**МИНЕСТРЕСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Балтийский государственный технический университет «ВОЕНМЕХ»

им. Д.Ф. Устинова»

Кафедра О7

«Информационные системы и программная инженерия»

Практическое задание № 1

по дисциплине «Информационные технологии и программирование»

на тему «Структура программы, основные типы данных, ввод/вывод»

Выполнила:

Студент(ка) Карпинская Полина Анатольевна

Группа Е721б

Преподаватель: Удовиченко Андрей Сергеевич,

канд. физ.-мат. наук, доцент

Санкт-Петербург

2023

**Цель работы** – изучить структуру программы, научиться использовать переменные различных типов, освоить функции форматного ввода и вывода, арифметические операции и операции присваивания.

Скрипт practical\_task\_1\_1.c:

#include <stdio.h>

int main () {

int a, b = 5, c;

float x, y = -.5, z;

printf("a = ");

scanf("%d", &a);

c = a;

x = c;

printf("a = %d, c = %d, x = %f\n", a, c, x);

a += b;

printf("a = %d\n", a);

x += b + a;

printf("x = %f\n", x);

b += a--;

printf("b = %d\n", b);

x -= ++c;

printf("x = %f\n", x);

c = a/b;

printf("c = %d\n", c);

c = a%b;

printf("c = %d\n",c);

x = 5.3;

y += ((--x - 1) / x++);

printf("x = %.2f\ty = %.2f\n\nx = %.0f\ty = %.0f\n", x - 1, y, x, y - 1);

z = a / 2;

printf("z = %f\n", z);

z = (float) a/2;

printf("z = %f\n", z);

y = x/2;

printf("y = %f\n", y);

y = (int)x/2;

printf("y = %f\n", y);

z = a%2 - (x+b)/c + (x-y)/(a-1) + 1/4\*a - y++ + ++b/3.;

/\*

float temp1 = a % 2;

float temp2 = (x + b) / c;

float temp3 = (x - y) / (a - 1);

float temp4 = 1/4 \* a;

float temp5 = y++;

float temp6 = ++b;

z = temp1 - temp2 + temp3 + temp4 - temp5 + temp6 / 3.;

\*/

printf("a = %d b = %d c = %d x = %f y = %f z = %f\n", a, b, c, x, y, z);

return 0;

}

Скрипт practical\_task\_1\_2.c:

#include <stdio.h>

int main() {

float a = 5, b;

int c = 5, d;

printf("b = ");

scanf("%f", &b);

printf("d = ");

scanf("%d", &d);

a = a + b - 2;

printf("a = %.2f, b = %.2f, c = %d, d = %d\n", a, b, c, d);

c += 1;

printf("a = %.2f, b = %.2f, c = %d, d = %d\n", a, b, c, d);

d += c - a;

printf("a = %.2f, b = %.2f, c = %d, d = %d\n", a, b, c, d);

a \*= c;

printf("a = %.2f, b = %.2f, c = %d, d = %d\n", a, b, c, d);

c -= 1;

printf("a = %.2f, b = %.2f, c = %d, d = %d\n", a, b, c, d);

a /= 10;

printf("a = %.2f, b = %.2f, c = %d, d = %d\n", a, b, c, d);

c /= 2;

printf("a = %.2f, b = %.2f, c = %d, d = %d\n", a, b, c, d);

b -= 1;

printf("a = %.2f, b = %.2f, c = %d, d = %d\n", a, b, c, d);

d \*= (c+b+a);

printf("a = %.2f, b = %.2f, c = %d, d = %d\n", a, b, c, d);

return 0;

}

**Ответы на контрольные вопросы:**

1. Структура программы на языке C обычно состоит из объявления переменных, определения функции main(), в которой находится основная логика программы, и других объявленных функций, если они есть.

2. Директива #include используется для включения содержимого указанного файла в программу. Она позволяет использовать функции и определения, которые находятся в других файлах или библиотеках.

3. main() - это главная функция программы на языке C. Она является точкой входа в программу, с которой начинается ее выполнение. В функции main() содержится основная логика программы.

4. Скалярные типы данных языка C включают int (целое число), float (вещественное число одинарной точности), double (вещественное число двойной точности), char (символ) и bool (логический тип).

5. Тип данных определяет набор значений, которые может принимать переменная этого типа, а также операции, которые можно выполнять над переменными этого типа.

6. void - это специальный тип данных в языке C, который обозначает отсутствие типа или отсутствие информации. Он используется, например, для функций, которые не возвращают значения.

7. Явное приведение типов (type casting) - это процесс преобразования значения одного типа данных в другой тип данных с использованием оператора приведения типа ((тип)выражение). Неявное приведение типов (type conversion) - это автоматическое преобразование типов, которое выполняется компилятором при необходимости.

8. Константа - это значение, которое не может быть изменено в ходе выполнения программы. Некоторые константы в языке C могут быть обозначены ключевыми словами, такими как const или #define, а некоторые могут быть указаны непосредственно в коде программы, например, числовые или символьные значения. В программах данной работы нет констант, все переменные в ходе работы принимают новые значения.

9. Переменная - это именованная область памяти, которая используется для хранения значения, которое может изменяться в ходе выполнения программы.

10. Переменную можно проинициализировать путем присваивания ей начального значения при ее объявлении, например: int x = 5;.

11. Оператор - это символ или комбинация символов, которые выполняют операцию над операндами. Операция - это сама выполненная операция или действие.

12. Унарные операции - это операции, которые выполняются над одним операндом, например инкремент или декремент. Бинарные операции - это операции, которые выполняются над двумя операндами, например сложение, умножение или сравнение.

13. Арифметические операции включают сложение (+), вычитание (-), умножение (\*), деление (/) и остаток от деления (%). Приоритет арифметических операций следующий: умножение и деление выполняются раньше сложения и вычитания.

14. При одинаковом приоритете операций, порядок выполнения зависит от ассоциативности оператора. Например, для операторов сложения и вычитания ассоциативность левоассоциативная, то есть они выполняются слева направо.

15. В случае целочисленных операндов операция деления производит целочисленное деление, результатом которого является только целая часть от деления. Если хотя бы один из операндов вещественный, то операция деления выполняется вещественным способом, результатом является число с плавающей точкой.

16. Выражение в программировании - это комбинация операторов, операндов и других выражений, которая вычисляется в некоторое значение. Оно может быть математическим выражением, логическим выражением или строковым выражением и т.д.

17. Операция присваивания вычисляет значение выражения справа от оператора присваивания и присваивает его переменной слева от оператора. Например, x = 5; присваивает переменной x значение 5.

18. В C не существует гарантированного порядка выполнения присваиваний, если их несколько в выражении. Порядок выполнения зависит от реализации или компилятора.

19. Дополнительные операции присваивания позволяют выполнять арифметические операции и присваивать результат переменной в одной команде. Например, x += 5; эквивалентно x = x + 5;.

20. Префиксная форма операции инкремента или декремента (++ или --) увеличивает или уменьшает значение переменной на единицу до ее использования в выражении. Постфиксная форма (x++ или x--) сначала использует текущее значение переменной в выражении, а затем увеличивает или уменьшает его.

21. Функция scanf() используется для ввода информации. Она позволяет взаимодействовать с пользователем или считывать значения переменных. scanf() может принимать форматированный ввод.

22. Функция printf() используется для форматированного вывода информации на экран. Она позволяют выводить значения переменных и текстовые сообщения, а также форматировать вывод, например, указывать количество десятичных знаков или выравнивать текст.

23. Функции scanf() и printf() называются функциями форматного ввода и вывода, так как они принимают специальную строку формата, в которой указывается, как должны быть считаны или отформатированы значения. Они работают, сканируя ввод или вывод в соответствии с указанными форматами и преобразовывая значения в соответствующие типы данных.

24. Управляющая строка функции scanf() содержит специальные символы формата, которые указывают, какие типы данных будут считаны из ввода. Управляющая строка функции printf() содержит специальные символы формата, которые указывают, какие значения будут отформатированы при выводе.

25. Спецификатор типа - это символ, который используется для указания типа данных переменной при ее объявлении или использовании в функции. Он определяет, как должна интерпретироваться и обрабатываться переменная компилятором или функцией. Спецификаторы типа позволяют определить размер и формат данных, а также осуществлять корректную обработку значений.

26. После управляющей строки в функции scanf() указываются параметры, которые представляют собой адреса переменных, в которые будут считываться значения. Количество параметров должно быть равно количеству управляющих символов в управляющей строке.

27. Если тип считываемой переменной не соответствует спецификатору типа в функции scanf(), то могут произойти ошибки при считывании значения. Например, если переменная имеет тип int, но используется спецификатор %f для считывания значения, то значение будет некорректно интерпретировано как вещественное число.

28. После управляющей строки в функции printf() указываются параметры, которые представляют собой значения, которые будут форматированно выводиться. Количество параметров должно быть равно количеству управляющих символов в управляющей строке.

29. Если тип выводимого значения не соответствует спецификатору типа в функции printf(), то выводимое значение может быть некорректно интерпретировано. Например, если значение имеет тип int, но используется спецификатор %f для вывода, то значение будет некорректно отформатировано как вещественное число.

30. Управляющие символы - это символы, которые используются в управляющей строке функций scanf() и printf() для задания формата вывода или чтения данных. Некоторые примеры управляющих символов:

- %d - для вывода или чтения целых чисел

- %f - для вывода или чтения вещественных чисел

- %c - для вывода или чтения символов

- %s - для вывода или чтения строк

- %p - для вывода указателей

Управляющие символы позволяют управлять форматированием и типизацией данных при вводе или выводе.