LẬP TRÌNH C

**MỤC LỤC**

[CHƯƠNG 1: MÔI TRƯỜNG 1](#_Toc164641063)

[1. GNU (GNU's Not Unix) 1](#_Toc164641064)

[2. GCC (GNU Compiler Collection) 1](#_Toc164641065)

[3. Trình biên dịch g++ 1](#_Toc164641066)

[4. Trình biên dịch gcc 2](#_Toc164641067)

[5. MinGW (Minimalist GNU for Windows) 2](#_Toc164641068)

[6. Cygwin 2](#_Toc164641069)

[CHƯƠNG 2: BUILD CHƯƠNG TRÌNH C/C++ BẰNG GCC 3](#_Toc164641070)

[1. Giới thiệu 3](#_Toc164641071)

[2. Kiểm tra hệ thống đã cài đặt trình biên dịch chưa? 3](#_Toc164641072)

[3. Biên dịch mã nguồn C/C++ với gcc hay g++ 4](#_Toc164641073)

[3.1. Lệnh build chương trình C bằng GCC 4](#_Toc164641074)

[3.2. Lệnh build chương trình C bằng G++ 4](#_Toc164641075)

[CHƯƠNG 3: BUILD CHƯƠNG TRÌNH C/C++ BẰNG GCC - NÂNG CAO 5](#_Toc164641076)

[1. Nhắc lại về kiến thức cũ 5](#_Toc164641077)

[2. Các bước để build 1 chương trình C/C++ 5](#_Toc164641078)

[3. Build module C/C++ với GCC 6](#_Toc164641079)

[4. Link thư viện với GCC 8](#_Toc164641080)

[4.1. Sơ lược thư viện liên kết tĩnh - Static Library 8](#_Toc164641081)

[4.2. Sơ lược thư viện liên kết động - Dynamic Library 8](#_Toc164641082)

[5. Bài tập 8](#_Toc164641083)

[CHƯƠNG 4: BẮT ĐẦU VỚI NGÔN NGỮ C 11](#_Toc164641084)

[1. Hello World 11](#_Toc164641085)

[2. Chỉnh sửa chương trình 13](#_Toc164641086)

[3. Biên dịch và chạy chương trình 13](#_Toc164641087)

[3.1. Biên dịch bằng GCC 13](#_Toc164641088)

[3.2. Sử dụng trình biên dịch clang 13](#_Toc164641089)

[3.3. Sử dụng trình biên dịch Microsoft C từ dòng lệnh 14](#_Toc164641090)

[3.4. Thực hiện chương trình 14](#_Toc164641091)

[4. “HELLO WORLD!” trong K&R C 14](#_Toc164641092)

[4.1. C90 §5.1.2.2.3 Kết thúc chương trình 15](#_Toc164641093)

[4.2. C90 §6.6.6.4 Câu lệnh return 15](#_Toc164641094)

[4.3. C99 §5.1.2.2.3 Kết thúc chương trình 15](#_Toc164641095)

[CHƯƠNG 5: CHÚ THÍCH 16](#_Toc164641096)

[1. Chú thích bằng cách sử dụng tiền xử lý (Preprocessor) 16](#_Toc164641097)

[2. Chú thích được giới hạn bằng /\* \*/ 16](#_Toc164641098)

[3. Chú thích được giới hạn bằng dấu // 18](#_Toc164641099)

[4. Rủi ro có thể xảy ra do TRIGRAPH 18](#_Toc164641100)

[CHƯƠNG 6: KIỂU DỮ LIỆU 20](#_Toc164641101)

[1. Giải thích các khai báo 20](#_Toc164641102)

[2. Kiểu số nguyên có độ rộng cố định ( kể từ c99) 23](#_Toc164641103)

[3. Kiểu số nguyên và hằng số 23](#_Toc164641104)

[4. Hằng số thực 26](#_Toc164641105)

[5. Chuỗi kí tự 26](#_Toc164641106)

[CHƯƠNG 7: TOÁN TỬ 29](#_Toc164641107)

[1. Toán tử quan hệ 29](#_Toc164641108)

[1.1. Bằng “==” 29](#_Toc164641109)

[1.2. Không bằng (Khác) “!=” 29](#_Toc164641110)

[1.3. Not “!” 30](#_Toc164641111)

[1.4. So sánh lớn hơn “>” 30](#_Toc164641112)

[1.5. So sánh bé hơn “<” 30](#_Toc164641113)

[1.6. So sánh lớn hơn hoặc bằng “>=” 30](#_Toc164641114)

[1.7. So sánh bé hơn hoặc bằng “<=” 31](#_Toc164641115)

[2. Toán tử điều kiện/ toán tử 3 ngôi 31](#_Toc164641116)

[3. Toán tử BITWISE 32](#_Toc164641117)

[4. Hoạt động ngắn mạch của toán tử logic 36](#_Toc164641118)

[5. Toán tử dấu phẩy 36](#_Toc164641119)

[6. Toán tử số học 36](#_Toc164641120)

[4.6.1 Số học cơ bản 36](#_Toc164641121)

[4.6.2 Toán tử cộng 37](#_Toc164641122)

[4.6.3 Toán tử trừ 37](#_Toc164641123)

[4.6.4 Toán tử nhân 37](#_Toc164641124)

[4.6.5 Toán tử chia 38](#_Toc164641125)

[4.6.6 Toán tử modulo (Phép chia lấy dư) 40](#_Toc164641126)

[4.6.7 Toán tử tăng giảm 40](#_Toc164641127)

[7. Toán tử truy cập 42](#_Toc164641128)

[4.7.1 Thuộc tính của đối tượng 42](#_Toc164641129)

[4.7.2 Thành viên của đối tượng được trỏ tới 42](#_Toc164641130)

[4.7.3 Lấy địa chỉ 43](#_Toc164641131)

[4.7.4 Dereference 44](#_Toc164641132)

[4.7.5 Chỉ số mảng 44](#_Toc164641133)

[4.7.6 Khả năng hoán đổi của chỉ số mảng 44](#_Toc164641134)

[8. Toán tử SIZEOF 45](#_Toc164641135)

[4.8.1 With a type as operand 45](#_Toc164641136)

[4.8.2 With an expression as operand 45](#_Toc164641137)

[9. Toán tử ép kiểu 45](#_Toc164641138)

[10. Toán tử gọi hàm 46](#_Toc164641139)

[11. TOÁN TỬ TĂNG/ TOÁN TỬ GIẢM 46](#_Toc164641140)

[12. Toán tử gán 47](#_Toc164641141)

[13. Toán tử logic 49](#_Toc164641142)

[4.13.1 Logic AND 49](#_Toc164641143)

[4.13.2 Logic OR 49](#_Toc164641144)

[4.13.3 Logic NOT 50](#_Toc164641145)

[14. TOÁN HỌC TRÊN CON TRỎ 51](#_Toc164641146)

[4.14.1 Cộng con trỏ 51](#_Toc164641147)

[4.14.2 Trừ con trỏ 51](#_Toc164641148)

[15. \_Alignof 51](#_Toc164641149)

[CHƯƠNG 8: BOOLEAN 52](#_Toc164641150)

[1. SỬ DỤNG STDBOOL.H 52](#_Toc164641151)

[2. Sử dụng #DEFINE 53](#_Toc164641152)

[3. SỬ DỤNG KIỂU \_BOOL NỘI TẠI(TÍCH HỢP) 54](#_Toc164641153)

[4. Số nguyên và con trỏ trong biểu thức BOOLEAN 55](#_Toc164641154)

[5. Xác định kiểu bool bằng cách sử dụng TYPEDEF 56](#_Toc164641155)

[CHƯƠNG 9: CHUỖI 58](#_Toc164641156)

[1. PHÂN TÁCH TỪ: STRTOK(), STRTOK\_R() VÀ STRTOK\_S() 58](#_Toc164641157)

[2. STRING LITERALS 61](#_Toc164641158)

[3. Tính độ dài chuỗi: STRLEN() 62](#_Toc164641159)

[4. Giới thiệu cơ bản về chuỗi 64](#_Toc164641160)

[5. SAO CHÉP CHUỖI 65](#_Toc164641161)

[5.1. Gán con trỏ không sao chép chuỗi 65](#_Toc164641162)

[5.2. Sao chép chuỗi dùng các hàm tiêu chuẩn 66](#_Toc164641163)

[5.2.1. strcpy() 66](#_Toc164641164)

[5.2.2. snprintf() 66](#_Toc164641165)

[5.2.3. strncat() 67](#_Toc164641166)

[5.2.4. strncpy() 68](#_Toc164641167)

[6. LẶP QUA CÁC KÍ TỰ TRONG MỘT CHUỖI 69](#_Toc164641168)

[7. Tạo mảng chuỗi 70](#_Toc164641169)

[8. Chuyển đổi chuỗi thành số: ATOI(), ATOF() (CẨN THẬN KHÔNG NÊN SỬ DỤNG) 71](#_Toc164641170)

[9. Đọc ghi dữ liệu dạng chuỗi 72](#_Toc164641171)

[10. TÌM VỊ TRÍ XUẤT HIỆN ĐẦU TIÊN/ CUỐI CÙNG CỦA MỘT KÍ TỰ CỤ THỂ: STRCHR(), STRRCHR() 73](#_Toc164641172)

[11. Sao chép và nối chuỗi: STRCPY(), STRCAT() 75](#_Toc164641173)

[12. So sánh: STRCMP(), STRNCMP(), STRCASECMP(), STRNCASECMP() 77](#_Toc164641174)

[13. Chuyển đổi chuỗi thành số một cách an toàn: Hàm STRTOX 80](#_Toc164641175)

[14. STRSPN và STRCSPN 82](#_Toc164641176)

[CHƯƠNG 10: CHỮ SỐ, KÝ TỰ VÀ CHUỖI 84](#_Toc164641177)

[1. FLOATING POINT LITERALS( CHỮ SỐ DẤU CHẤM ĐỘNG) 84](#_Toc164641178)

[2. STRING LITERALS( CHUỖI KÍ TỰ CHỮ) 84](#_Toc164641179)

[3. CHARACTER LITERALS(KÝ TỰ) 85](#_Toc164641180)

[4. INTEGER LITERALS(SỐ NGUYÊN) 86](#_Toc164641181)

[CHƯƠNG 11: COMPOUND LITERALS 88](#_Toc164641182)

[1. ĐỊNH NGHĨA/ KHỞI TẠO COMPOUND LITERALS 88](#_Toc164641183)

[CHƯƠNG 12: BIT- FIENDS 91](#_Toc164641184)

[CHƯƠNG 13: MẢNG 92](#_Toc164641185)

[1. Khai báo và khởi tạo mảng 92](#_Toc164641186)

[2. Lặp qua mảng hiệu quả và thứ tự ưu tiên hàng( row major order) 94](#_Toc164641187)

[3. Độ dài mảng 96](#_Toc164641188)

[4. Truyền mảng đa chiều vào hàm 98](#_Toc164641189)

[5. Mảng đa chiều 100](#_Toc164641190)

[5.1. Mảng hai chiều 100](#_Toc164641191)

[5.1.1. Khởi tạo mảng 2 chiều 101](#_Toc164641192)

[5.1.2. Truy cập phần tử trong mảng hai chiều 102](#_Toc164641193)

[5.2. Mảng ba chiều 103](#_Toc164641194)

[5.2.1. Bộ nhớ mảng ba chiều 103](#_Toc164641195)

[5.2.2. Khởi tạo một mảng ba chiều 103](#_Toc164641196)

[6. Define mảng và truy cập vào phần tử mảng 104](#_Toc164641197)

[7. Xóa nội dung của mảng( Thiết lập giá trị bằng 0) 105](#_Toc164641198)

[8. Thiết lập giá trị trong mảng 106](#_Toc164641199)

[9. Duyệt qua mảng bằng con trỏ 108](#_Toc164641200)

[CHƯƠNG 14: LINKED LISTS 110](#_Toc164641201)

[1. DOUBLY LINKED LIST (DANH SÁCH LIÊN KẾT KÉP) 110](#_Toc164641202)

[2. REVERSING A LINKED LIST (ĐẢO NGƯỢC DANH SÁCH LIÊN KẾT) 115](#_Toc164641203)

[3. INSERTING A NODE AT THE NTH POSITION (CHÈN NÚT Ở VỊ TRÍ THỨ N) 118](#_Toc164641204)

[4. INSERTING A NODE AT THE BEGINNING OF A SINGLY LINKED LIST (CHÈN NÚT VÀO ĐẦU DANH SÁCH LIÊN KẾT ĐƠN) 121](#_Toc164641205)

[CHƯƠNG 15: ENUMERATIONS 126](#_Toc164641206)

[1. SIMPLE ENUMERATION (ENUMERATION ĐƠN GIẢN) 126](#_Toc164641207)

[2. ENUMERATION CONSTANT WITHOUT TYPENAME (HẰNG SỐ ENUMERATION KHÔNG CẦN TÊN KIỂU) 128](#_Toc164641208)

[3. ENUMERATION WITH DUPLICATE VALUE (ENUMERATION VỚI GIÁ TRỊ TRÙNG LẶP) 128](#_Toc164641209)

[4. TYPEDEF ENUM 129](#_Toc164641210)

[CHƯƠNG 16: STRUCTS (CẤU TRÚC) 132](#_Toc164641211)

[1. FLEXIBLE ARRAY MEMBERS (THÀNH PHẦN MẢNG LINH HOẠT) 132](#_Toc164641212)

[1.1. Khai báo kiểu dữ liệu 132](#_Toc164641213)

[1.2. Ảnh hưởng đến kích thước và phần đệm 133](#_Toc164641214)

[1.3. Sử dụng 133](#_Toc164641215)

[1.4. Struct hack 135](#_Toc164641216)

[1.5. Khả năng tương thích 135](#_Toc164641217)

[2. TYPEDEF STRUCTS 137](#_Toc164641218)

[3. POINTERS TO STRUCTS (CON TRỎ TỚI CẤU TRÚC) 138](#_Toc164641219)

[4. PASSING STRUCTS TO FUNCTIONS (TRUYỀN STRUCTS VÀO HÀM) 142](#_Toc164641220)

[5. OBJECT- BASED PROGRAMMING USING STRUCTS (LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG SỬ DỤNG STRUCTS) 144](#_Toc164641221)

[6. SIMPLE DATA STRUCTURES (CÁC CẤU TRÚC DỮ LIỆU ĐƠN GIẢN) 148](#_Toc164641222)

[CHƯƠNG 17: STANDARD MATH (TOÁN HỌC TIÊU CHUẨN) 150](#_Toc164641223)

[1. CÁC HÀM MŨ – POW(), POWF(), POWL() 150](#_Toc164641224)

[2. DOUBLE PRECISION FLOATING- POINT REMAINDER FMOD() (PHẦN DƯ SỐ THỰC VỚI ĐỘ CHÍNH XÁC KÉP FMOD()) 152](#_Toc164641225)

[3. SINGLE PRECISION AND LONG DOUBLE PRECISION FLOATING- POINT REMAINDER: FMODF(), FMODL() 153](#_Toc164641226)

[CHƯƠNG 18: CÁC LỆNH LẶP/ VÒNG LẶP: FOR, WHILE, DO WHILE 155](#_Toc164641227)

[1. Vòng lặp FOR 155](#_Toc164641228)

[2. LOOP UNROLLING AND DUFF’S DEVICE 156](#_Toc164641229)

[3. VÒNG LẶP WHILE 157](#_Toc164641230)

[4. VÒNG LẶP DO – WHILE 158](#_Toc164641231)

[5. VÒNG LẶP VÔ HẠN 160](#_Toc164641232)

[CHƯƠNG 19: SELECTION STATEMENTS (CÂU LỆNH LỰA CHỌN) 162](#_Toc164641233)

[1. CÂU LỆNH IF() 162](#_Toc164641234)

[2. NESTED IF()… ELSE VS IF ELSE LADDER ( CÂU LỆNH IF LỒNG NHAU VÀ CÂU LỆNH IF BẬC THANG) 163](#_Toc164641235)

[3. CÂU LỆNH SWITCH() 165](#_Toc164641236)

[4. CÂU LỆNH IF()… ELSE VÀ CÚ PHÁP 169](#_Toc164641237)

[5. NỐI HAI HOẶC NHIỀU LỆNH IF()… ELSE 170](#_Toc164641238)

[CHƯƠNG 20: INITIALIZATION (KHỞI TẠO) 172](#_Toc164641239)

[1. KHỞI TẠO BIẾN TRONG C 172](#_Toc164641240)

[1.1. Khởi tạo một mảng 173](#_Toc164641241)

[1.2. 17.1.2 Khởi tạo mảng kí tự 173](#_Toc164641242)

[2. SỬ DỤNG DESIGNATED INITIALIZERS 174](#_Toc164641243)

[2.1. Designated initializers cho các phần tử mảng 174](#_Toc164641244)

[2.2. Designated initializers cho structures 175](#_Toc164641245)

[2.3. Designated initializer cho unions 175](#_Toc164641246)

[2.4. Designated initializers cho arrays của structures, etc 177](#_Toc164641247)

[2.5. Specifying ranges in array initializers (Chỉ định phạm vi trong bộ khởi tạo mảng) 178](#_Toc164641248)

[3. INITIALIZING STRUCTURES AND ARRAYS OF STRUCTURES (KHỞI TẠO CẤU TRÚC VÀ MẢNG CẤU TRÚC) 178](#_Toc164641249)

[CHƯƠNG 21: KHAI BÁO VÀ ĐỊNH NGHĨA 180](#_Toc164641250)

[1. Hiểu về khai báo và định nghĩa 180](#_Toc164641251)

[CHƯƠNG 22: COMMAND- LINE ARGUMENTS (ĐỐI SỐ DÒNG LỆNH) 182](#_Toc164641252)

[1. In ra các đối số của chương trình và chuyển đổi thành các giá trị số nguyên 182](#_Toc164641253)

[2. IN RA CÁC ĐỐI SỐ CỦA DÒNG LỆNH 183](#_Toc164641254)

[3. SỬ DỤNG CÔNG CỤ GETOPT CỦA GNU 184](#_Toc164641255)

[CHƯƠNG 23: FILES VÀ LUỒNG INPUT/OUTPUT 186](#_Toc164641256)

[1. Mở và ghi vào tệp tin 186](#_Toc164641257)

[1.1. Đóng một tập tin 189](#_Toc164641258)

[2. RUN PROCESS (QUÁ TRÌNH CHẠY) 189](#_Toc164641259)

[3. FPRINTF 190](#_Toc164641260)

[4. LẤY CÁC DÒNG TỪ TỆP SỬ DỤNG GETLINE() 191](#_Toc164641261)

[5. FSCANF() 197](#_Toc164641262)

[6. ĐỌC CÁC DÒNG TỪ TẬP TIN 199](#_Toc164641263)

[7. MỞ VÀ VIẾT VÀO FILE NHỊ PHÂN 201](#_Toc164641264)

[CHƯƠNG 24: FORMATTED INPUT/OUTPUT (ĐỊNH DẠNG ĐẦU VÀO ĐẦU RA) 205](#_Toc164641265)

[1. CÁC BỘ CHUYỂN ĐỔI CHO VIỆC IN RA 205](#_Toc164641266)

[2. HÀM PRINTF() 207](#_Toc164641267)

[3. CÁC CỜ ĐỊNH DẠNG KHI IN 208](#_Toc164641268)

[4. IN GIÁ TRỊ CỦA CON TRỎ TỚI ĐỐI TƯỢNG 210](#_Toc164641269)

[5. IN GIÁ TRỊ CỦA HIỆU HAI CON TRỎ ĐẾN MỘT ĐỐI TƯỢNG 212](#_Toc164641270)

[6. LENGTH MODIFIERS 213](#_Toc164641271)

[CHƯƠNG 25: CON TRỎ 216](#_Toc164641272)

[1. Giới thiệu 216](#_Toc164641273)

[2. Các lỗi thường gặp 220](#_Toc164641274)

[2.1. Không kiểm tra lỗi trong việc cấp phát bộ nhớ 220](#_Toc164641275)

[2.2. Sử dụng số nguyên cố định thay vì sizeof khi yêu cầu cấp phát bộ nhớ 220](#_Toc164641276)

[2.3. Rò rỉ bộ nhớ (Memory leaks) 221](#_Toc164641277)

[2.4. Lỗi Logic 221](#_Toc164641278)

[2.5. Tạo con trỏ trỏ tới biến được tạo trong vùng nhớ (stack) ngăn xếp 222](#_Toc164641279)

[2.6. Tăng giảm và dereferencing 225](#_Toc164641280)

[3. Dereferencing a pointer 225](#_Toc164641281)

[4. Dereferencing a pointer to a struct 226](#_Toc164641282)

[5. Con trỏ hằng 228](#_Toc164641283)

[5.1. Single Pointers 228](#_Toc164641284)

[5.2. Pointer to Pointer 229](#_Toc164641285)

[6. Con trỏ hàm 232](#_Toc164641286)

[7. Polymorphic behaviour with void pointers 234](#_Toc164641287)

[8. Toán tử địa chỉ 235](#_Toc164641288)

[9. Khởi tạo con trỏ 235](#_Toc164641289)

[10. Pointer to pointer 237](#_Toc164641290)

[11. Con trỏ void\* như là đối số và giá trị trả về của các hàm tiêu chuẩn 237](#_Toc164641291)

[12. Cùng một dấu hoa thị \* nhưng có ý nghĩa khác nhau 238](#_Toc164641292)

[CHƯƠNG 26: SEQUENCE POINTS 241](#_Toc164641293)

[1. BIỂU THỨC KHÔNG CÓ THỨ TỰ 241](#_Toc164641294)

[2. BIỂU THỨC CÓ THỨ TỰ 242](#_Toc164641295)

[3. BIỂU THỨC ĐƯỢC XÁC ĐỊNH THỨ TỰ MỘT CÁCH KHÔNG XÁC ĐỊNH 243](#_Toc164641296)

[CHƯƠNG 27: CON TRỎ HÀM 245](#_Toc164641297)

[1. GIỚI THIỆU 245](#_Toc164641298)

[2. TRẢ VỀ CON TRỎ HÀM TỪ MỘT HÀM 246](#_Toc164641299)

[3. THỰC HÀNH TỐT NHẤT 248](#_Toc164641300)

[4. GÁN CON TRỎ HÀM 250](#_Toc164641301)

[5. GHI NHỚ ĐỂ VIẾT CON TRỎ HÀM 251](#_Toc164641302)

[6. KHÁI NIỆM CƠ BẢN 252](#_Toc164641303)

[CHƯƠNG 28: THAM SỐ CỦA HÀM 255](#_Toc164641304)

[1. THAM SỐ ĐƯỢC TRUYỀN BẰNG GIÁ TRỊ (PASS BY VALUE) 255](#_Toc164641305)

[2. TRUYỀN MẢNG VÀO HÀM 256](#_Toc164641306)

[3. THỨ TỰ THỰC THI THAM SỐ TRONG HÀM 257](#_Toc164641307)

[4. SỬ DỤNG THAM SỐ CON TRỎ ĐỂ TRẢ VỀ NHIỀU GIÁ TRỊ 258](#_Toc164641308)

[5. VÍ DỤ VỀ HÀM TRẢ VỀ CẤU TRÚC CHỨA CÁC GIÁ TRỊ CÓ MÃ LỖI 258](#_Toc164641309)

[CHƯƠNG 29: TRUYỀN MỘT MẢNG 2D ĐẾN MỘT HÀM 261](#_Toc164641310)

[1. TRUYỀN MỘT MẢNG 2D ĐẾN MỘT HÀM 261](#_Toc164641311)

[2. SỬ DỤNG MẢNG PHẲNG LÀM MẢNG 2D 272](#_Toc164641312)

[CHƯƠNG 30: XỬ LÝ LỖI 275](#_Toc164641313)

[1. ERRNO 275](#_Toc164641314)

[2. STRERROR 275](#_Toc164641315)

[3. PERROR 276](#_Toc164641316)

[CHƯƠNG 31: UNDEFINED BEHAVIOR (HÀNH VI KHÔNG XÁC ĐỊNH) 277](#_Toc164641317)

[1. DEREFERENCING MỘT CON TRỎ TỚI BIẾN NGOÀI THỜI GIAN TỒN TẠI CỦA NÓ 277](#_Toc164641318)

[2. SAO CHÉP BỘ NHỚ CHỒNG CHÉO 278](#_Toc164641319)

[3. SIGNED INTEGER OVERFLOW (TRÀN SỐ NGUYÊN CÓ DẤU) 279](#_Toc164641320)

[4. SỬ DỤNG MỘT BIẾN CHƯA ĐƯỢC KHỞI TẠO 281](#_Toc164641321)

[5. DATA RACE 283](#_Toc164641322)

[6. ĐỌC GIÁ TRỊ CỦA CON TRỎ ĐÃ ĐƯỢC GIẢI PHÓNG 283](#_Toc164641323)

[7. SỬ DỤNG ĐỊNH DẠNG KHÔNG CHÍNH XÁC TRONG PRINTF 283](#_Toc164641324)

[8. SỬA ĐỔI CHUỖI KÝ TỰ 284](#_Toc164641325)

[9. TRUYỀN MỘT CON TRỎ NULL ĐỂ CHUYỂN ĐỔI PRINTF(%S) 284](#_Toc164641326)

[10. SỬA ĐỔI BẤT KÌ ĐỐI TƯỢNG NÀO NHIỀU LẦN GIỮA HAI SEQUENCE POINTS 285](#_Toc164641327)

[11. GIẢI PHÓNG BỘ NHỚ HAI LẦN 287](#_Toc164641328)

[12. DỊCH CHUYỂN BIT BẰNG CÁCH SỬ DỤNG SỐ ÂM HOẶC VƯỢT QUÁ CHIỀU RỘNG CỦA KIỂU DỮ LIỆU 287](#_Toc164641329)

[13. QUAY LẠI TỪ MỘT HÀM ĐƯỢC KHAI BÁO BẰNG BỘ XÁC ĐỊNH HÀM ‘\_NORETURN’ HOẶC ‘NORETURN’ 288](#_Toc164641330)

[14. TRUY CẬP BỘ NHỚ NGOÀI VÙNG NHỚ ĐƯỢC PHÂN BỔ 290](#_Toc164641331)

[15. THAY ĐỔI BIẾN CONST THÔNG QUA CON TRỎ 290](#_Toc164641332)

[16. ĐỌC MỘT ĐỐI TƯỢNG CHƯA KHỞI TẠO KHÔNG ĐƯỢC HỖ TRỢ BỞI BỘ NHỚ 291](#_Toc164641333)

[17. PHÉP CỘNG HOẶC PHÉP TRỪ CON TRỎ KHÔNG ĐƯỢC GIỚI HẠN ĐÚNG CÁCH 291](#_Toc164641334)

[18. DEREFERENCING CON TRỎ NULL 292](#_Toc164641335)

[19. SỬ DỤNG FFLUSH TRÊN LUỒNG ĐẦU VÀO 292](#_Toc164641336)

[20. LIÊN KẾT KHÔNG NHẤT QUÁN CỦA CÁC ĐỊNH DANH 293](#_Toc164641337)

[21. THIẾU CÂU LỆNH RETURN TRONG CÂU LỆNH TRẢ VỀ GIÁ TRỊ 294](#_Toc164641338)

[22. CHIA CHO SỐ 0 295](#_Toc164641339)

[23. CHUYỂN ĐỔI GIỮA CÁC LOẠI CON TRỎ TẠO RA KẾT QUẢ ĐƯỢC CĂN CHỈNH KHÔNG CHÍNH XÁC 296](#_Toc164641340)

[24. SỬA ĐỔI CHUỖI ĐƯỢC TRẢ VỀ BỞI CÁC HÀM GETENV, STRERROR, VÀ SETLOCALE 297](#_Toc164641341)

[CHƯƠNG 32: TẠO SỐ NGẪU NHIÊN 298](#_Toc164641342)

[1. TẠO SỐ NGẪU NHIÊN CƠ BẢN 298](#_Toc164641343)

[2. TRÌNH TẠO ĐỒNG DƯ ĐƯỢC HOÁN VỊ 299](#_Toc164641344)

[3. THẾ HỆ XORSHIFT 300](#_Toc164641345)

[4. HẠN CHẾ TẠO TRONG MỘT PHẠM VI NHẤT ĐỊNH 301](#_Toc164641346)

[CHƯƠNG 33: TIỀN XỬ LÝ VÀ MACRO 303](#_Toc164641347)

[1. HEADER INCLUDE GUARDS 303](#_Toc164641348)

[2. SỬ DỤNG #IF 0 ĐỂ CHẶN MÃ TRONG CÁC PHẦN 307](#_Toc164641349)

[3. MACROS CÓ HÌNH THỨC CỦA HÀM 308](#_Toc164641350)

[4. CHÈN CÁC TỆP NGUỒN 310](#_Toc164641351)

[5. CONDITIONAL INCLUSION AND CONDITIONAL FUNCTION SIGNATURE MODIFICATION 311](#_Toc164641352)

[6. \_ \_ CPLUSPLUS ĐỂ SỬ DỤNG CÁC PHẦN TỬ C NGOÀI MÃ C ++ KHI BIÊN DỊCH C++ BẰNG TÊN BỊ MÃ HÓA 315](#_Toc164641353)

[7. DÁN MÃ THÔNG BÁO 317](#_Toc164641354)

[8. MACRO ĐƯỢC XÁC ĐỊNH TRƯỚC 317](#_Toc164641355)

[8.1. Macro xác định trước bắt buộc 318](#_Toc164641356)

[8.2. Macro được xác định trước khác (không bắt buộc) 318](#_Toc164641357)

[9. MACRO VỚI ĐỐI SỐ BIẾN THIÊN 320](#_Toc164641358)

[10. THAY THẾ MACRO 321](#_Toc164641359)

[11. CHỈ THỊ LỖI 323](#_Toc164641360)

[12. KHAI TRIỂN FOREACH 323](#_Toc164641361)

# MÔI TRƯỜNG

GNU là dự án phần mềm miễn phí, cung cấp nhiều công cụ và phần mềm quan trọng như GCC (bao gồm 'gcc' và 'g++').

Trong đó MinGW và Cygwin là hai môi trường phát triển phần mềm khác nhau trên Windows và Linux, đều sử dụng các công cụ từ dự án GNU, bao gồm như GCC, GDB, để biên dịch mã nguồn C/C++ thành chương trình thực thi tương ứng.

## GNU (GNU's Not Unix)

GNU (GNU's Not Unix) là một dự án phần mềm tự do, được khởi xướng bởi Richard Stallman vào năm 1983. Mục tiêu của dự án là phát triển một hệ điều hành giống Unix hoàn toàn miễn phí và mã nguồn mở. Dự án GNU cung cấp nhiều phần mềm quan trọng như:

* GCC (GNU Compiler Collection) - Bộ công cụ biên dịch cho nhiều ngôn ngữ lập trình
* GNU Binutils - Các công cụ để xử lý tệp đối tượng
* GNU Make - Công cụ tự động hóa việc biên dịch
* GNU Bash - Trình thông dịch dòng lệnh
* GNU Emacs - Trình soạn thảo văn bản
* GNU Debugger (GDB) - Trình gỡ lỗi
* GNU Core Utils - Các tiện ích cơ bản cho dòng lệnh
* Và nhiều phần mềm khác...

Tất cả các phần mềm của GNU đều có mã nguồn mở và được phát hành dưới Giấy phép Công cộng GNU (GNU General Public License - GPL).

## GCC (GNU Compiler Collection)

* GCC là một bộ công cụ biên dịch mã nguồn mở, được phát triển bởi dự án GNU.
* GCC hỗ trợ biên dịch cho nhiều ngôn ngữ lập trình khác nhau như C, C++, Fortran, Ada, Go, và nhiều ngôn ngữ khác.
* Trong GCC, có các trình biên dịch riêng cho từng ngôn ngữ lập trình, ví dụ như 'gcc' cho C và 'g++' cho C++.

## Trình biên dịch g++

* g++ là tên của trình biên dịch C++ trong bộ công cụ GNU Compiler Collection (GCC).
* g++ được sử dụng để biên dịch mã nguồn C++ thành mã máy có thể thực thi.
* Cả MinGW và Cygwin đều bao gồm g++ trong bộ công cụ của chúng.

## Trình biên dịch ****gcc****

* gcc là tên của trình biên dịch C trong bộ công cụ GNU Compiler Collection (GCC).
* gcc được sử dụng để biên dịch mã nguồn C thành mã máy có thể thực thi.
* Giống như g++, cả MinGW và Cygwin đều bao gồm gcc trong bộ công cụ của chúng.

## MinGW (Minimalist GNU for Windows)

* MinGW là một môi trường phát triển phần mềm miễn phí và mã nguồn mở cho Windows.
* Nó cung cấp một bộ công cụ nhỏ gọn bao gồm trình biên dịch GCC (đặc biệt là g++ và gcc), trình liên kết, thư viện và các công cụ hỗ trợ khác từ dự án GNU.
* Khi sử dụng MinGW, bạn có thể biên dịch mã nguồn C/C++ thành các chương trình thực thi năng (executable) cho Windows, không cần bất kỳ thành phần bổ sung nào khác.

## Cygwin

* Cygwin là một môi trường giả lập Unix/Linux cho Windows, cung cấp một lớp tương thích để chạy các ứng dụng Unix/Linux trên Windows.
* Cygwin bao gồm một bộ công cụ GNU đầy đủ, bao gồm trình biên dịch GCC, và cung cấp một môi trường dòng lệnh giống Unix.
* Ứng dụng được biên dịch bằng Cygwin yêu cầu cài đặt Cygwin trên máy chạy để có thể chạy đúng cách.
* Khi sử dụng Cygwin, bạn có thể biên dịch mã nguồn C/C++ thành các chương trình thực thi cho môi trường Cygwin, nhưng cần cài đặt Cygwin trên máy chạy để chương trình có thể hoạt động đúng cách.

# BUILD CHƯƠNG TRÌNH C/C++ BẰNG GCC

## Giới thiệu

Nhìn lại quá khứ, lúc mà các IDE còn chưa phát triển, lập trình viên phải thủ công build source code thông qua Terminal với các trình biên dịch C/C++ như gcc, g++. Thời đại công nghệ hiện nay có vô số IDE hỗ trợ build source code C/C++ dễ dàng.

|  |
| --- |
| **Câu hỏi:** Vậy tại sao ngày nay các IDE hỗ trợ mạnh mẽ như vậy mà phải học lại cách build source code C/C++ thủ công ?  **Trả lời:** Vẫn còn nhiều nền tảng không hỗ trợ các IDE như Server, không cần thiết phải có UI vì toàn bộ năng lực máy tính cần tập trung vào xử lý. |

## Kiểm tra hệ thống đã cài đặt trình biên dịch chưa?

Sử dụng lệnh version để kiểm tra phiên bản của trình biên dịch gcc hay g++ trong hệ điều hành Windows.

|  |
| --- |
| gcc –version  g++ –version |

Xem các tham số truyền vào khi sử dụng gcc hay g++ bằng lệnh.

|  |
| --- |
| gcc –help  g++ –help |

Hoặc đọc document về gcc hay g++.

|  |
| --- |
| man gcc  man g++ |

## Biên dịch mã nguồn C/C++ với gcc hay g++

Bắt đầu với 1 chương trình C đơn giản: **HelloWord**.

Tạo file **main.c** có nội dung như sau:

|  |
| --- |
| #include <stdio.h>  int main(int argc, char\* arg[]) {      printf("Hello World\n");      return 0;  } |

### Lệnh build chương trình C bằng GCC

Cú pháp:

|  |
| --- |
| gcc <source file> -o <outputfile> |

Cụ thể: **gcc main.c -o main**

Sau khi gọi lệnh này, gcc sẽ build ra file chương trình với tên là **main** và có khả năng thực thi. Để thực thi chương trình **main**, gõ lệnh như sau: **./main**

Kết quả xuất ra sẽ là: Hello World

Ngoài ra có thể sử dụng 1 trong các lệnh bên dưới.

**gcc -o <output file> <source file>**

hoặc

**gcc -Wall <source file> -o <outputfile>**

hoặc

**gcc -Wall -o <output file> <source file>**

* **-Wall**: sẽ in ra tất cả các warning trong quá trình build.
* **-o**: có nghĩa là build source file ra file thực thi.

### Lệnh build chương trình C bằng G++

Nếu build với g++ hoàn toàn tương tự chỉ thay thế gcc bằng g++.

Cụ thể: **g++ main.c -o main**

Tiến hành build lại để có file thực thi **main**. Và để chạy chương trình sử dụng lệnh:**./main**

# BUILD CHƯƠNG TRÌNH C/C++ BẰNG GCC - NÂNG CAO

**Tóm tắt**: Hướng dẫn build chương trình C/C++ bằng GCC thông qua Terminal với project nhiều file và link thư viện. Trong hầu hết project C/C++ gồm nhiều file và nhiều module, sử dụng thêm các thư viện liên kết, và GCC có thể hỗ trợ build 1 chương trình như vậy.

## Nhắc lại về kiến thức cũ

Để build 1 chương trình, sử dụng cú pháp như sau:

**gcc -Wall -o <file thực thi> <các file nguồn>**

* **-Wall**: chỉ định gcc xuất ra tất cả các warning trong quá trình build.
* **-o**: chỉ định gcc build source file thành file thực thi (excutable).
* **<file thực thi>:** là tên file output sau quá trình build và link các thư viện.
* **<các file nguồn>:** các file nguồn đầu vào (.c/.cpp) để build.

Hoặc

**gcc -c <các source cần build>**

Để build source file thành các file object (chưa build và link ra file thực thi).

**gcc -Wall -o main main.c**

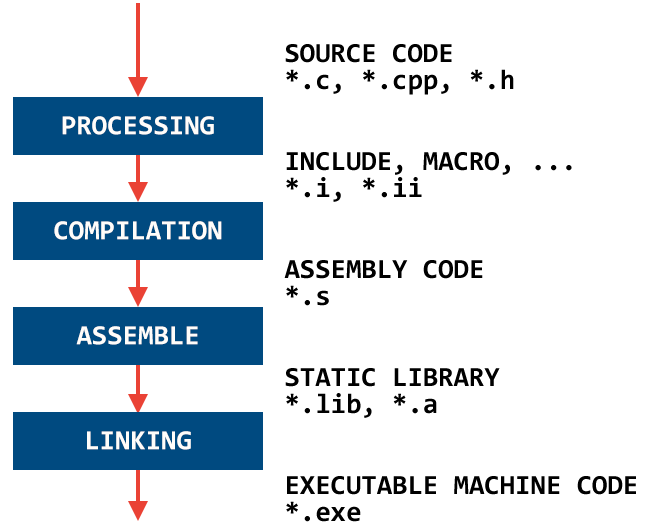
Hoặc

**gcc -c main.c**

**gcc -Wall -o main main.o**

Và để chạy chương trình sử dụng lệnh: **./main**

## Các bước để build 1 chương trình C/C++



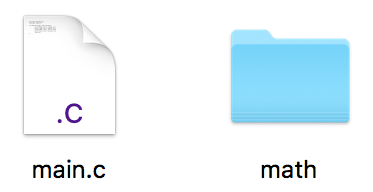
**Hình 1:** Các bước để build 1 chương trình

Quá trình build 1 chương trình trải qua 4 bước như dưới đây:

* Bước 1: Preprocessing - Bộ tiền xử lý thay thế giá trị của các directive: #define, #include, #if, ... trong file mã nguồn .h và .cpp.
* Bước 2: Compilation - Trình biên dịch chuyển mã nguồn thành mã assembly.
* Bước 3: Assemble - Chuyển từ mã assembly sang mã nhị phân (mã máy).
* Bước 4: Linking - Liên kết các module và các thư viện liên kết thành file thực thi.

## Build module C/C++ với GCC

Giả sử trong Project có 1 module **math**, chức năng tính toán cộng trừ nhân chia.



Trong thư mục **math** có 2 file là **math.h** định nghĩa prototype của các hàm và **math.c** hiện thực các prototype trong **math.h**.

**File math.h**

|  |
| --- |
| #ifndef \_\_MATH\_H\_\_  #define \_\_MATH\_H\_\_  int add(int a, int b);  int sub(int a, int b);  int mul(int a, int b);  int div(int a, int b);  #endif |

**File math.c**

|  |
| --- |
| #include "math.h"  int add(int a, int b) {      return a + b;  }  int sub(int a, int b) {      return a - b;  }  int mul(int a, int b) {      return a \* b;  }  int div(int a, int b) {      return a / b;  } |

**File main.c**

|  |
| --- |
| #include <stdio.h>  #include "math/math.h"  int main(int argc, char\* argv[]) {      printf("Add: %d\n", add(5, 6));      printf("Sub: %d\n", sub(10, 6));      printf("Mul: %d\n", mul(5, 40));      printf("Div: %d\n", div(60, 6));      return 0;  } |

Trong file **main.c** sử dụng module math, vì vậy trước khi build **main.c** phải build **math.c** sau đó link **math.o** (được build từ math.c).

* Bước 1: build module math
* gcc -c math/math.c
* Sau khi build sẽ được file **math.o**
* Bước 2: link **math.o** và **main.c**
* gcc -Wall -o main math.o main.c
* Sau khi chạy xong sẽ tạo được file thực thi file **main** bằng lệnh: **./main**

**Kết quả:**

|  |
| --- |
| Add: 11  Sub: 4  Mul: 200  Div: 10 |

## Link thư viện với GCC

### Sơ lược thư viện liên kết tĩnh - Static Library

Trong Linux thư viện liên kết tĩnh có phần mở rộng là **.a**, khi link thư viện liên kết tĩnh với mã để tạo ra file thực thi thì GCC sẽ nạp tất cả mã của thư viện liên kết tĩnh vào file thực thi, vì vậy kích thước của file thực thi lúc này sẽ tăng lên.

### Sơ lược thư viện liên kết động - Dynamic Library

Trong Linux thư viện liên kết động có phần mở rộng là **.so**, khác với thư viện liên kết tĩnh, khi link thư viện liên kết động mã thực thi sẽ không link vào file thực thi. Khi thực thi chương trình mới tìm các thư viện liên kết động và nạp vào bộ nhớ. Vì vậy mà file thực thi sau khi link xong sẽ không gia tăng kích thước từ thư viện.

Cả 2 thư viện này đều chứa mã nhị phân và có những file header **.h** chứa các prototype của các hàm trong thư viện.

Đa số các thư viện lớn như FFmpeg, OpenCV, SDL sau khi build xong sẽ tạo thư viện liên kết động và cả thư viện liên kết tĩnh.

## Bài tập

Dưới đây là 1 đoạn mã sử dụng OpenCV.

|  |
| --- |
| #include <iostream>  #include <opencv2/core/core.hpp>  #include <opencv2/highgui/highgui.hpp>  using namespace std;  using namespace cv;  int main(int argc, char\* argv[]) {      cout << Mat::zeros(4, 4, CV\_8UC1);      return 0;  } |

***Vậy làm sao build được chương trình này?***

1. Cần chỉ định cho gcc đường dẫn chứa các file header của thư viện sử dụng **-I**
2. Cần chỉ định cho gcc đường dẫn chứa các file library của thư viện sử dụng **-L**
3. Cần link các module thư viện liên kết.

Ví dụ thư viện OpenCV sẽ có:

* /usr/local/include: đường dẫn chứa header của thư viện.
* /usr/local/lib: đường dẫn chứa lib của thư viện.

Và các module của OpenCV:

**Gõ lệnh để build như sau:**

|  |
| --- |
| g++ -Wall -o main main.cpp -I/usr/local/include -L/usr/local/lib -lopencv\_calib3d -lopencv\_core -lopencv\_features2d -lopencv\_flann -lopencv\_highgui -lopencv\_imgcodecs -lopencv\_imgproc -lopencv\_ml -lopencv\_objdetect -lopencv\_photo -lopencv\_shape -lopencv\_stitching -lopencv\_superres -lopencv\_ts -lopencv\_video -lopencv\_videoio -lopencv\_videostab |

Vì OpenCV được viết bằng C++ nên sử dụng **g++** để build thay vì dùng **gcc**:

Sau khi gõ dòng lệnh **./main** để thực thi chương trình, kết quả như sau:

|  |
| --- |
| [ 0, 0, 0, 0;  0, 0, 0, 0;  0, 0, 0, 0;  0, 0, 0, 0 ] |

Một nhược điểm rất lớn là cần gõ trên Terminal khá nhiều nếu project có rất nhiều file **.c**. Để khắc phục nhược điểm này có thể viết makefile.

# BẮT ĐẦU VỚI NGÔN NGỮ C

|  |  |  |
| --- | --- | --- |
| **Phiên bản** | **Tiêu chuẩn** | **Ngày xuất bản** |
| K&R | n/a | 1978-02-22 |
| C89 | ANSI X3.159-1989 | 1989-12-14 |
| C90 | ISO/IEC 9899:1990 | 1990-12-20 |
| C95 | ISO/IEC 9899/AMD1:1995 | 1995-03-30 |
| C99 | ISO/IEC 9899:1999 | 1999-12-16 |
| C11 | ISO/IEC 9899:2011 | 2011-12-15 |

## Chương trình Hello World

Để tạo một chương trình C đơn giản in dòng chữ "Hello World " lên màn hình, sử dụng trình soạn thảo văn bản để tạo file mới (ví dụ: hello.c — phần mở rộng file phải là .c) chứa mã nguồn sau:

**File hello.c**

|  |
| --- |
| # include<stdio.h>  int main(int argc, char\* argv[]) {      puts("Hello World");      return 0;  } |

**Hãy xem từng dòng trong chương trình đơn giản này**

|  |
| --- |
| # include<stdio.h> |

Dòng mã này yêu cầu trình biên dịch cần thêm nội dung của tệp tiêu đề thư viện chuẩn **stdio.h** vào chương trình. Tệp tiêu đề thường chứa các khai báo của các hàm, macros và kiểu dữ liệu và bạn cần phải thêm vào tệp tiêu đề trước khi sử dụng chúng. Dòng này thêm vào thư viện **stdio.h** để nó có thể gọi hàm **puts().**

|  |
| --- |
| int main(int argc, char\* argv[]) |

Đây là hàm **main** , và đây là điểm bắt đầu được thực thi khi bạn chạy chương trình.

Các tham số của hàm main :

1. **int argc** : Đây là số lượng các tham số được truyền vào chương trình khi nó được chạy. argc viết tắt của “argument count” (số lượng tham số). Nó là một số nguyên (integer) và luôn luôn ít nhất bằng 1, vì chương trình luôn tính là một tham số (tên chương trình).
2. **char \*argv** : Đây là một mảng chứa các chuỗi (mảng ký tự) đại diện cho các tham số được truyền vào chương trình. argv viết tắt của “argument vector” (vector tham số). Điều quan trọng cần lưu ý là argv là một mảng hai chiều (một mảng các con trỏ đến mảng ký tự). Mỗi phần tử trong mảng argv là một con trỏ đến một chuỗi (cũng là một mảng ký tự).

|  |
| --- |
| {  ...  } |

Dấu ngoặc nhọn (curly braces) được sử dụng thành cặp để chỉ định vị trí bắt đầu và kết thúc một khối mã (khối lệnh). Chúng có thể được sử dụng theo nhiều cách khác nhau, nhưng trong trường hợp này, chúng chỉ định vị trí bắt đầu và kết thúc của một hàm.

|  |
| --- |
| puts("Hello World"); |

Dòng này gọi hàm puts() để đưa ra văn bản ra đầu ra tiêu chuẩn (mặc định là màn hình), tiếp theo là một dòng mới.

Chuỗi cần đưa ra được bao gồm trong cặp dấu ngoặc đơn(). "Hello, World" là chuỗi sẽ được ghi ra màn hình. Trong ngôn ngữ C, mọi giá trị chuỗi phải nằm trong dấu ngoặc kép "..."

Trong các chương trình C, mỗi câu lệnh cần được kết thúc bằng dấu chấm phẩy (tức là **;**).

|  |
| --- |
| return 0; |

Khi chúng ta định nghĩa hàm **main(),** chúng ta khai báo nó là một hàm trả về kiểu **int**, có nghĩa là nó cần trả về một giá trị số nguyên. Trong ví dụ này, chúng ta đang trả về giá trị nguyên là **0**, được sử dụng để chỉ rằng chương trình đã kết thúc thành công. Sau câu lệnh **return 0;**, quá trình thực thi sẽ kết thúc.

## Chỉnh sửa chương trình

* Các trình soạn thảo, chỉnh sửa văn bản đơn giản bao gồm **vim** hoặc **gedit** trên **Linux,** hoặc **Notepad** trên **Windows.** Các trình soạn thảo đa nền tảng khác bao gồm **Visual Studio Code** hoặc **Sublime Text.**
* Trình chỉnh sửa phải tạo các tệp văn bản thuần túy, không phải RTF hoặc bất kỳ định dạng nào khác.

## Biên dịch và chạy chương trình

Để chạy chương trình, file hello.c trước tiên cần được biên dịch thành file thực thi (ví dụ: **hello** trên hệ thống **Unix/Linux** hoặc **hello.exe** trên **Windows**). Điều này được thực hiện bằng cách sử dụng trình biên dịch cho ngôn ngữ C.

### Biên dịch bằng GCC

GCC **(GNU Compiler Collection)** là một trình biên dịch C được sử dụng rộng rãi. Để sử dụng nó, hãy mở **Terminal**, sử dụng dòng lệnh để điều hướng đến vị trí của tệp nguồn rồi chạy câu lệnh sau:

|  |
| --- |
| gcc hello.c -o hello |

Nếu không tìm thấy lỗi trong mã nguồn (hello.c), trình biên dịch sẽ tạo một tệp nhị phân, tên của tệp được cung cấp bởi đối số cho tùy chọn dòng lệnh -o (hello). Đây là tập tin thực thi cuối cùng.

Chúng tôi cũng có thể sử dụng các tùy chọn cảnh báo *-Wall -Wextra -Werror*, giúp xác định các sự cố có thể khiến chương trình bị lỗi hoặc tạo ra kết quả không mong muốn. Chúng không cần thiết cho chương trình đơn giản này nhưng đây là cách thêm chúng:

|  |
| --- |
| gcc -Wall -Wextra -Werror -o hello hello.c |

### Sử dụng trình biên dịch Clang

Để biên dịch chương trình bằng **clang**, bạn có thể sử dụng:

|  |
| --- |
| clang -Wall -Wextra -Werror -o hello hello.c |

Theo thiết kế, các tùy chọn dòng lệnh **clang** tương tự như của **GCC.**

### Thực hiện chương trình

Sau khi file được biên dịch, mở **Terminal** để thực thi chương trình bằng cách nhập:

|  |
| --- |
| ./hello |

Khi thực thi, chương trình đã biên dịch sẽ in **Hello World**, theo sau là một dòng mới, tới dấu nhắc lệnh.

# CHÚ THÍCH

Chú thích (comments) được sử dụng để chỉ điều gì đó cho người đọc mã nguồn. Chú thích được coi như một phần trống trên mã nguồn và không thay đổi bất kỳ ý nghĩa thực sự nào của mã.

Có hai cú pháp được sử dụng cho chú thích trong ngôn ngữ C, đó là **/\* \*/** (bản gốc) và **//** (bản mới hơn 1 chút). Một số hệ thống tài liệu sử dụng chú thích được định dạng đặc biệt để hỗ trợ tạo tài liệu cho mã nguồn.

## Chú thích bằng cách sử dụng tiền xử lý (Preprocessor)

Một phần lớn mã nguồn cũng có thể bị "chú thích" bằng cách sử dụng các chỉ thị tiền xử lý **#if 0** và **#endif.**

Điều này hữu ích khi mã chứa các chú thích nhiều dòng mà không thể lồng nhau nếu không có cơ chế chú thích này.

|  |
| --- |
| #if 0 /\* Starts the "comment", anything from here on is removed by preprocessor \*/  /\* A large amount of code with multi-line comments \*/  int foo() {      /\* lots of code \*/      ...        /\* ... some comment describing the if statement ... \*/      if (someTest) {          /\* some more comments \*/          return 1;      }      return 0;  }  #endif /\* 0 \*/  /\* code from here on is "uncommented" (included in compiled executable) \*/  ... |

## Chú thích được giới hạn bằng /\* \*/

Một chú thích bắt đầu với một dấu gạch chéo kề ngay sau đó là một dấu hoa thị (/\*) và kết thúc ngay khi gặp một dấu hoa thị kề ngay sau đó là một dấu gạch chéo (\*/). Mọi thứ nằm giữa các cặp ký tự này đều là chú thích và được coi là một phần trống (tức là bị bỏ qua) bởi trình biên dịch.

|  |
| --- |
| /\* Đây là một chú thích \*/ |

Chú thích ở trên là chú thích trên một dòng. Chú thích kiểu /\* này có thể trải dài trên nhiều dòng, như sau:

|  |
| --- |
| /\* Đây là một chú thích  nhiều dòng \*/ |

Mặc dù điều này không bắt buộc, quy ước phong cách thông thường với chú thích trên nhiều dòng là đặt khoảng trắng và dấu hoa thị ở các dòng sau đầu tiên và đặt /\* và \*/ trên các dòng mới, sao cho chúng xếp thành hàng:

|  |
| --- |
| /\* Đây là   \* một chú   \* thích   \* nhiều dòng  \*/ |

Những dấu hoa thị thêm không có tác dụng chức năng nào đối với chú thích vì chúng không có dấu gạch chéo liên quan.

Các chú thích kiểu /\* này có thể được sử dụng trên một dòng riêng biệt, ở cuối một dòng mã hoặc ngay cả trong các dòng mã:

|  |
| --- |
| /\* Chú thích này nằm trên một dòng riêng biệt \*/  if(x && y) { /\* Chú thích này nằm ở cuối dòng mã \*/      if ((complexCondition1) /\* Chú thích này nằm trong một dòng mã \*/          && (complexCondition2)) {      /\* Chú thích này nằm trong một câu lệnh if, trên một dòng riêng biệt \*/      }  } |

**Lưu ý:**

Chú thích không thể lồng nhau. Điều này xảy ra vì bất kỳ /\* nào tiếp theo sẽ bị bỏ qua (là một phần của chú thích) và đầu tiên \*/ gặp được sẽ được coi là kết thúc chú thích.

**VD:** Chú thích trong ví dụ dưới đây sẽ không hoạt động:

|  |
| --- |
| /\* Chú thích bên ngoài, có nghĩa là điều này bị bỏ qua => /\* chú thích nội bộ \*/ <= kết thúc chú thích, không phải chú thích này => \*/ |

Để chú thích các khối mã chứa chú thích kiểu này, mà nếu không sẽ bị lồng nhau, bạn có thể tham khảo ví dụ về Chú thích bằng cách sử dụng tiền xử lý.

## Chú thích được giới hạn bằng dấu //

***Phiên bản ≥ C99***

Phiên bản C99 đã giới thiệu việc sử dụng chú thích trên một dòng theo kiểu C++. Loại chú thích này bắt đầu bằng hai dấu gạch chéo kề nhau và kéo dài đến cuối dòng:

|  |
| --- |
| // Mỗi dòng trong số này đều là một chú thích trên một dòng  // Chú ý rằng mỗi dòng phải bắt đầu bằng  // Hai dấu gạch chéo kép liên tiếp (//) |

Loại chú thích này có thể được sử dụng trên một dòng riêng biệt hoặc ở cuối một dòng mã. Tuy nhiên, do chú thích này kéo dài đến cuối dòng, nên chúng không thể được sử dụng trong một dòng mã.

|  |
| --- |
| // Chú thích này nằm trên một dòng riêng biệt  if (x && y) { // Chú thích này nằm ở cuối dòng      // Chú thích này nằm trong một câu lệnh if, nằm trên một dòng  } |

## Rủi ro có thể xảy ra do Trigraph

Trong lập trình C, Trigraph là một chuỗi ba ký tự đặc biệt được sử dụng để thay thế cho một ký tự đặc biệt khác mà không phải là ký tự trong bảng mã ASCII chuẩn.

Trigraph được sử dụng để tạo ra các ký tự đặc biệt như dấu ngoặc vuông, dấu nháy kép, dấu gạch ngược, v.v. trên các hệ thống hoặc thiết bị đầu cuối không hỗ trợ những ký tự đó.

Một số **Trigraph** phổ biến trong C:

* ??= thay thế cho # (ký tự để bắt đầu một dòng tiền xử lý)
* ??/ thay thế cho \ (ký tự gạch ngược)
* ??' thay thế cho ^ (ký tự dấu mũ)
* ??( thay thế cho [ (dấu ngoặc vuông mở)
* ??! thay thế cho | (ký tự ống dẫn)
* ??< thay thế cho { (dấu ngoặc nhọn mở)
* ??> thay thế cho } (dấu ngoặc nhọn đóng)
* ??- thay thế cho ~ (ký tự gạch ngang)

***Phiên bản ≥ C99***

Trong quá trình viết chú thích được giới hạn bởi //, có thể xảy ra lỗi chính tả ảnh hưởng đến hoạt động dự kiến của chú thích.

**VD:** Ta có 1 đoạn mã như sau:

|  |
| --- |
| int foo = 20; // Bắt đầu từ 20 ??/  int bar = 0;  // Dòng sau sẽ gây lỗi biên dịch (biến 'bar' chưa được khai báo)  // vì 'int bar = 0;' là một phần của chú thích trên dòng trước đó  bar += foo; |

Ta hiểu **??/** tạo thành một trigraph là dấu gạch ngược **\** (đây là ký hiệu tiếp tục dòng). Điều này có nghĩa là trình biên dịch sẽ hiểu rằng dòng tiếp theo là tiếp tục của dòng hiện tại, tức là tiếp tục của chú thích. Điều này sẽ gây ra lỗi chương trình vì biến bar chưa được khai báo.