения препроцессорной обработки в тексте программы не остается ни одной препроцессорной директивы. Программа представляет собой набор описаний и определений, и состоит из набора функций. Среди этих функций всегда должна быть функция с именем main. Без нее программа не может быть выполнена. Перед именем функции помещаются сведения о типе возвращаемого функцией значения (тип результата). Если функция ничего не возвращает, то указывается тип void: void main ( ). Каждая функция, в том числе и main должна иметь набор параметров, он может быть пустым, тогда в скобках указывается (void).

За заголовком функции размещается тело функции. Тело функции - это последовательность определений, описаний и исполняемых операторов, заключенных в фигурные скобки. Каждое определение, описание или оператор заканчивается точкой с запятой.

Определения - вводят объекты (объект - это именованная область памяти, частный случай объекта - переменная), необходимые для представления в программе обрабатываемых данных. Примером являются

```
int y = 10 ; //именованная константа
float x ;   //переменная
```

Описания - уведомляют компилятор о свойствах и именах объектов и функций, описанных в других частях программы.

Операторы - определяют действия программы на каждом шаге ее исполнения.

1.2. Константы и переменные

Константа - это значение, которое не может быть изменено. Синтаксис языка определяет 5 типов констант:

- символы;
- константы перечисляемого типа;
- вещественные числа;
- целые числа;
- нулевой указатель (NULL).

Переменные можно изменять. При задании значения переменной в соответствующую ей область памяти помещается код этого значения. Доступ к значению возможен через имя переменной, а доступ к участку памяти - по его адресу. Каждая переменная перед использованием в программе должна быть определена, т. е. ей должна быть выделена память. Размер участка памяти, выделяемой для переменной и интерпретация содержимого зависят от типа, указанного в определении переменной. Простейшая форма определения переменных:

тип список_имен_переменных;

Основные типы данных

тип данных	название	размер, бит	диапазон значений
unsigned char	беззнаковый целый длиной не менее 8 бит	8	0 . . 255
char	целый длиной не менее 8 бит	8	-128 . . 127
enum	перечисляемый	16	-32768 . . 32767
unsigned int	беззнаковый целый	16	0 . . 65535
short int (short)	короткий целый	16	-32768 . . 32767
unsigned short	беззнаковый короткий целый	16	0 . . 65535
int	целый	16	-32768 . . 32767
unsigned long	беззнаковый длинный целый	32	0 . . 4294967295
long	длинный целый	32	-214748348 . . 2147483647
float	вещественный одинарной точности	32	3.4E-38 . . 3.4E+38
double	вещественный двойной точности	64	1.7E-308 . . 1.7E+308
long double	вещественный максимальной точности	80	3.4E-4932 . . 1.1E+4932

В соответствии с синтаксисом языка переменные автоматической памяти после определения по умолчанию имеют неопределенные значения. Переменным можно присваивать начальные значения, явно указывая их в определениях:

тип имя_переменной = начальное_значение;

Этот прием называется инициализацией.

Примеры:

```
float pi = 3.14 , cc=1.3456;
unsigned int year = 1999;
```

1.3. Операции

Унарные:

&	получение адреса операнда
*	обращение по адресу (разыменование)
-	унарный минус, меняет знак арифметического операнда
~	поразрядное инвертирование внутреннего двоичного кода (побитовое отрицание)
!	логическое отрицание (НЕ). В качестве логических значений используется 0 - ложь и не 0 - истина, отрицанием 0 будет 1, отрицанием любого ненулевого числа будет 0.
++	увеличение на единицу: префиксная операция - увеличивает операнд до его использования, постфиксная операция увеличивает операнд после его использования.
- -	уменьшение на единицу: префиксная операция - уменьшает операнд до его использования, постфиксная операция уменьшает операнд после его использования.
sizeof	вычисление размера (в байтах) для объекта того типа, который имеет операнд

Бинарные операции.

Аддитивные:

+	бинарный плюс (сложение арифметических операндов)
-	бинарный минус (вычитание арифметических операндов)

Мультипликативные:

*	умножение операндов арифметического типа
/	деление операндов арифметического типа (если операнды целочисленные, то выполняется целочисленное деление)
%	получение остатка от деления целочисленных операндов

Операции сдвига (определены только для целочисленных операндов).

Формат выражения с операцией сдвига:

операнд_левый операция_сдвига операнд_правый

<<	сдвиг влево битового представления значения левого целочисленного операнда на количество разрядов, равное значению правого операнда
>>	сдвиг вправо битового представления значения правого целочисленного операнда на количество разрядов, равное значению правого операнда

Поразрядные операции:

&	поразрядная конъюнкция (И) битовых представлений значений целочисленных операндов
|	поразрядная дизъюнкция (ИЛИ) битовых представлений значений целочисленных операндов
^	поразрядное исключающее ИЛИ битовых представлений значений целочисленных операндов

Операции сравнения:

<	меньше, чем
>	больше, чем
<=	меньше или равно
>=	больше или равно

==	равно
!=	не равно

Логические бинарные операции:

&&	конъюнкция (И) целочисленных операндов или отношений, целочисленный результат ложь(0) или истина(1)
||	дизъюнкция (ИЛИ) целочисленных операндов или отношений, целочисленный результат ложь(0) или истина(1)

Условная операция.

В отличие от унарных и бинарных операций в ней используется три операнда.

Выражение1 ? Выражение2 : Выражение3;

Первым вычисляется значение выражения1. Если оно истинно, то вычисляется значение выражения2, которое становится результатом. Если при вычислении выражения1 получится 0, то в качестве результата берется значение выражения3.

Например:

x<0 ? -x : x ; //вычисляется абсолютное значение x.

Операция явного (преобразования) приведения типа.

Существует две формы: каноническая и функциональная:

1) (имя_типа) операнд
2) имя_типа (операнд)

Приоритеты операций.

Ранг	Операции
1	( ) [ ] -> .
2	! ~ - ++ -- & * (тип) sizeof тип( )
3	* / % (мультипликативные бинарные)
	+ - (аддитивные бинарные)
5	<< >> (поразрядного сдвига)
6	< > <= >= (отношения)
7	== != (отношения)
8	& (поразрядная конъюнкция «И»)
9	^ (поразрядное исключающее «ИЛИ»)
10	| (поразрядная дизъюнкция «ИЛИ»)
11	&& (конъюнкция «И»)
12	|| (дизъюнкция «ИЛИ»)
13	?: (условная операция)
14	= *= /= %= -= &= ^= |= <<= >>= (операция присваивания)
15	, (операция запятая)

1.4. Выражения

Из констант, переменных, разделителей и знаков операций можно конструировать выражения. Каждое выражение состоит из одного или нескольких операндов, символов операций и ограничителей, в качестве которых чаще всего выступают квадратные скобки. Если выражение формирует целое или вещественное число, то это арифметическое выражение. В арифметических выражениях допустимы операции: + - * / %.

Отношение - это пара арифметических выражений, объединенных знаком операции отношения. Логический тип в Си отсутствует,

поэтому принято, что отношение имеет ненулевое значение, если оно истинно и 0, если оно ложно.

1.5. Ввод и вывод

1.5.1. Ввод и вывод в стандартном Си

Обмен данными с внешним миром программа на стандартном Си реализует с помощью библиотеки функций ввода-вывода

```
#include <stdoi.h>
```

1) printf (<форматная строка>,<список аргументов>);

<форматная строка> - строка символов, заключенных в кавычки, которая показывает, как должны быть напечатаны аргументы. Например:

```
printf ( "Значение числа Пи равно  %f\n", pi);
```

Форматная строка может содержать

1) символы печатаемые текстуально;
2) спецификации преобразования
3) управляющие символы.

Каждому аргументу соответствует своя спецификация преобразования:

%d - десятичное целое число;
%f - число с плавающей точкой;
%c - символ;
%s - строка.
\n - управляющий символ новая строка.

2) scanf (<форматная строка>,<список аргументов>);

В качестве аргументов используются указатели. Например:

```
scanf(" %d%f ", &x,&y);
```

1.5.2. Ввод и вывод в Си++

Используется библиотечный файл iostream.h, в котором определены стандартные потоки ввода данных от клавиатуры cin и вывода данных на экран дисплея cout, а также соответствующие операции

1) << - операция записи данных в поток;
2) >> - операция чтения данных из потока.

Например:

```
#include <iostream.h>;
. . . . . .
cout << "\nВведите количество элементов: ";
cin >> n;
```

2. Постановка задачи

1. Вычислить значение выражения при различных вещественных типах данных (float и double). Вычисления следует выполнять с использованием промежуточных переменных. Сравнить и объяснить полученные результаты.
2. Вычислить значения выражений. Объяснить полученные результаты.

3. Варианты

№	Задание 1	Задание 2
1	$\frac{(a+b)^2-(a^2+2ab)}{b^2}$,	1) n+++m 2) m-- >n

	при a=1000, b=0.0001	3) n-- >m
2	$\frac{(a-b)^2-(a^2-2ab)}{b^2}$, при a=1000, b=0.0001	1) ++n*++m 2) m++<n 3) n++>m
3	$\frac{(a+b)^3-(a^3+3a^2b)}{3ab^2+b^3}$, при a=1000, b=0.0001	1) n---m 2) m--<n 3) n++>m
4	$\frac{(a+b)^3-(a^3)}{3ab^2+b^3+3a^2b}$, при a=1000, b=0.0001	1) n++*m 2) n++<m 3) m-- >m
5	$\frac{(a-b)^3-(a^3-3a^2b)}{b^3-3ab^2}$, при a=1000, b=0.0001	1) - -m-++n 2) m*n<n++ 3) n-- > m++
6	$\frac{(a-b)^3-(a^3-3ab^2)}{b^3-3a^2b}$, при a=1000, b=0.0001	1) m-++n 2) ++m>--n 3) --n<++m
7	$\frac{(a-b)^3-(a^3)}{b^3-3ab^2-3a^2b}$, при a=1000, b=0.0001	1) m+--n 2) m++<++n 3) n--< --m
8	$\frac{(a+b)^4-(a^4+4a^3b+6a^2b^2)}{4ab^3+b^4}$, при a=100, b=0.001	1) n++-m 2) m-- >n 3) n-- >m
9	$\frac{(a+b)^4-(a^4+4a^3b)}{6a^2b^2+4ab^3+b^4}$, при a=100, b=0.001	1) ++n*++m 2) m++<n 3) n++>m
10	$\frac{(a-b)^4-(a^4-4a^3b+6a^2b^2)}{b^4-4ab^3}$, при a=100, b=0.001	1) n---m 2) m--<n 3) n++>m
11	$\frac{(a-b)^4-(a^4-4a^3b)}{6a^2b^2-4ab^3+b^4}$, при a=100, b=0.001	1) n++*m 2) n++<m 3) m-- >m
12	$\frac{(a+b)^2-(a^2+2ab)}{b^2}$, при a=1000, b=0.0001	1) - -m-++n 2) m*n<n++ 3) n-- > m++
13	$\frac{(a-b)^2-(a^2-2ab)}{b^2}$, при a=1000, b=0.0001	1) m-++n 2) ++m>--n 3) --n<++m

14	$\frac{(a+b)^3-(a^3+3a^2b)}{3ab^2+b^3}$, при a=1000, b=0.0001	1) m+--n 2) m++<++n 3) n--< --m
15	$\frac{(a+b)^3-(a^3)}{3ab^2+b^3+3a^2b}$, при a=1000, b=0.0001	1) n++-m 2) m-- >n 3) n-- >m
16	$\frac{(a-b)^3-(a^3-3a^2b)}{b^3-3ab^2}$, при a=1000, b=0.0001	1) ++n*++m 2) m++<n 3) n++>m
17	$\frac{(a-b)^3-(a^3-3ab^2)}{b^3-3a^2b}$, при a=1000, b=0.0001	1) n---m 2) m--<n 3) n++>m
18	$\frac{(a-b)^3-(a^3)}{b^3-3ab^2-3a^2b}$, при a=1000, b=0.0001	1) n++*m 2) n++<m 3) m-- >m
19	$\frac{(a+b)^4-(a^4+4a^3b+6a^2b^2)}{4ab^3+b^4}$, при a=100, b=0.001	1) - -m-++n 2) m*n<n++ 3) n-- > m++
20	$\frac{(a+b)^4-(a^4+4a^3b)}{6a^2b^2+4ab^3+b^4}$, при a=100, b=0.001	1) m-++n 2) ++m>--n 3) --n<++m
21	$\frac{(a-b)^4-(a^4-4a^3b+6a^2b^2)}{b^4-4ab^3}$, при a=100, b=0.001	1) n++-m 2) m-- >n 3)n-- >m
22	$\frac{(a-b)^4-(a^4-4a^3b)}{6a^2b^2-4ab^3+b^4}$, при a=100, b=0.001	1) ++n*++m 2) m++<n 3) n++>m
23	$\frac{(a+b)^3-(a^3+3a^2b)}{3ab^2+b^3}$, при a=1000, b=0.0001	1) n---m 2) m--<n 3) n++>m
24	$\frac{(a+b)^3-(a^3)}{3ab^2+b^3+3a^2b}$, при a=1000, b=0.0001	1) n++*m 2) n++<m 3) m-- >m
25	$\frac{(a-b)^3-(a^3-3a^2b)}{b^3-3ab^2}$, при a=1000, b=0.0001	1) - -m-++n 2) m*n<n++ 3) n-- > m++

4. Методические указания

1. Для ввода и вывода данных использовать операции >> и << и стандартные потоки cin и cout.
2. Для вычисления степени можно использовать функцию pow(x,y) из библиотечного файла math.h.
3. При выполнении задания 1 надо использовать вспомогательные переменные для хранения промежуточных результатов.
   Например: c=pow(a,3);d=3*a*a*b;e=3*a*b*b;f=pow(b,3);

5. Содержание отчета

1. Постановка задачи.
2. Программа решения задания1.
3. Результаты работы программы для данных типа float.
4. Результаты работы программы для данных типа double.
5. Объяснение результатов.
6. Программа решения задания2.
7. Результаты работы программы.
8. Объяснение результатов.