

Some Topics in Elementary Computer Science/Grade 11

Nguyễn Quân Bá Hồng¹

Ngày 8 tháng 9 năm 2022

¹Independent Researcher, Ben Tre City, Vietnam
e-mail: nguyenquanbahong@gmail.com; website: <https://nqbh.github.io>.

Mục lục

1	1 Số Khái Niệm về Lập Trình & Ngôn Ngữ Lập Trình	2
1.1	Khái Niệm Lập Trình & Ngôn Ngữ Lập Trình	2
1.1.1	Thông dịch	2
1.1.2	Biên dịch	2
1.2	Các Ngôn Ngữ Lập Trình	3
1.3	Các Thành Phần Của Ngôn Ngữ Lập Trình	4
1.3.1	Các thành phần cơ bản	4
1.3.2	1 số khái niệm	4
1.3.2.1	Tên	4
1.3.2.2	Hằng & biến	5
1.3.2.3	Chú thích	5
1.3.3	Tóm tắt	5
1.4	Ngôn Ngữ Pascal	5
1.4.1	Vài nét về tác giả của ngôn ngữ Pascal	5
1.4.2	Pascal – Ngôn ngữ của học đường?	6
1.4.2.1	Pascal – Sự ra đời & đặc điểm	6
1.4.2.2	Pascal & lập trình có cấu trúc	6
1.4.2.3	Pascal có tiếp tục tồn tại?	7
2	Chương Trình Đơn Giản	8
2.1	Cấu Trúc Chương Trình	8
2.1.1	Cấu trúc chung	8
2.1.2	Các thành phần của chương trình	8
2.1.2.1	Phần khai báo	8
2.1.2.2	Phần thân chương trình	9
2.1.3	Ví dụ chương trình đơn giản	9
2.2	1 Số Kiểu Dữ Liệu Chuẩn	9
2.2.1	Kiểu nguyên	10
2.2.2	Kiểu thực	10
2.2.3	Kiểu ký tự	10
2.2.4	Kiểu logic	10
2.3	Khai Báo Biến	10
2.4	Phép Toán, Biểu Thức, Câu Lệnh Gán	11
2.4.1	Phép toán	11
2.4.2	Biểu thức số học	11
2.4.3	Hàm số học chuẩn	12
2.4.4	Biểu thức quan hệ	12
2.4.5	Biểu thức logic	12
2.4.6	Câu lệnh gán	12
2.5	Các Thủ Tục Chuẩn Vào/Ra Đơn Giản	12
2.5.1	Nhập dữ liệu vào từ bàn phím	13
2.5.2	Đưa dữ liệu ra màn hình	13
2.6	Soạn Thảo, Dịch, Thực Hiện & Hiệu Chính Chương Trình	13
2.6.1	Tóm tắt	14
2.7	Bài Tập & Thực Hành	15
3	Cấu Trúc Rẽ Nhánh & Lặp	16
3.1	Cấu Trúc Rẽ Nhánh	16

3.1.1	Rẽ nhánh	16
3.1.2	Câu lệnh if-then	16
3.1.3	Câu lệnh ghép	16
3.2	Cấu Trúc Lặp	16
4	Kiểu Dữ Liệu Có Cấu Trúc	17
4.1	Kiểu Mảng	17
4.2	Kiểu Xâu	17
4.3	Kiểu Bản Ghi	17
5	Tệp & Thao Tác với Tệp	18
5.1	Kiểu Dữ Liệu Tệp	18
5.2	Thao Tác với Tệp	18
5.3	Ví Dụ Làm Việc với Tệp	18
6	Chương Trình Con & Lập Trình Có Cấu Trúc	19
6.1	Chương Trình Con & Phân Loại	19
6.2	Ví Dụ về Cách Viết & Sử Dụng Chương Trình Con	19
6.3	Ai Là Lập Trình Viên Đầu Tiên?	19
6.4	Thư Viện Chương Trình Con Chuẩn	19
6.5	Âm Thanh	19
A	Miscellaneous	20
A.1	1 Số Phép Toán Thường Dùng	20
A.2	Giá Trị Phép Toán Logic	20
A.3	Môi Trường Turbo Pascal	20
A.4	1 Số Tên Dành Riêng	20
A.5	1 Số Kiểu Dữ Liệu Chuẩn	20
A.6	1 Số Thủ Tục & Hàm Chuẩn	20
A.7	Câu Lệnh Rẽ Nhánh & Lặp	20
A.8	Câu Lệnh with	20
A.9	1 Số Thông Báo Lỗi	20
A.10	Câu Lệnh Rẽ Nhánh & Lặp Trong C++	20
B	Lazarus on Ubuntu	21
B.1	Definition of lazarus?	21
	Tài liệu tham khảo	22

Preface

Tóm tắt kiến thức Tin học lớp 11 theo chương trình giáo dục của Việt Nam & một số chủ đề nâng cao.

Chương 1

1 Số Khái Niệm về Lập Trình & Ngôn Ngữ Lập Trình

Nội dung. *Khái niệm cơ sở về lập trình, khái niệm & các thành phần của ngôn ngữ lập trình, vai trò & phân loại chương trình dịch.*

1.1 Khái Niệm Lập Trình & Ngôn Ngữ Lập Trình

“Mọi bài toán có thuật toán đều có thể giải được trên máy tính điện tử. Khi giải bài toán trên máy tính điện tử, sau các bước xác định bài toán & xây dựng hoặc lựa chọn thuật toán khả thi là *bước lập trình*. *Lập trình* là sử dụng cấu trúc dữ liệu & các câu lệnh của ngôn ngữ lập trình cụ thể để mô tả dữ liệu & diễn đạt các thao tác của thuật toán. *Chương trình viết bằng ngôn ngữ máy* có thể được nạp trực tiếp vào bộ nhớ & thực hiện ngay, còn *chương trình viết bằng ngôn ngữ lập trình bậc cao* phải được chuyển đổi thành chương trình trên ngôn ngữ máy mới có thể thực hiện được. Chương trình đặc biệt có chức năng chuyển đổi chương trình được viết bằng ngôn ngữ lập trình bậc cao thành chương trình thực hiện được trên máy tính được gọi là *chương trình dịch*. Chương trình dịch nhận đầu vào là chương trình viết bằng ngôn ngữ lập trình bậc cao (*chương trình nguồn*), thực hiện chuyển đổi sang ngôn ngữ máy (*chương trình đích*).” – Đàm et al., 2014, p. 4

Chương trình nguồn \rightarrow Chương trình dịch \rightarrow Chương trình đích

Cf. *thông dịch* vs. *biên dịch*. “Chương trình dịch của 2 loại là *thông dịch* & *biên dịch*.” – Đàm et al., 2014, p. 5

1.1.1 Thông dịch

“*Thông dịch (interpreter)* được thực hiện bằng cách lặp lại các bước sau:

1. Kiểm tra tính đúng đắn của câu lệnh tiếp theo trong chương trình nguồn;
2. Chuyển đổi câu lệnh đó thành 1 hay nhiều câu lệnh tương ứng trong ngôn ngữ máy;
3. Thực hiện các câu lệnh vừa chuyển đổi được.

Như vậy, quá trình dịch & thực hiện các câu lệnh là luân phiên. Các chương trình thông dịch lần lượt dịch & thực hiện từng câu lệnh. Loại chương trình dịch này đặc biệt thích hợp cho môi trường đối thoại giữa người & hệ thống. Tuy nhiên, 1 câu lệnh nào đó phải thực hiện bao nhiêu lần thì nó phải được dịch bấy nhiêu lần. Các ngôn ngữ khai thác hệ quản trị cơ sở dữ liệu, ngôn ngữ đối thoại với hệ điều hành, ... đều sử dụng trình thông dịch.” – Đàm et al., 2014, p. 5

1.1.2 Biên dịch

“*Biên dịch (compiler)* được thực hiện qua 2 bước:

1. Duyệt, phát hiện lỗi, kiểm tra tính đúng đắn của các câu lệnh trong chương trình nguồn;
2. Dịch toàn bộ chương trình nguồn thành 1 chương trình đích có thể thực hiện trên máy & có thể lưu trữ để sử dụng lại khi cần thiết.

Như vậy, trong thông dịch, không có chương trình đích để lưu trữ, trong biên dịch cả chương trình nguồn & chương trình đích có thể lưu trữ lại để sử dụng về sau. Thông thường, trong môi trường làm việc trên 1 ngôn ngữ lập trình cụ thể, ngoài chương trình dịch còn có 1 số thành phần có chức năng liên quan như biên soạn, lưu trữ, tìm kiếm, cho biết các kết quả trung gian, ... E.g., Turbo Pascal 7.0, Free Pascal 1.2, Visual Pascal 2.1, ... là các môi trường làm việc trên ngôn ngữ Pascal; Turbo C++, Visual C++, ... là các môi trường làm việc trên ngôn ngữ C++. Các môi trường lập trình khác nhau ở những dịch vụ mà nó cung cấp, đặc biệt là các dịch vụ nâng cấp, tăng cường các khả năng mới cho ngôn ngữ lập trình.” – Đàm et al., 2014, p. 5

1.2 Các Ngôn Ngữ Lập Trình

“Đã có hàng nghìn ngôn ngữ lập trình được thiết kế & mỗi năm lại có thêm nhiều ngôn ngữ lập trình mới xuất hiện. Các ngôn ngữ thường được nhắc đến là: ADA, ALGOL, APL, ASSEMBLY, BASIC, C, C++, C#, COBOL, DELPHI, FORTRAN, JAVA, JAVASCRIPT, LISP, LOGO, PASCAL, PERL, PHP, PROLOG, PYTHON, RUBY, ... Sự phát triển của ngôn ngữ lập trình gắn liền với sự phát triển của tin học. Mỗi loại ngôn ngữ phù hợp hơn với 1 số lớp bài toán nhất định. Cùng với tên các ngôn ngữ lập trình, các thuật ngữ thường được nhắc tới là “lập trình cấu trúc”, “lập trình hướng đối tượng”, “lập trình web”, ...

Những ngôn ngữ lập trình hiện nay thường cung cấp các thư viện bao gồm nhiều hàm hỗ trợ giao diện người dùng & các thiết bị đầu cuối. *Cập nhật dữ liệu theo thời gian thực* là 1 hướng phát triển nhằm đáp ứng các nhu cầu đồng bộ hóa nhanh dữ liệu dùng chung cho nhiều nơi hay là để thỏa mãn nhu cầu cần đồng bộ hóa dữ liệu của các dịch vụ (như trong ngân hàng, hàng không & quân sự). Ngoài việc hỗ trợ cho các giao diện, ngày nay hầu hết các hệ điều hành (UNIX/Linux, Netware & Windows) đều có khả năng đa nhiệm nâng cao hiệu quả của máy tính. Do đó, các ngôn ngữ thường có thêm các hàm, thủ tục hay các biến cho phép người lập trình tận dụng điều này. Dưới đây giới thiệu 1 số ngôn ngữ lập trình thông dụng: FORTRAN, ALGOL, LISP, COBOL, BASIC, PASCAL, C, C++, JAVA, ...

- FORTRAN là 1 ngôn ngữ lập trình được phát triển từ những năm 1950 & vẫn được dùng nhiều trong tính toán khoa học cho đến hơn nửa thế kỷ sau. Tên gọi này xuất phát từ việc ghép các từ tiếng Anh **Formula Translator**, i.e., *dịch công thức*. Các phiên bản đầu có tên chính thức là FORTRAN. Điểm yếu của FORTRAN là thiếu hỗ trợ trực tiếp cho các kết cấu có cấu trúc, kiểu dữ liệu còn nghèo, không thuận lợi cho xử lý xâu. Fortran được phát triển ban đầu như là 1 ngôn ngữ thủ tục. Tuy nhiên các phiên bản mới của Fortran đã có các tính năng hỗ trợ lập trình hướng đối tượng.
- ALGOL do Ủy ban các nhà tin học châu Âu & Hoa Kỳ tạo ra năm 1958, là ngôn ngữ tiên phong đưa ra tập các thủ tục, định kiểu dữ liệu cực kỳ phong phú, ... & có ảnh hưởng mạnh tới các ngôn ngữ đời sau.
- LISP do John McCarthy của Học viện Công nghệ Massachusetts tạo ra năm 1958, là ngôn ngữ đặc biệt thích hợp cho thao tác ký hiệu & xử lý danh sách thường gặp trong các bài toán tổ hợp, thích hợp cho việc chứng minh định lý. Gần đây LISP được dùng để phát triển hệ chuyên gia & các hệ thống dựa trên tri thức.
- COBOL ra đời năm 1959, được chấp nhận dùng cho các ứng dụng xử lý dữ liệu thương mại, kinh doanh.
- BASIC là ngôn ngữ được phát triển năm 1963 bởi John Kemeny & Thomas Kurtz. BASIC là ngôn ngữ còn nhiều hạn chế như thực hiện câu lệnh chủ yếu là tuần tự từ trên xuống, điều khiển chương trình chỉ nhờ lệnh IF ... THEN & GOSUB.
- PASCAL do Niklaus Wirth phát triển dựa trên Algol năm 1970. Pascal là tên nhà toán học & triết học người Pháp Blaise Pascal. Pascal là ngôn ngữ đặc biệt thích hợp cho kiểu lập trình cấu trúc. Cho đến nay, Pascal vẫn được dùng để giảng dạy về lập trình trong nhiều trường trung học & đại học trên thế giới. Đó là ngôn ngữ cho phép mô tả thuật toán thuận tiện. Pascal cũng phục vụ nhiều ứng dụng kỹ nghệ khoa học & lập trình hệ thống. Phần lớn hệ điều hành Macintosh được viết bằng Pascal. Hệ sắp chữ \TeX được Donald Knuth viết bằng ngôn ngữ mang nhiều yếu tố của Pascal. Trình biên dịch Free Pascal được viết bằng Pascal là 1 trình biên dịch mạnh có khả năng biên dịch cả ứng dụng cũ & mới (phân phối miễn phí dưới giấy phép GNU), hỗ trợ nhiều hệ điều hành.
- C là ngôn ngữ được xây dựng bởi Dennis Ritchie năm 1972 & được dùng trong hệ điều hành UNIX. Từ đó C còn được dùng trong nhiều hệ điều hành khác & trở thành 1 trong những ngôn ngữ phổ dụng nhất. C rất hiệu quả & được ưa chuộng để viết các phần mềm hệ thống, mặc dù nó cũng được dùng cho việc viết các ứng dụng. Ngoài ra, C cũng thường được dùng làm ngôn ngữ giảng dạy lập trình. Ngày nay, C được phát triển & mang nhiều tính năng mới làm cho nó mềm dẻo thêm.
- C++ là ngôn ngữ lập trình hỗ trợ lập trình cấu trúc (thủ tục, dữ liệu trừu tượng), lập trình hướng đối tượng. Từ thập niên 1990, C++ đã trở thành 1 trong những ngôn ngữ phổ biến nhất. C++ góp phần xây dựng những ứng dụng lớn nhất hiện nay như hệ điều hành Windows, trình duyệt & máy tìm kiếm Google, ... Năm 1983, Bjarne Stroustrup ở phòng thí nghiệm Bell đã phát triển C++. Trong suốt thập niên 1980 “C với các lớp” được coi là 1 bản nâng cao của ngôn ngữ C. Tên C++ được dùng lần đầu tiên vào tháng 12/1983. Cái tên C++ cho biết C++ là ngôn ngữ được phát triển trên cơ sở ngôn ngữ C.

- JAVA được khởi đầu bởi James Gosling & các đồng nghiệp ở Sun Microsystems năm 1991 là 1 phần của *Dự án Xanh*. Ban đầu ngôn ngữ này được gọi là Oak (i.e., cây sồi, do bên ngoài cơ quan của ông Gosling có trồng nhiều loại cây này). Họ dự định phát triển ngôn ngữ này thay cho C++. Công ty Sun Microsystems đang giữ bản quyền & phát triển Java thường xuyên. Java được phát hành vào năm 1994, rồi nó trở nên nổi tiếng khi Netscape tuyên bố tại hội thảo *SunWorld năm 1995* là trình duyệt Navigator của họ sẽ hỗ trợ Java. Java có thể tương thích với nhiều họ máy như PC, Macintosh, tương thích với nhiều hệ điều hành như Windows, Linux. Người ta nói Java là ngôn ngữ lập trình 1 lần (trên 1 máy) nhưng có thể chạy nhiều lần (trên nhiều máy). Java được sử dụng chủ yếu để lập trình trên môi trường mạng & Internet.” – Đàm et al., 2014, pp. 6–8

1.3 Các Thành Phần Của Ngôn Ngữ Lập Trình

1.3.1 Các thành phần cơ bản

“Mỗi ngôn ngữ lập trình thường có 3 thành phần cơ bản là *bảng chữ cái*, *cú pháp* & *ngữ nghĩa*.” “*Bảng chữ cái* là tập các ký tự được dùng để viết chương trình. Không được phép dùng bất kỳ ký tự nào ngoài các ký tự quy định trong bảng chữ cái. Trong Pascal, bảng chữ cái bao gồm các ký tự sau:

- Các chữ cái thường & các chữ cái in hoa của bảng chữ cái tiếng Anh: a b c d e f g h i j k l m n o p q r s t u v w x y z A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
- 10 chữ số thập phân Ả Rập: 0 1 2 3 4 5 6 7 8 9
- Các ký tự đặc biệt: + - * / = < > [] . , ; # ^ \$ @ & () { } : ' - , dấu cách (mã ASCII 32).

Bảng chữ cái của các ngôn ngữ lập trình nói chung không khác nhau nhiều. E.g., bảng chữ cái của ngôn ngữ lập trình C++ chỉ khác Pascal là có sử dụng thêm các ký tự như dấu nháy kép ("), dấu sổ ngược (\), dấu chấm than (!). *Cú pháp* là bộ quy tắc để viết chương trình. Dựa vào chúng, người lập trình & chương trình dịch biết được tổ hợp nào của các ký tự trong bảng chữ cái là hợp lệ & tổ hợp nào là không hợp lệ. Nhờ đó, có thể mô tả chính xác thuật toán để máy thực hiện. *Ngữ nghĩa* xác định ý nghĩa thao tác cần phải thực hiện, ứng với tổ hợp ký tự dựa vào ngữ cảnh của nó.” – Đàm et al., 2014, p. 9

“Cú pháp cho biết cách viết 1 chương trình hợp lệ, còn ngữ nghĩa xác định ý nghĩa của các tổ hợp ký tự trong chương trình. Các lỗi cú pháp được chương trình dịch phát hiện & thông báo cho người lập trình biết. Chỉ có các chương trình không còn lỗi cú pháp mới được dịch sang ngôn ngữ máy. Các lỗi ngữ nghĩa khó phát hiện hơn. Phần lớn các lỗi ngữ nghĩa chỉ được phát hiện khi thực hiện chương trình trên dữ liệu cụ thể.” – Đàm et al., 2014, p. 10

1.3.2 1 số khái niệm

1.3.2.1 Tên

“Mọi đối tượng trong chương trình đều phải được đặt tên theo quy tắc của ngôn ngữ lập trình & từng chương trình dịch cụ thể. Trong Turbo Pascal, tên là 1 dãy liên tiếp *không quá 127 ký tự bao gồm chữ số, chữ cái hoặc dấu gạch dưới & bắt đầu bằng chữ cái hoặc dấu gạch dưới*. Trong chương trình dịch Free Pascal, tên có thể có độ dài tới 255 ký tự.” – Đàm et al., 2014, p. 10. “Ngôn ngữ Pascal không phân biệt chữ hoa, chữ thường trong tên. 1 số ngôn ngữ lập trình khác (e.g., C++) phân biệt chữ hoa, chữ thường. E.g., *AB* & *Ab* là 1 tên trong Pascal, nhưng lại là 2 tên khác nhau trong C++. Nhiều ngôn ngữ lập trình, trong đó có Pascal, phân biệt 3 loại tên: tên dành riêng, tên chuẩn, tên do người lập trình đặt.

- **Tên dành riêng.** 1 số tên được ngôn ngữ lập trình quy định dùng với *ý nghĩa riêng xác định*, người lập trình không được sử dụng với ý nghĩa khác. Những tên này được gọi là *tên dành riêng* (còn được gọi là *từ khóa*). E.g., 1 số tên dành riêng: Trong Pascal: `program`, `uses`, `const`, `type`, `var`, `begin`, `end`. Trong C++: `main`, `include`, `if`, `while`, `void`.
- **Tên chuẩn.** 1 số tên được ngôn ngữ lập trình dùng với *ý nghĩa nhất định nào đó*. Những tên này được gọi là *tên chuẩn*. Tuy nhiên, người lập trình có thể khai báo & dùng chúng với ý nghĩa & mục đích khác. Ý nghĩa của các tên chuẩn được quy định trong các *thư viện* của ngôn ngữ lập trình. E.g., 1 số tên chuẩn: Trong Pascal: `abs`, `integer`, `real`, `sqr`, `longint`, `extended`, `sqrt`, `byte`, `break`. Trong C++: `cin`, `cout`, `getchar`.
- **Tên do người lập trình đặt.** Tên do người lập trình đặt được dùng với ý nghĩa riêng, xác định bằng cách khai báo trước khi sử dụng. Các tên này không được trùng với tên dành riêng.” – Đàm et al., 2014, p. 11

1.3.2.2 Hằng & biến

1.3.2.2.1 Hằng.

Định nghĩa 1.3.1 (Hằng). Hằng là đại lượng có giá trị không thay đổi trong quá trình thực hiện chương trình.

“Trong các ngôn ngữ lập trình thường có các hằng số học, hằng logic, hằng xâu.

- *Hằng số học* là các số nguyên hay số thực (dấu phẩy tĩnh hoặc dấu phẩy động).
- *Hằng logic* là giá trị *đúng* hoặc *sai* tương ứng với *true* hoặc *false*.
- *Hằng xâu* là dãy ký tự trong bộ mã ASCII. Khi viết, chuỗi ký tự này được đặt trong cặp dấu nháy (Pascal dùng dấu nháy đơn, còn C++ dùng dấu nháy kép).

Lưu ý 1.3.1. *Hằng dấu nháy đơn trong Pascal được viết là '''. Để có xâu tiếng Anh I'm a student, trong Pascal cần viết là 'I'm a student'.*

1.3.2.2.2 Biến.

Định nghĩa 1.3.2 (Biến). Biến là đại lượng được đặt tên, dùng để lưu trữ giá trị & giá trị có thể được thay đổi trong quá trình thực hiện chương trình.

Tùy theo cách lưu trữ & xử lý, Pascal phân biệt nhiều loại biến. Các biến dùng trong chương trình đều phải khai báo. Việc khai báo biến sẽ được trình bày ở các phần sau.” – Đàm et al., 2014, p. 12

1.3.2.3 Chú thích

“Có thể đặt ra các đoạn chú thích trong chương trình nguồn. Các chú thích này giúp cho người đọc chương trình nhận biết ý nghĩa của chương trình đó dễ hơn. Chú thích không ảnh hưởng đến nội dung chương trình nguồn & được chương trình dịch bỏ qua. Trong Pascal các đoạn chú thích được đặt giữa cặp dấu { & } hoặc (* *). 1 trong những cách tạo chú thích trong C++ là đặt chúng giữa cặp dấu /* & */.” – Đàm et al., 2014, p. 13

1.3.3 Tóm tắt

- “Cần có chương trình dịch để chuyển chương trình nguồn thành chương trình đích.
- Có 2 loại chương trình dịch: thông dịch & biên dịch.
- Các thành phần của ngôn ngữ lập trình: bảng chữ cái, cú pháp & ngữ nghĩa.
- Mọi đối tượng trong chương trình đều phải được đặt tên:
 - *Tên dành riêng*: Được dùng với ý nghĩa riêng, không được dùng với ý nghĩa khác.
 - *Tên chuẩn*: Tên dùng với ý nghĩa nhất định, khi cần dùng với ý nghĩa khác thì phải khai báo.
 - *Tên do người lập trình đặt*: Cần phải khai báo trước khi sử dụng.
- *Hằng*: Đại lượng có giá trị không thay đổi trong quá trình thực hiện chương trình.
- *Biến*: Đại lượng được đặt tên. Giá trị của biến có thể thay đổi trong quá trình thực hiện chương trình.” – Đàm et al., 2014, p. 13

1.4 Ngôn Ngữ Pascal

1.4.1 Vài nét về tác giả của ngôn ngữ Pascal

“GS. NIKLAUS WIRTH, tác giả của ngôn ngữ Pascal sinh năm 1934 tại Thụy Sĩ. Ông tốt nghiệp DH Công nghệ Liên bang Thụy Sĩ¹ (ETH) tại thành phố quê hương Zurich vào năm 1959. Ông nhận bằng Thạc sĩ tại trường Đại học Tổng hợp Laval ở Quebec, Canada năm 1960. Năm 1963, tại Đại học Tổng hợp California (Mỹ) dưới sự lãnh đạo của GS. Harry Huskey ông thực hiện đề án mở rộng ngôn ngữ Algol-60 (ngôn ngữ Euler) & bảo vệ luận án tiến sĩ. Trong các năm 1963–1967, ông giảng dạy tại Đại học Tổng hợp Stanford (Mỹ). Cũng trong thời gian này ông được mời vào nhóm chuyên gia quốc tế IFIP thiết kế

¹ETH Zurich (English: ETH; Swiss Federal Institute of Technology in Zürich; German: Eidgenössische Technische Hochschule Zürich). Có thể xem [Wikipedia/ETH Zurich](https://ethz.ch/en.html). Official website: <https://ethz.ch/en.html>.

Algol-68. Năm 1967, Wirth trở về nước & giảng dạy tại Tổng hợp Zurich. Năm 1968, ông chuyển sang ETH, tại đây ông bắt đầu tham gia thiết kế Pascal. Năm 1970, chương trình dịch Pascal đầu tiên được hoàn thành. Trong thời gian 1978–1981, Wirth lãnh đạo dự án thiết kế ngôn ngữ Modula-2, máy tính cá nhân 16-bit Lilit dựa trên nó & hệ điều hành OS Medos. Tất cả các chương trình, kể cả chương trình hệ thống, được thực hiện hoàn toàn trên Modula-2. Năm 1984, do công lao to lớn trong việc phát triển các ngôn ngữ lập trình & thiết kế máy tính cá nhân Lilit, ông được giải thưởng Alan Turing – giải thưởng cao quý nhất trong giới Tin học, mà về ý nghĩa được coi là tương đương với giải Nobel. Trong thời gian 1986–1989, Wirth lãnh đạo dự án phát triển ngôn ngữ Oberon, hệ điều hành hướng đối tượng Oberon & trạm làm việc 32-bit Ceres. Rất nhiều ý tưởng của dự án này được các đồng nghiệp từ phòng thí nghiệm Sun Labs sử dụng cho ngôn ngữ & công nghệ Java. Từ năm 1990 ông lãnh đạo Viện Các hệ thống máy tính tại ETH. Năm 1999, ông nghỉ hưu & trở thành GS danh dự của ETH.” – Đàm et al., 2014, p. 14

1.4.2 Pascal – Ngôn ngữ của học đường?

“Pascal! Hỡi có người lập trình nào không biết ngôn ngữ này? Thành công vang dội của nó bắt đầu vào những năm 1980, thời kỳ của cuộc cách mạng trong công nghiệp máy tính & giai đoạn nở rộ của lập trình có cấu trúc. Có thể nói Pascal xứng đáng là điểm khởi đầu cho 1 kỷ nguyên mới của các ngôn ngữ lập trình.” – Đàm et al., 2014, p. 15

1.4.2.1 Pascal – Sự ra đời & đặc điểm

“Vào đầu năm 1971, bản mô tả ngôn ngữ mới của Viện Công nghệ Liên bang Thụy Sĩ được công bố trong số đầu tiên của tạp chí Acta Informatica. Sự ra đời của Pascal có thể được tính từ thời điểm này. Tác giả của nó, GS. Niklaus Wirth trở nên nổi tiếng vì sự xuất hiện của Pascal. Những dự án sau này của ông chứng minh hùng hồn cho thế giới rằng chìa khóa tới các bí mật của máy tính chính là ở sự kết hợp hài hòa giữa Toán học, Công nghệ & Lập trình. & nếu tiếp cận vấn đề 1 cách hợp lý thì có thể tạo ra các ngôn ngữ, hệ điều hành & ngay cả các máy tính tuyệt vời vượt các chuẩn công nghiệp, ... chỉ bằng sức lực của những sinh viên.

Người ta thường nói đến các điểm khác biệt của Pascal so với những ngôn ngữ khác, e.g., ngôn ngữ C. Nhưng chính Dennis Ritchie, tác giả của C đã phát biểu (1993): [translated] “Tôi khẳng định rằng Pascal rất gần với C. 2 ngôn ngữ này khác biệt về chi tiết, nhưng về cơ sở chúng là giống nhau ... Khi nhìn vào các kiểu dữ liệu, cũng như các phép toán trên chúng, ta có thể phát hiện ra những sự giống nhau rất lớn, mặc dù rằng ý đồ của Wirth khi tạo ra Pascal rất khác với ý đồ của chúng tôi khi tạo ra C. Wirth tạo ngôn ngữ để giảng dạy & do vậy tất nhiên cần đáp ứng các yêu cầu sư phạm.

Khác với C, Pascal không được tạo ra để làm ngôn ngữ lập trình hệ thống. Để đề cao tính đơn giản & hiệu quả dựa trên mức độ hiểu về khoa học lập trình thời bấy giờ, Wirth chủ tâm chấp nhận các hạn chế của ngôn ngữ, trước tiên trong các vấn đề liên quan đến thế giới bên ngoài (vào/ra & các công cụ phụ thuộc hệ thống). Mặc dù vậy, nếu nghĩ rằng Pascal là ngôn ngữ chỉ dành để giảng dạy thì sẽ sai lầm. Hãy nghe chính ý kiến của Wirth về vấn đề này (1984): [translated] “Có người cho rằng Pascal được thiết kế như 1 ngôn ngữ để giảng dạy. Mặc dù điều này là đúng, nhưng việc sử dụng nó để giảng dạy không phải là mục đích duy nhất. Thực tế, tôi không tin vào sự thành công của việc áp dụng trong khi học các công cụ & phương pháp mà không thể sử dụng để giải quyết các bài toán thực tế. Theo các tiêu chuẩn ngày nay thì Pascal có những nhược điểm rõ ràng khi lập trình các hệ thống lớn, nhưng 15 năm trước, nó là thỏa hiệp hợp lý giữa cái mong muốn & hiệu quả.” – Đàm et al., 2014, p. 15

1.4.2.2 Pascal & lập trình có cấu trúc

“Ngôn ngữ Pascal được Wirth tạo ra dưới ảnh hưởng các tư tưởng của C. A. R. Hoare, đăng trong công trình “Bàn về cấu trúc dữ liệu” (*Notes on Data Structuring*, Academic Press, 1972). Đóng góp của nhà bác học người Anh lớn tới mức có thể gọi ông là cha đỡ đầu của Pascal. Pascal được coi như khởi đầu của kỷ nguyên lập trình cấu trúc. Tất cả bắt đầu từ bài báo của chuyên gia người Hà Lan E. W. Dijkstra “Lập trình cấu trúc” (*Structured Programming*, 1969). Trong bài báo này ông đề xuất hạn chế các cấu trúc điều khiển chương trình chỉ ở 3 dạng là tuần tự, rẽ nhánh, & lặp. Từ đó suy ra rằng *câu lệnh chuyển vô điều kiện* (*goto*) trong các ngôn ngữ ALGOL & PL/1 (rất phổ biến thời bấy giờ) là hoàn toàn không cần thiết. Thật sự là Wirth cũng không dám loại câu lệnh này khỏi Pascal. Nhưng điều chủ yếu nằm ở chỗ khác: Lập trình cấu trúc liên quan đến nguyên tắc “từ trên xuống dưới” (làm mịn từng bước), yêu cầu tính cấu trúc của điều khiển & dữ liệu, dựa vào sự đơn giản & cơ sở toán học mà tăng độ tin cậy của phần mềm. Tất cả những điều này đều khả thi nhờ các khả năng của Pascal.

Về ý đồ khi xây dựng Pascal, Wirth viết: “Điểm mới của Pascal là đưa ra các cấu trúc & kiểu dữ liệu phong phú, cũng giống như ALGOL đưa ra các loại cấu trúc điều khiển. Trong ALGOL chỉ có 3 kiểu dữ liệu cơ sở: các số nguyên & thực, giá trị chân lý, mảng; Pascal đã đưa thêm các kiểu dữ liệu cơ sở & còn cho khả năng xác định những kiểu cơ sở mới (kiểu liệt kê, kiểu miền con), cũng như các dạng cấu trúc dữ liệu mới: bản ghi, tập hợp, tệp, mà 1 số trong chúng đã có trong COBOL. & tất nhiên, quan trọng nhất là *tính đệ quy* trong việc mô tả các cấu trúc & hệ quả của điều này là khả năng kết hợp & lồng các cấu trúc.” – Đàm et al., 2014, p. 15

1.4.2.3 Pascal có tiếp tục tồn tại?

“Để kết luận, xin trích dẫn lời của Dennis Ritchie tác giả ngôn ngữ C: [translated] “Pascal là 1 ngôn ngữ thanh lịch. Nó vẫn tiếp tục tồn tại. Nó đã khởi nguồn cho không ít ngôn ngữ đàn em & có ảnh hưởng sâu sắc đến việc thiết kế các ngôn ngữ lập trình nói chung”.” – Đàm et al., 2014, p. 15

Chương 2

Chương Trình Đơn Giản

Nội dung. Cấu trúc chương trình; các kiến thức cơ bản về kiểu dữ liệu, phép toán, biểu thức, câu lệnh gán, tổ chức vào/ra đơn giản; cách thực hiện chương trình trong môi trường Pascal.

2.1 Cấu Trúc Chương Trình

2.1.1 Cấu trúc chung

“Nói chung, chương trình được viết bằng 1 ngôn ngữ lập trình bậc cao thường gồm *phần khai báo & phần thân*. Phần thân chương trình nhất thiết phải có. Phần khai báo có thể có hoặc không tùy theo từng chương trình cụ thể. Khi diễn giải cú pháp của ngôn ngữ lập trình người ta thường sử dụng ngôn ngữ tự nhiên. Các diễn giải bằng ngôn ngữ tự nhiên được đặt giữa cặp dấu < & >. Các thành phần của chương trình có thể có hoặc không được đặt trong cặp dấu [&].” – Đàm et al., 2014, p. 18. Với quy ước này, cấu trúc của 1 chương trình này có thể được mô tả như sau:

```
[<phần khai báo>]
<phần thân>
```

2.1.2 Các thành phần của chương trình

2.1.2.1 Phần khai báo

“Có thể có các khai báo cho: tên chương trình, thư viện, hằng, biến & chương trình con.

2.1.2.1.1 Khai báo tên chương trình. Phần này có thể có hoặc không. Với Pascal, nếu có, phần khai báo tên chương trình bắt đầu bằng từ khóa `program`, tiếp đến là tên chương trình:

```
program <tên chương trình>;
```

trong đó, `tên chương trình` là tên do người lập trình đặt theo đúng quy định về tên.

2.1.2.1.2 Khai báo thư viện. Mỗi ngôn ngữ lập trình thường có sẵn 1 số thư viện cung cấp 1 số chương trình thông dụng đã được lập sẵn. Để sử dụng các chương trình đó cần khai báo thư viện chứa nó.

Ví dụ 2.1.1 (Khai báo thư viện). Trong Pascal: `uses crt`;. Trong C++:

```
#include <stdio.h>
#include <conio.h>
```

Thư viện `crt` trong Pascal hoặc `stdio.h` & `conio.h` trong C++ cung cấp các chương trình có sẵn để làm việc với màn hình & bàn phím. E.g., muốn xóa những gì đang có trên màn hình: Trong Pascal, sau khi khai báo thư viện `crt`, ta dùng lệnh: `clrscr`;. Trong C++, sau khi khai báo thư viện `conio.h`, ta dùng lệnh: `clrscr()`;

2.1.2.1.3 Khai báo hằng.

Ví dụ 2.1.2 (Khai báo hằng). Trong Pascal:

```
const MaxN = 1000;
      PI = 3.1416;
      KQ = 'Ket qua:';
```

Trong C++:

```
const int MaxN = 1000;
const float PI = 3.1416;
const char* KQ = "ketqua:";
```

Khai báo hằng thường được sử dụng cho những giá trị xuất hiện nhiều lần trong chương trình.

2.1.2.1.4 Khai báo biến. Tất cả các biến dùng trong chương trình đều phải đặt tên & phải khai báo cho chương trình dịch biết để lưu trữ & xử lý. Biến chỉ nhận 1 giá trị tại mỗi thời điểm thực hiện chương trình được gọi là *biến đơn*.” – Đàm et al., 2014, pp. 18–19

2.1.2.2 Phần thân chương trình

“Dãy lệnh trong phạm vi được xác định bởi cặp dấu hiệu mở đầu & kết thúc tạo thành thân chương trình.

Ví dụ 2.1.3 (Thân chương trình trong Pascal). `begin`
[<Dãy lệnh>]
`end.`

với `begin` là tên dành riêng *bắt đầu*, `end.` là tên dành riêng *kết thúc*.” – Đàm et al., 2014, p. 19

2.1.3 Ví dụ chương trình đơn giản

Ví dụ 2.1.4. *Chương trình sau thực hiện việc đưa ra màn hình thông báo “announcement”. Trong Pascal:*

```
program print_text;
begin
writeln('announcement');
end.
```

Phần khai báo chỉ có khai báo tên chương trình gồm tên dành riêng `program` & tên chương trình là `print_text`. Phần thân chương trình chỉ có 1 câu lệnh `writeln`, đưa thông báo ra màn hình. Trong C++:

```
#include <stdio.h>
void main()
{
printf("announcement");
}
```

Phần khai báo chỉ có 1 câu lệnh `include` khai báo thư viện `stdio.h`. Phần thân chương trình chỉ có 1 câu lệnh `printf` đưa thông báo ra màn hình.

Ví dụ 2.1.5. *Chương trình Pascal sau đưa ra các thông báo “1st announcement” & “2nd announcement” ra màn hình.*

```
begin
    writeln('1st announcement');
    writeln('2nd announcement');
end
```

Chương trình trên không có phần khai báo. Phần thân chương trình có 2 câu lệnh đưa 2 thông báo tương ứng ra màn hình.

2.2 1 Số Kiểu Dữ Liệu Chuẩn

“Các bài toán trong thực tế thường có dữ liệu vào & kết quả ra thuộc những kiểu dữ liệu quen biết như số nguyên, số thực, ký tự, ... Khi cần lập trình cho những bài toán như vậy, người lập trình sử dụng các kiểu dữ liệu đó thường gặp 1 số hạn chế nhất định, phụ thuộc vào các yếu tố như dung lượng bộ nhớ, khả năng xử lý của CPU, ... Vì vậy, mỗi ngôn ngữ lập trình thường cung cấp 1 số kiểu dữ liệu chuẩn cho biết phạm vi giá trị có thể lưu trữ, dung lượng bộ nhớ cần thiết để lưu trữ & các phép toán tác động lên dữ liệu. Dưới đây xét 1 số kiểu dữ liệu chuẩn thường dùng cho các biến đơn trong Pascal.” – Đàm et al., 2014, p. 21

2.2.1 Kiểu nguyên

Kiểu	Bộ nhớ lưu trữ 1 giá trị	Phạm vi giá trị
byte	1 byte	$0 \rightarrow 255$
integer	2 byte	$-32768 \rightarrow 32767$
word	2 byte	$0 \rightarrow 65535$
longint	4 byte	$-2147483648 \rightarrow 2147483647$

Bảng 2.1: Kiểu nguyên.

2.2.2 Kiểu thực

“Có nhiều kiểu dùng để khai báo các đại lượng nhận giá trị là số thực. Thường dùng hơn cả là các kiểu được liệt kê trong bảng sau:

Kiểu	Bộ nhớ lưu trữ 1 giá trị	Phạm vi giá trị
real	6 byte	0 hoặc có giá trị tuyệt đối $\in [2.9 \times 10^{-39}; 1.7 \times 10^{38}]$
extended	10 byte	0 hoặc có giá trị tuyệt đối $\in [10^{-4932}; 10^{4932}]$

Bảng 2.2: Kiểu thực.

2.2.3 Kiểu ký tự

“Ta hiểu ký tự là các ký tự thuộc bộ mã ASCII gồm 256 ký tự có mã ASCII thập phân từ 0–255. E.g., ký tự *A* có mã ASCII là 65, ký tự *a* có mã ASCII là 97. Ký tự đặc biệt, dùng để thể hiện sự ngăn cách giữa 2 từ viết liên tiếp trong các văn bản, là dấu cách. Dấu cách được gõ bằng phím *Space* – phím dài nhất trên bàn phím & có mã ASCII bằng 32.” – Đàm et al., 2014, p. 22

Kiểu	Bộ nhớ lưu trữ 1 giá trị	Phạm vi giá trị
char	1 byte	0 256 ký tự trong bộ mã ASCII

Bảng 2.3: Kiểu ký tự.

2.2.4 Kiểu logic

Kiểu	Bộ nhớ lưu trữ 1 giá trị	Phạm vi giá trị
boolean	1 byte	0 true hoặc false

Bảng 2.4: Kiểu ký tự.

Lưu ý 2.2.1. “Người lập trình cần tìm hiểu đặc trưng của các kiểu dữ liệu chuẩn được xác định bởi bộ dịch & sử dụng để khai báo biến.” – Đàm et al., 2014, p. 22

2.3 Khai Báo Biến

“Như đã nêu ở trên, mọi biến dùng trong chương trình đều cần khai báo tên & kiểu dữ liệu. Tên biến dùng để xác lập quan hệ giữa biến với địa chỉ bộ nhớ nơi lưu trữ giá trị biến. Mỗi biến chỉ được khai báo 1 lần. Trong phần này ta chỉ xét khai báo các biến đơn.

Trong Pascal, khai báo biến bắt đầu bằng từ khóa **var** có dạng

```
var <danh sách biến>: <kiểu dữ liệu>;
```

trong đó:

- *danh sách biến* là 1 hoặc nhiều tên biến, các tên biến được viết cách nhau bởi dấu phẩy ;
- *kiểu dữ liệu* thường là 1 trong các kiểu dữ liệu chuẩn hoặc kiểu dữ liệu do người lập trình định nghĩa.

Sau từ khóa **var** có thể khai báo nhiều danh sách khác nhau, i.e., cấu trúc:

<danh sách biến>: <kiểu dữ liệu>;

có thể xuất hiện nhiều lần.” – Đàm et al., 2014, pp. 22–23

Lưu ý 2.3.1 (1 số chú ý khi khai báo biến). • “Cần đặt tên biến sao cho gọi nhớ đến ý nghĩa của biến đó. Điều này rất có lợi cho việc đọc, hiểu & sửa đổi chương trình khi cần thiết.”

- “Không nên đặt tên biến quá ngắn hay quá dài, dễ mắc lỗi khi viết nhiều lần tên biến.”
- “Khi khai báo biến cần đặc biệt lưu ý đến phạm vi giá trị của nó.” – Đàm et al., 2014, p. 23

2.4 Phép Toán, Biểu Thức, Câu Lệnh Gán

“Để mô tả các thao tác trong thuật toán, mỗi ngôn ngữ lập trình đều xác định & sử dụng 1 số khái niệm cơ bản: phép toán, biểu thức, gán giá trị cho biến. Dưới đây sẽ xét các khái niệm đó trong Pascal.” – Đàm et al., 2014, p. 24

2.4.1 Phép toán

“Tương tự trong toán học, trong các ngôn ngữ lập trình đều có những phép toán số học như cộng, trừ, nhân, chia trên các đại lượng thực, các phép toán chia nguyên & lấy phần dư, các phép toán quan hệ, ... Bảng dưới đây là ký hiệu các phép toán đó trong toán & trong Pascal:

Phép toán	Trong toán học	Trong Pascal
Các phép toán số học với số nguyên	+, −, ×, div (chia nguyên), mod (lấy phần dư)	+, −, *, div, mod
Các phép toán số học với số thực	+, −, ×, :	+, −, *, /
Các phép toán quan hệ	<, ≤, >, ≥, =, ≠	<, <=, >, >=, =, <>
Các phép toán logic	¬ (phủ định), ∨ (hoặc), ∧ (&)	not, or, and

Bảng 2.5: Ký hiệu các phép toán trong toán & trong Pascal.

Lưu ý 2.4.1. *Kết quả của các phép toán quan hệ cho giá trị logic. 1 trong những ứng dụng của phép toán logic là để tạo ra các biểu thức phức tạp từ các quan hệ đơn giản.* – Đàm et al., 2014, p. 24

2.4.2 Biểu thức số học

“Trong lập trình, biểu thức số học là 1 biến kiểu số hoặc 1 hằng số hoặc các biến kiểu số & các hằng số liên kết với nhau bởi 1 số hữu hạn phép toán số học, các dấu ngoặc tròn (&) tạo thành 1 biểu thức có dạng tương tự như cách viết trong toán học với những quy tắc sau:

- Chỉ dùng cặp ngoặc tròn để xác định trình tự thực hiện phép toán trong trường hợp cần thiết;
- Viết lần lượt từ trái qua phải;
- Không được bỏ qua dấu nhân (*) trong tích.

Các phép toán được thực hiện theo thứ tự:

- Thực hiện các phép toán trong ngoặc trước;
- Trong dãy các phép toán không chứa ngoặc thì thực hiện từ trái sang phải, theo thứ tự các phép toán nhân (*), chia (/), chia nguyên (div), lấy phần dư (mod) thực hiện trước & các phép toán cộng (+), trừ (−) thực hiện sau.”

Lưu ý 2.4.2. • “Nếu biểu thức chứa 1 hằng hay biến kiểu thực thì ta có biểu thức số học thực, giá trị của biểu thức cũng thuộc kiểu thực.

- Trong 1 số trường hợp nên dùng biến trung gian để có thể tránh được việc tính 1 biểu thức nhiều lần.” – Đàm et al., 2014, p. 25

2.4.3 Hàm số học chuẩn

“Để lập trình được dễ dàng, thuận lợi hơn, các ngôn ngữ lập trình đều có thư viện chứa 1 số chương trình tính giá trị những hàm toán học thường dùng. Các chương trình như vậy được gọi là các *hàm số học chuẩn*. Mỗi hàm chuẩn có tên chuẩn riêng. Đối số của hàm là 1 hay nhiều biểu thức số học & được đặt trong cặp ngoặc tròn (&) sau tên hàm. Bản thân hàm chuẩn cũng được coi là 1 biểu thức số học & nó có thể tham gia vào biểu thức số học như 1 toán hạng (giống như biến & hằng). Kết quả của hàm có thể là nguyên hoặc thực hay phụ thuộc vào kiểu của đối số. Bảng dưới đây cho biết 1 số hàm chuẩn thường dùng.

Hàm	Biểu diễn toán học	Biểu diễn trong Pascal	Kiểu đối số	Kiểu kết quả
Bình phương	x^2	<code>sqr(x)</code>	thực hoặc nguyên	theo kiểu của đối số
Căn bậc 2	\sqrt{x}	<code>sqr(x)</code>	thực hoặc nguyên	thực
giá trị tuyệt đối	$ x $	<code>abs(x)</code>	thực hoặc nguyên	theo kiểu của đối số
logarith tự nhiên	$\ln x$	<code>ln(x)</code>	thực	thực
Lũy thừa của số e	e^x	<code>exp()</code>	thực	thực
Sin	$\sin x$	<code>sin(x)</code>	thực	thực
Cos	$\cos x$	<code>cos(x)</code>	thực	thực

Bảng 2.6: 1 số hàm chuẩn thường dùng.

Ngoài những hàm số học chuẩn trên, còn có các hàm chuẩn khác được giới thiệu trong những phần sau.” – Đàm et al., 2014, p. 26

2.4.4 Biểu thức quan hệ

“2 biểu thức cùng kiểu liên kết với nhau bởi phép toán quan hệ cho ta 1 biểu thức quan hệ. Biểu thức quan hệ có dạng:

<biểu thức 1> <phép toán quan hệ> <biểu thức 2>

trong đó, biểu thức 1 & biểu thức 2 cùng là xâu hoặc cùng là biểu thức số học.” “Biểu thức quan hệ được thực hiện theo trình tự:

- Tính giá trị các biểu thức.
- Thực hiện phép toán quan hệ.

Kết quả của biểu thức quan hệ là giá trị logic: **true** (đúng) hoặc **false** (sai).” – Đàm et al., 2014, p. 27

2.4.5 Biểu thức logic

“*Biểu thức logic đơn giản* là biến logic hoặc hằng logic. *Biểu thức logic* là các biểu thức logic đơn giản, các biểu thức quan hệ liên kết với nhau bởi phép toán logic. Giá trị biểu thức logic là **true** hoặc **false**. Các biểu thức quan hệ thường được đặt trong cặp ngoặc (&). Phép toán *not* được viết trước biểu thức cần phủ định, e.g.: `not(x < 1)` thể hiện phát biểu “ x không nhỏ hơn 1” & điều này tương đương với biểu thức quan hệ $x \geq 1$. Các phép toán **and** & **or** dùng để kết hợp nhiều biểu thức logic hoặc quan hệ thành 1 biểu thức, thường được dùng để diễn tả các điều kiện phức tạp.” – Đàm et al., 2014, p. 28

2.4.6 Câu lệnh gán

“Lệnh gán là 1 trong những lệnh cơ bản nhất của các ngôn ngữ lập trình. Trong Pascal câu lệnh gán có dạng:

<tên biến> := <biểu thức>;

Trong trường hợp đơn giản, **tên biến** là tên của biến đơn. Kiểu của giá trị biểu thức phải phù hợp với kiểu của biến. Chức năng của lệnh gán là đặt cho biến có tên ở vế trái dấu “:=” giá trị mới bằng giá trị của biểu thức ở vế phải.” – Đàm et al., 2014, p. 28

2.5 Các Thủ Tục Chuẩn Vào/Ra Đơn Giản

“Để khởi tạo giá trị ban đầu cho biến, ta có thể dùng lệnh gán để gán 1 giá trị cho biến. Như vậy, mỗi chương trình luôn làm việc với 1 bộ dữ liệu vào. Để chương trình có thể làm việc với nhiều bộ dữ liệu vào khác nhau, thư viện của các ngôn ngữ lập trình cung cấp 1 số chương trình dùng để đưa dữ liệu vào & đưa dữ liệu ra. Những chương trình đưa dữ liệu vào có

phép đưa dữ liệu từ bàn phím hoặc từ đĩa vào gán cho các biến, làm cho chương trình trở nên linh hoạt, có thể tính toán với nhiều bộ dữ liệu đầu vào khác nhau. Kết quả tính toán được lưu trữ tạm thời trong bộ nhớ. Những chương trình đưa dữ liệu ra dùng để đưa các kết quả này ra màn hình, in ra giấy hoặc lưu trên đĩa. Các chương trình đưa dữ liệu vào & ra đó được gọi chung là các *thủ tục chuẩn vào/ra đơn giản*. Trong phần này, ta sẽ xét các *thủ tục chuẩn vào/ra đơn giản* của Pascal để nhập dữ liệu vào từ bàn phím & đưa thông tin ra màn hình.” – Đàm et al., 2014, p. 29

2.5.1 Nhập dữ liệu vào từ bàn phím

“Việc nhập dữ liệu từ bàn phím được thực hiện bằng thủ tục chuẩn: `read(<danh sách biến vào>);` hoặc `readln(<danh sách biến vào>);` trong đó *danh sách biến vào* là 1 hay nhiều tên biến đơn (trừ biến kiểu boolean). Trong trường hợp nhiều biến thì các tên biến được viết cách nhau bởi *dấu phẩy*.” – Đàm et al., 2014, p. 29

“Khi nhập giá trị cho nhiều biến, những giá trị này được gõ cách nhau bởi ít nhất 1 dấu cách hoặc ký tự xuống dòng (nhấn phím **Enter**). Các giá trị ứng với biến nguyên phải được biểu diễn dưới dạng số nguyên (không có dấu chấm thập phân). Các giá trị ứng với biến thực có thể được nhập dưới dạng số nguyên, số thực dạng thông thường hoặc số thực dạng dấu phẩy động.” – Đàm et al., 2014, p. 30

2.5.2 Đưa dữ liệu ra màn hình

“Để đưa dữ liệu ra màn hình, Pascal cung cấp thủ tục chuẩn: `write(<danh sách kết quả ra>);` hoặc `writeln(<danh sách kết quả ra>);` trong đó, *danh sách kết quả ra* có thể là *tên biến đơn*, *biểu thức* hoặc *hằng*. Các hằng xâu thường được dùng để tách các kết quả hoặc đưa ra chú thích. Các thành phần trong kết quả ra được viết cách nhau bởi dấu phẩy. Với thủ tục `write`, sau khi đưa các kết quả ra màn hình, con trỏ không chuyển xuống dòng tiếp theo. Với thủ tục `writeln`, sau khi đưa thông tin ra màn hình, con trỏ sẽ chuyển xuống đầu dòng tiếp theo.” – Đàm et al., 2014, p. 30

“Để chương trình được sử dụng 1 cách tiện lợi, khi nhập giá trị từ bàn phím cho biến, ta nên có thêm xâu ký tự nhắc nhở việc nhập giá trị cho biến nào, kiểu dữ liệu gì, ... E.g., khi cần nhập 1 số $N \in \mathbb{N}^*$, $N \leq 100$ từ bàn phím, ta có thể sử dụng cặp thủ tục sau:

```
write('Nhập số nguyên dương N <= 100: ');
readln(N);
```

” “Thủ tục `readln` cuối cùng dùng để tạm dừng thực hiện chương trình cho người dùng có thể quan sát kết quả của chương trình đưa ra trên màn hình. Muốn chương trình chạy tiếp cần nhấn phím **Enter**.

Lưu ý 2.5.1. • Các thủ tục `readln` & `writeln` có thể không có tham số.

- Trong thủ tục `write` hoặc `writeln`, sau mỗi kết quả ra (biến, hằng, biểu thức) có thể có quy cách ra. Quy cách ra có dạng: Đối với kết quả thực: : <độ rộng>:<số chữ số thập phân>. Đối với các kết quả khác: :<độ rộng> trong đó, độ rộng & số chữ số thập phân là các hằng nguyên không âm. E.g.,

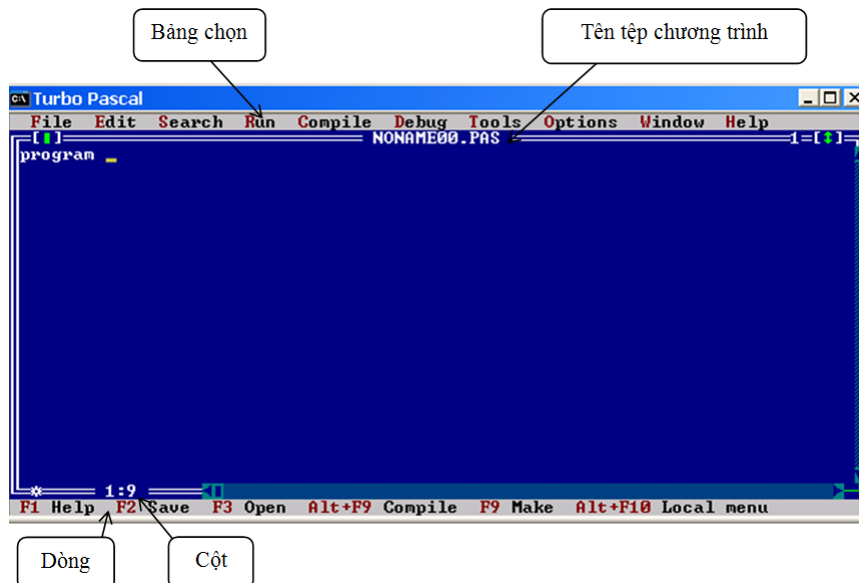
```
writeln(N:5,x:6:2);
write(i:3,j:4,a+b:8:3);
```

Trong thủ tục thứ nhất, 5 vị trí kể từ vị trí con trỏ hiện thời được dành để đưa ra giá trị N . Nếu N có giá trị nguyên dưới 5 chữ số hoặc giá trị âm dưới 4 chữ số thì những vị trí đầu sẽ được điền dấu cách. Tiếp theo là 6 vị trí được dành để đưa x ra, trong đó 2 vị trí dành để đưa ra phần thập phân. Do phần nguyên & phần thập phân được cách nhau bởi dấu chấm nên còn lại 3 vị trí cho phần nguyên.

Trong thủ tục thứ 2, i được đưa ra trên 3 vị trí, j được đưa ra trên 4 vị trí & kết quả $a + b$ được đưa ra trên 8 vị trí, trong đó có 3 vị trí dành cho phần thập phân.” – Đàm et al., 2014, p. 31

2.6 Soạn Thảo, Dịch, Thực Hiện & Hiệu Chính Chương Trình

“Để có thể thực hiện chương trình được viết bằng 1 ngôn ngữ lập trình, ta cần soạn thảo, sử dụng chương trình dịch để dịch chương trình đó sang ngôn ngữ máy. Các hệ thống lập trình cụ thể thường cung cấp phần mềm phục vụ cho việc soạn thảo, dịch & hiệu chỉnh chương trình. Với ngôn ngữ Pascal, người ta thường dùng các phần mềm Turbo Pascal 7.0 hay Free Pascal. Free Pascal cho phép tận dụng nhiều hơn khả năng của hệ thống. Tuy nhiên, cách làm việc với Turbo Pascal & Free Pascal là tương tự nên dưới đây chỉ giới thiệu cách làm việc với Turbo Pascal. Trong khuôn khổ SGK, để thực hiện các ví dụ & bài thực hành, trong máy tính cần có các tệp: `turbo.exe`, `turbo.tpl`, `graph.tpu`, `egavga.bgi`. Màn hình làm việc của Pascal có dạng như Fig. 2.1.



Hình 2.1: Màn hình làm việc của Pascal.

Dòng thứ 2 của màn hình được gọi là *thanh bảng chọn*, mỗi mục trong bảng chọn tương ứng với 1 nhóm việc ta có thể lựa chọn, 2 số ở phía dưới của màn hình ngăn cách nhau bằng dấu 2 chấm (:) cho ta biết con trỏ soạn thảo đang ở dòng nào & cột nào trên màn hình.

Dưới đây là 1 số thao tác & phím tắt thường sử dụng để soạn thảo & thực hiện 1 chương trình viết bằng Pascal.

- *Soạn thảo*: Gõ nội dung của chương trình gồm phần khai báo & các lệnh trong thân chương trình. Về cơ bản, việc soạn thảo chương trình tương tự như soạn thảo văn bản. Lưu chương trình vào đĩa bằng cách nhấn phím F2, nhập tên tệp rồi nhấn phím Enter (phần mở rộng của tệp ngầm định là .pas).
- *Biên dịch chương trình*: Nhấn tổ hợp phím Alt+F9. Nếu chương trình có lỗi cú pháp, phần mềm sẽ hiển thị 1 thông báo. Cần phải sửa lỗi nếu có, lưu lại chương trình rồi tiến hành biên dịch lại cho tới khi không còn lỗi.
- *Chạy chương trình*: Nhấn tổ hợp phím Ctrl+F9.
- *Đóng cửa sổ chương trình*: Nhấn tổ hợp phím Alt+F3.
- *Thoát khỏi phần mềm*: Nhấn tổ hợp phím Alt+X.” – Đàm et al., 2014, pp. 32–33

2.6.1 Tóm tắt

- “Dữ liệu của bài toán được biểu diễn thông qua biến trong chương trình theo các quy tắc của ngôn ngữ lập trình cụ thể.
- Kiểu dữ liệu của mọi ngôn ngữ lập trình chỉ cho phép mô tả các đại lượng rời rạc & hữu hạn.
- 1 chương trình thường có 2 phần: Phần khai báo & phần thân chương trình. Phần khai báo có thể có hoặc không.
- Kiểu dữ liệu chuẩn: Kiểu nguyên, kiểu thực, kiểu ký tự, kiểu logic.
- Các biến đều phải được khai báo & mỗi biến chỉ khai báo 1 lần.
- Các phép toán: số học, quan hệ & logic.
- Các ngôn ngữ lập trình có:
 - Lệnh gán dùng để gán giá trị của biểu thức cho biến.
 - Các thủ tục chuẩn dùng để đưa dữ liệu vào & ra.” – Đàm et al., 2014, p. 33

2.7 Bài Tập & Thực Hành

Mục đích, yêu cầu. “Giới thiệu 1 chương trình Pascal hoàn chỉnh đơn giản. Làm quen với 1 số dịch vụ cơ bản của Turbo Pascal/Free Pascal trong việc soạn thảo, lưu trữ, dịch & thực hiện chương trình.” “Dấu chấm phẩy ; dùng để ngăn cách các khai báo & các câu lệnh. Có thể bỏ qua dấu chấm phẩy sau câu lệnh trước từ khóa **end**. Sau từ khóa **end** cuối chương trình phải đặt dấu chấm.” – Đàm et al., 2014, p. 34. Xem chương trình Pascal giải phương trình bậc 2 ở link sau: [NQBH/hobby/elementary computer science/Pascal/solve_quadratic_equation.pas](#) & its executable file: [NQBH/hobby/elementary computer science/Pascal/solve_quadratic_equation](#).

Bài toán 2.7.1 (Đàm et al., 2014, 10., p. 36). *Lập trình tính \mathcal{E} đưa ra màn hình vận tốc v khi chạm đất của 1 vật rơi từ độ cao h , biết rằng $v = \sqrt{2gh}$, trong đó g là gia tốc rơi tự do $\mathcal{E} g = 9.8\text{m/s}^2$. Độ cao h (m) được nhập từ bàn phím.*

Chương 3

Cấu Trúc Rẽ Nhánh & Lặp

Nội dung. Cấu trúc rẽ nhánh & lặp trong lập trình; các câu lệnh thực hiện rẽ nhánh & lặp của Pascal.

3.1 Cấu Trúc Rẽ Nhánh

3.1.1 Rẽ nhánh

“Thường ngày, có rất nhiều việc chỉ được thực hiện khi 1 điều kiện cụ thể nào đó được thỏa mãn.” – Đàm et al., 2014, p. 38

Cách diễn đạt thuộc dạng thiếu: *Nếu ... thì ...*. Cách diễn đạt thuộc dạng đủ: *Nếu ... thì ..., nếu không thì ...*. “Từ đó có thể thấy, trong nhiều thuật toán, các thao tác tiếp theo sẽ phụ thuộc vào kết quả nhận được từ các bước trước đó. Cấu trúc dùng để mô tả các mệnh đề có dạng như trên được gọi là *cấu trúc rẽ nhánh thiếu & đủ*.” – Đàm et al., 2014, p. 38

“Mọi ngôn ngữ lập trình đều có các câu lệnh để mô tả cấu trúc rẽ nhánh.” – Đàm et al., 2014, p. 39

3.1.2 Câu lệnh if-then

“Để mô tả cấu trúc rẽ nhánh, Pascal dùng câu lệnh **if-then**. Tương ứng với 2 dạng thiếu & đủ nói ở trên, Pascal có 2 dạng câu lệnh **if-then**: • *Dạng thiếu*: **if <điều kiện> then <câu lệnh>;** • *Dạng đủ*: **if <điều kiện> then <câu lệnh 1> else <câu lệnh 2>;** trong đó: *điều kiện* là biểu thức logic, *câu lệnh*, *câu lệnh 1*, *câu lệnh 2* là 1 câu lệnh của Pascal.” – Đàm et al., 2014, p. 39 “Ở dạng thiếu: *điều kiện* sẽ được tính & kiểm tra. Nếu *điều kiện* đúng (có giá trị **true**) thì *câu lệnh* sẽ được thực hiện, ngược lại thì *câu lệnh* sẽ bị bỏ qua. Ở dạng đủ: *điều kiện* cũng được tính & kiểm tra. Nếu *điều kiện* đúng thì *câu lệnh 1* sẽ được thực hiện, ngược lại thì *câu lệnh 2* sẽ được thực hiện.” – Đàm et al., 2014, p. 40

Ví dụ 3.1.1 (Tìm max). “Để tìm số lớn nhất max trong 2 số a & b, có thể thực hiện 2 cách sau: • Dùng câu lệnh gán $\text{max} := a$ & lệnh **if-then** dạng thiếu: **if b > a then max:=b;** • Dùng 1 lệnh **if-then** dạng đủ: **if b > a then max:=b else max:=a;**” – Đàm et al., 2014, p. 40

3.1.3 Câu lệnh ghép

“Theo cú pháp, sau 1 số từ khóa (như **then** hoặc **else**) phải là 1 câu lệnh. Nhưng trong nhiều trường hợp, các thao tác sau những tên dành riêng đó khá phức tạp, đòi hỏi không phải chỉ 1 mà là nhiều câu lệnh để mô tả. Trong các trường hợp như vậy, ngôn ngữ lập trình cho phép gộp 1 dãy câu lệnh thành 1 *câu lệnh ghép* (hoặc câu lệnh hợp thành). E.g., câu lệnh ghép của Pascal có dạng:

```
begin
    <các câu lệnh>;
end;
```

Câu lệnh, *câu lệnh 1*, *câu lệnh 2* trong các câu lệnh **if-then** ở mục trên có thể là câu lệnh ghép. Thuật ngữ *câu lệnh* được hiểu chung cho câu lệnh đơn & câu lệnh ghép.” – Đàm et al., 2014, p. 40

3.2 Cấu Trúc Lặp

Chương 4

Kiểu Dữ Liệu Có Cấu Trúc

4.1 Kiểu Mảng

4.2 Kiểu Xâu

4.3 Kiểu Bản Ghi

Chương 5

Tập & Thao Tác với Tập

5.1 Kiểu Dữ Liệu Tập

5.2 Thao Tác với Tập

5.3 Ví Dụ Làm Việc với Tập

Chương 6

Chương Trình Con & Lập Trình Có Cấu Trúc

6.1 Chương Trình Con & Phân Loại

6.2 Ví Dụ về Cách Viết & Sử Dụng Chương Trình Con

6.3 Ai Là Lập Trình Viên Đầu Tiên?

6.4 Thư Viện Chương Trình Con Chuẩn

6.5 Âm Thanh

Phụ lục A

Miscellaneous

- A.1 1 Số Phép Toán Thường Dùng
- A.2 Giá Trị Phép Toán Logic
- A.3 Môi Trường Turbo Pascal
- A.4 1 Số Tên Dành Riêng
- A.5 1 Số Kiểu Dữ Liệu Chuẩn
- A.6 1 Số Thủ Tục & Hàm Chuẩn
- A.7 Câu Lệnh Rẽ Nhánh & Lặp
- A.8 Câu Lệnh with
- A.9 1 Số Thông Báo Lỗi
- A.10 Câu Lệnh Rẽ Nhánh & Lặp Trong C++

Phụ lục B

Lazarus on Ubuntu

B.1 Definition of lazarus?

Definition B.1.1 (Lazarus). “Lazarus is an IDE to create (graphical & console) applications with Free Pascal, the (L)GPLed Pascal & Object Pascal compiler that runs on Windows, Linux, Mac OS X, FreeBSD & more.” – *Installati.one/how to install lazarus on Ubuntu 22.04*

See also, e.g., [Wikipedia/Lazarus \(IDE\)](#).

Tài liệu tham khảo

Đàm, Hồ Sĩ et al. (2014). *Tin Học 11*. Tái bản lần thứ 4. Nhà Xuất Bản Giáo Dục Việt Nam, p. 144.