Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Методы трансляции

ОТЧЁТ
к лабораторной работе №1
на тему

ОПРЕДЕЛЕНИЕ МОДЕЛИ ЯЗЫКА. ВЫБОР ИНСТРУМЕНТАЛЬНОЙ ЯЗЫКОВОЙ СРЕДЫ

Выполнил: студент гр.253504
Фроленко К.Ю.
Проверил: ассистент кафедры информатики
Гриценко Н.Ю.

Минск 2025

СОДЕРЖАНИЕ

1 ФОРМУЛИРОВКА ЗАДАЧИ

Цель данной лабораторной работы – выделить ключевые элементы языка программирования *PL/1*, сформировав его подмножество, включающее:

– числовые и строковые константы;

– все типы переменных;

– управляющие конструкции;

– структуры данных;

– функции.

В процессе выполнения работы необходимо изучить и объяснить принципы работы указанных элементов в *PL/1*, а также их значение в процессе разработки программ.

Кроме того, необходимо выбрать и описать инструментальную среду, включающую следующие компоненты:

1 Язык программирования для разработки программного кода – *R*. Программы, написанные на *R*, будут использоваться для демонстрации и тестирования выделенного подмножества конструкций, а также для реализации инструментов автоматизированного анализа.

2 Язык, код которого будет подвергаться разбору – *PL/1*. Основное внимание уделяется анализу программ, написанных на *PL/1*, с целью выявления и демонстрации ключевых конструктивных элементов языка.

3 Операционная система – *Windows 11*. Указывается выбранная операционная система, на которой будет осуществляться разработка, тестирование и анализ программного кода.

4 Аппаратная платформа – *PC*. Используемый компьютер должен обеспечивать необходимую производительность для выполнения поставленных задач.

Практическая часть работы включает создание трёх программ на *R*, демонстрирующих применение всех перечисленных элементов для анализа и обработки *PL/1*-кода. Эти программы должны показать, как можно автоматически выявлять и обрабатывать числовые и строковые константы, различные типы переменных, управляющие конструкции, структуры данных и функции, присутствующие в *PL/1*-программах.

Особая часть работы посвящена разбору кода на *PL/1* с использованием инструментальных средств, реализованных на *R*. В ходе выполнения задания будет рассмотрен подход к анализу и обработке исходных текстов на *PL/1*, что позволит глубже изучить особенности синтаксиса и семантики языка, а также продемонстрировать возможности автоматизированного анализа программного кода.

Также в работе будет дано подробное описание инструментальной среды, что поможет систематизировать процесс разработки, тестирования и анализа программ на каждом этапе лабораторной работы.

2 ПОДМНОЖЕСТВА ЯЗЫКА ПРОГРАММИРОВАНИЯ

PL/1 – процедурный и императивный язык программирования, появившийся в 1964 году и предназначенный для решения широкого спектра задач – от инженерных расчётов до обработки данных и системного программированиях [1]. Его гибкий синтаксис, богатый набор встроенных функций и возможности описания сложных структур данных позволяют создавать программы различной направленности. В данном разделе рассматривается подмножество элементов *PL/1*, включающее числовые и текстовые константы, все типы переменных, операторы циклов (например, *do…while, do…repeat*), структуры данных, функции и условные операторы (*if…then…else*).

2.1 Общая структура и философия языка

PL/1 задумывался как универсальный язык, способный объединить возможности языков для научных вычислений, обработки данных и системного программирования. Программа на *PL/1* состоит из внешних и внутренних блоков, позволяющих организовать код в виде вложенных процедур с определёнными областями видимости переменных. Модульность достигается через процедуры, реализуемые с помощью конструкции *PROC*, а также посредством препроцессорных операторов для включения внешних файлов и текстовой замены [2].

2.2 Числовые и текстовые константы

Язык позволяет задавать неизменяемые значения непосредственно в исходном коде. Числовые константы могут представлять целые числа, числа с плавающей точкой и литералы в различных системах счисления. Текстовые константы определяются как строковые литералы, заключённые в апострофы или двойные кавычки. Эти элементы используются для задания фиксированных данных, необходимых для вычислений, сравнений или форматированного вывода [3].

2.3 Все типы переменных

Объявление переменных в *PL/1* производится с помощью оператора *DCL*. Поддерживаются базовые типы (например, *FIXED BINARY, FLOAT BINARY FIXED DECIMAL, FLOAT DECIMAL*), а также составные типы, такие как массивы для хранения упорядоченных наборов элементов и записи для объединения полей различных типов. Дополнительно, язык обеспечивает работу с указателями и битовыми типами, что позволяет решать задачи низкоуровневого и системного программирования [4].

2.4 Операторы циклов

Для организации повторяющихся операций используется оператор *DO*. Возможны циклы с фиксированным числом итераций (аналог конструкции *for*), где задаются начальное и конечное значения и шаг изменения переменной, а также циклы с условием (аналог *do…while* или *do…repeat*), где блок инструкций повторяется до тех пор, пока условие истинно или до его наступления.

2.5 Структуры данных

PL/1 обладает мощными средствами для моделирования сложных структур данных. К ним относятся массивы для хранения упорядоченных наборов элементов, записи для объединения полей различных типов, а также смешанные агрегаты, где массивы и записи могут комбинироваться. Атрибут *LIKE* позволяет создавать новые структуры на основе уже существующих, что упрощает повторное использование типовых решений.

2.6 Функции и процедуры

Модульность кода достигается посредством функций и процедур, определяемых через конструкцию *PROC*. Такие процедуры поддерживают передачу параметров (по значению или по ссылке) и могут иметь множественные точки входа (с использованием оператора *ENTRY*). Это позволяет создавать повторно используемые логические блоки, упрощать тестирование и отладку программ.

2.7 Условные операторы

Условное ветвление в *PL/1* реализуется с помощью конструкции *IF…THEN…ELSE*, которая позволяет выполнять один блок инструкций, если условие истинно, и другой – в противном случае. Эта конструкция облегчает управление потоком выполнения программы, делая алгоритмы более адаптивными к изменяющимся условиям и входным данным. Благодаря такому подходу программист получает возможность гибко реагировать на различные ситуации, улучшая читаемость и логическую структурированность кода.

2.8 Обобщённая таблица ключевых элементов подмножества PL/1

В таблице 2.1 представлено подмножество языка *PL/1*, включающее все основные элементы, необходимые для разработки программ – числовые и текстовые константы, типы переменных, операторы циклов, структуры данных, функции и условные операторы.

Таблица 2.1 – Подмножество языка программирования PL/1

Категория	Вид и пример
Константы	Числовые: `3674.799`, Текстовые: `'Привет!'`, Битовые: `'000100101'B`
Переменные	Арифметические: `FIXED BINARY`, `FLOAT BINARY`, `DECIMAL FLOAT`, `FIXED DECIMAL`, `COMPLEX FLOAT`; Строковые: `CHARACTER(n)`, `BIT(n)`; Управляющие: `LABEL`, `ENTRY`; Указатели: `POINTER`; Файлы: `FILE`.
Операторы цикла	Обычные: `DO WHILE`, `DO REPEAT`, `DO REPEAT WHILE`, `DO TO`, `DO TO BY`, `DO TO BY WHILE`, `DO BY`, составные циклы, `LEAVE`, `CONTINUE`, `GO TO`.
Структуры данных	Массивы: `DECLARE A (3,4) CHARACTER(2)`, структуры: `DECLARE 1 Person, 2 Name CHARACTER(20), 2 Age FIXED BINARY`, смешанные агрегаты данных: `DECLARE 1 student_list(100), 2 student_info, 3 last_name CHARACTER(20), 3 first_name CHARACTER(20),` атрибут `LIKE`: `DECLARE 1 X(1:100) BASED(P) LIKE Y.`
Функции	`p: PROC(a, b) RETURNS(FLOAT); … END p`
Условные операторы	`IF … THEN … [ELSE …].`

2.9 Итоговое описание

Подмножество языка *PL/1* включает все базовые элементы, необходимые для разработки программ: неизменяемые константы, переменные различных типов, операторы циклов, структуры данных, функции и условные операторы. Такое комплексное описание охватывает как средства задания и обработки данных, так и механизмы организации логики выполнения и управления ресурсами. Это позволяет глубоко изучить возможности *PL/1* и создает прочную основу для дальнейшего анализа программного кода в рамках лабораторной работы.

3 ИНСТРУМЕНТАЛЬЯ ЯЗЫКОВАЯ СРЕДА

Для выполнения лабораторной работы используется интегрированная среда разработки, включающая все необходимые компоненты — редактор кода, операционную систему, язык программирования и аппаратное обеспечение.

В данной работе разработка и тестирование программного кода ведётся в редакторе *Visual Studio Code*, который обеспечивает современную и настраиваемую рабочую среду с поддержкой автодополнения, подсветки синтаксиса и интеграции с системами контроля версий. Это значительно ускоряет процесс написания и тестирования кода.

Основным инструментом для компиляции и трансляции программ служит утилита *PLINK* версии 15.03.24, используемая для работы с исходным кодом на языке *PL/1*. *PLINK* обеспечивает высокую стабильность и скорость обработки программ, что особенно важно при выполнении лабораторной работы.

Для поддержки анализа и выполнения вычислительных задач применяется язык программирования *R* версии 4.4.2. *R* позволяет проводить статистические расчёты, визуализировать данные и выполнять глубокий анализ результатов, что является полезным при оценке работы программного кода.

Операционной системой для выполнения работы является *Windows 11*. Эта современная система гарантирует совместимость с необходимыми библиотеками и инструментами, обеспечивает высокую стабильность работы и безопасность.

Аппаратная платформа представлена ноутбуком с процессором *Intel Core Ultra 9* и 32 ГБ оперативной памяти. Такая конфигурация гарантирует высокую вычислительную мощность, позволяющую обрабатывать большие объёмы данных и эффективно компилировать даже сложные программы. Высокая производительность оборудования способствует стабильной работе среды разработки и ускоряет процесс тестирования.

В итоге, используемая инструментальная среда включает *Visual Studio Code* в качестве редактора, утилиту *PLINK* версии 15.03.24 для работы с *PL/1*, язык *R* версии 4.4.2 для анализа и вычислений, операционную систему *Windows 11* и ноутбук с процессором *Intel Core Ultra 9* с 32 ГБ оперативной памяти.

ЗАКЛЮЧЕНИЕ

В данной лабораторной работе была поставлена задача выделения ключевых элементов языка программирования *PL/1*, включающих числовые и строковые константы, все типы переменных, управляющие конструкции, структуры данных, функции и условные операторы. Проведённый анализ показал, что язык *PL/1* обладает широким спектром возможностей для решения как прикладных, так и системных задач благодаря гибкому синтаксису, богатому набору встроенных функций и продвинутым средствам моделирования данных.

Подмножество языка, определённое в работе, охватывает все необходимые компоненты – от задания неизменяемых констант до организации логики выполнения посредством циклов и условных операторов. Такое комплексное представление элементов *PL/1* создаёт прочную базу для дальнейшего анализа программного кода, позволяя более глубоко изучить особенности синтаксиса и семантики языка.

Определённая инструментальная языковая среда также полностью удовлетворяет требованиям лабораторной работы. Использование *Visual Studio Code* в качестве редактора обеспечивает удобную и современную рабочую среду, а утилита *PLINK* (версия 15.03.24) позволяет эффективно компилировать и транслировать исходный код *PL/1*. Дополнительно применение языка *R* (версия 4.4.2) способствует проведению статистического анализа и визуализации данных, что является важным инструментом при оценке работы программ. Операционная система *Windows 11* и высокопроизводительный ноутбук с процессором *Intel Core Ultra 9* и 32 ГБ оперативной памяти гарантируют стабильную и быструю работу среды разработки.

Таким образом, выполненная работа позволяет не только полноценно охватить все необходимые элементы языка *PL/1*, но и создать эффективную инструментальную среду для разработки, тестирования и анализа программного кода. Полученные результаты создают прочную основу для дальнейшей практической реализации и автоматизированного анализа исходных текстов на *PL/1*.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] PL/1 — Википедия [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/ПЛ/1. – Дата доступа: 02.02.2025.

[2] PL1-KT Documentation [Электронный ресурс]. – Режим доступа: https://pl1.su/compiler-pl-1-kt/documentation-load/. – Дата доступа: 02.02.2025.

[3] IBM Enterprise PL/I for z/OS Language Reference [Электронный ресурс]. – Режим доступа: https://www.ibm.com/support/pages/enterprise-pli-zos-documentation-library. – Дата доступа: 03.02.2025.

[4] Micro Focus Open PL/I Language Reference Manual [Электронный ресурс]. – Режим доступа: https://www.microfocus.com/documentation/enterprise-developer/ed60/ED-VS2017/BKPFPFPREF.html. – Дата доступа: 03.02.2025.

ПРИЛОЖЕНИЕ А
(обязательное)
Исходный код программы

```
#Код 1: Константы, переменные и функции
test: PROCEDURE OPTIONS(main);
   DECLARE S1 CHAR(*) VAR STATIC INITIAL('yes');
   DECLARE S2 FIXED DECIMAL(10,5) STATIC INITIAL(3674.799);
   DECLARE S3 BIT(9) STATIC INITIAL('000100101'B);
   DECLARE S4 FIXED BINARY(16) STATIC INITIAL(12345);
   DECLARE S5 FLOAT BINARY(24) STATIC INITIAL(3.14159);
   DECLARE S6 DECIMAL FLOAT(10) STATIC INITIAL(3674.799);
   DECLARE S7 COMPLEX FLOAT STATIC INITIAL('1+2I');
   DECLARE S8 LABEL;
   DECLARE S9 ENTRY VARIABLE;
   DECLARE S10 POINTER;
   DECLARE S11 FILE;

   OPEN FILE(S11) UPDATE RECORD TITLE('output.txt');

   DECLARE based_S2 FIXED DECIMAL(10,5) BASED(S10);

   S9 = simple_procedure;

   PUT SKIP LIST(S1);
   PUT SKIP LIST(S2);
   PUT SKIP LIST(S3);
   PUT SKIP LIST(S4);
   PUT SKIP LIST(S5);
   PUT SKIP LIST(S6);
   PUT SKIP LIST(S7);

   S10 = ADDR(S2);
   PUT SKIP LIST('До, S2 = ', S2);
   based_S2 = S2 + 100;
   PUT SKIP LIST('После, S2 = ', S2);

   IF S1 = 'yes' THEN S8 = geometric_mean;

   GOTO S8;

geometric_mean:
   PUT SKIP LIST('Перешли в geometric_mean');
   CALL S9;

   simple_procedure: PROCEDURE;
      PUT SKIP LIST('Вызвана simple_procedure');
   END simple_procedure;

END test;

#Код 2: Операторы цикла и условные операторы
test: PROCEDURE OPTIONS(main);

   DECLARE I FIXED BINARY (10) STATIC INITIAL(1);

    DO WHILE (I < 10);
      I = I + 1;

      IF MOD(I, 2) = 0 THEN CONTINUE;
```

```
      PUT LIST('Значение I (нечетное): ', I);
   END;

   I = 1;

   DO I = 1 REPEAT (I + 1);
      IF I = 2 THEN
         PUT LIST('Значение I (I = 2,  DO REPEAT): ', I);
      ELSE
         PUT LIST('Значение I (DO REPEAT): ', I);

      IF I > 5 THEN LEAVE;
   END I;

   I = 1;

   DO I = 1 REPEAT (I + 1) WHILE (I <= 5);
      PUT LIST('Значение I (DO REPEAT WHILE): ', I);
   END I;

   I = 1;

   DO I = 1 TO 5;
      PUT LIST('Значение I (DO TO): ', I);
   END I;

   DO I = 1 TO 10 BY 2;
      PUT LIST('Значение I (DO TO BY 2): ', I);
   END I;

   DO I = 1 TO 10 BY 2 WHILE (I <= 7);
      PUT LIST('Значение I (DO TO BY WHILE): ', I);
   END I;

   DO I = 1 BY 2 TO 10;
      PUT LIST('Значение I (DO BY с пределом): ', I);
   END I;

   DO I = 1, 3, 5, 15, 25, I = 100 TO 0 BY -4;
      PUT LIST('Значение I: ', I);
   END I;

END test;

#Код 3: Структуры данных
test: PROCEDURE OPTIONS(main);

   DECLARE A (3,4) CHARACTER(2);

   DECLARE 1 Person,
      2 Name CHARACTER(20),
      2 Age FIXED BINARY;

   DECLARE 1 student_list(100),
      2 student_info,
         3 last_name CHARACTER(20),
         3 first_name CHARACTER(20);

   DECLARE 1 Y,
      2 field1 FIXED BINARY,
      2 field2 CHARACTER(10);

   DECLARE 1 X(1:100) STATIC LIKE Y;
```

```
   A(1,1) = 'A1';
   A(1,2) = 'B1';
   A(2,1) = 'A2';
   A(2,2) = 'B2';

   Person.Name = 'Иван Иванов';
   Person.Age = 25;

   student_list(1).student_info.last_name = 'Петров';
   student_list(1).student_info.first_name = 'Алексей';

   X(1).field1 = 42;
   X(1).field2 = 'Пример';

   PUT LIST('Массив A(1,1): ', A(1,1));
   PUT LIST('Массив A(2,1): ', A(2,1));
   PUT LIST('Структура Person: ', Person.Name, Person.Age);
   PUT LIST('Смешанный агрегат student_list(1): ',
            student_list(1).student_info.last_name,
            student_list(1).student_info.first_name);
   PUT LIST('Переменная X(1) (LIKE Y): ', X(1).field1, X(1).field2);

END test;
```