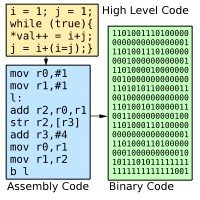
**Phần III. Lập trình**

**Câu 170.** **C.** Ngôn ngữ lập trình bậc cao

 Ngôn ngữ máy: mã máy, dưới dạng nhị phân.

Hợp ngữ: Hợp ngữ (Assembly language) là một ngôn ngữ lập trình bậc thấp, khó lập trình, hợp ngữ được dịch trực tiếp ra mã máy để máy tính thực hiện.

Ngôn ngữ lập trình bậc cao: gần với ngôn ngữ con người, được dịch ra hợp ngữ rồi dịch ra mã máy để máy tính thực hiện.

Ngôn ngữ lập trình logic: ngôn ngữ tựa lập trình, dùng để diễn tả thuật toán cho người khác hiểu.

**Câu 171. A. Trình biên dịch sẽ dịch ngôn ngữ lập trình thành mã máy. Code C được dịch sang hợp ngữ, rồi từ hợp ngữ dịch tiếp ra mã máy.**

**C là ngôn ngữ lập trình bậc cao.**

**Máy tính chỉ hiểu được mã máy (dạng nhị phân).**

**Hiện nay tồn tại nhiều phiên bản C khác nhau.**

**Câu 172. C. Thông dịch: cách thức dịch chương trình, dịch từng câu một và thực hiện luôn.**

**Câu 173. D. 02006**

**Trong C, hệ cơ số 16: 0x hoặc 0X ở đầu: 0x2006, 0X2006.**

**Hệ cơ số 8: số 0 ở đầu: 02006**

**Không có 0, 0x, 0X: hệ 10.**

**Câu 174. C. Định danh là dãy ký tự chữ cái, chữ số hoặc dấu gạch dưới trong đó ký tự đầu tiên không phải số.**

**Tên biến, và bao gồm cả tên hàm,… được gọi chung là định danh (identifier). Bạn phải tuân thủ các quy tắc sau khi đặt tên cho định danh:**

* **Trong cùng một khối mã, các biến không được trùng tên nhau. Tương tự, các hàm không được trùng tên nhau. Nhưng tên biến có thể trùng tên hàm.**
* **Tên luôn chỉ chứa ký tự hoặc số và dấu \_, không được chứa khoảng trắng và các ký tự đặc biệt khác**
* **Tên không được bắt đầu bằng số**
* **Tên có phân biệt kiểu chữ hoa thường**
* **Tên không được trùng với từ khóa:**

**auto break case char const continue default do double else enum extern float for goto if int long register return short signed sizeof static struct switch typedef union unsigned void volatile while**

* **Tên hàm cũng không nên trùng với các hàm đã có trong thư viện chuẩn của C**

**Câu 175. C. Ket Qua: chứa dấu cách.**

**Câu 176. D.** So\_Nguyen

Bien#2**: chứa ký tự đặc biệt** #

1SoNguyen**: bắt đầu bằng số**

So Nguyen**: chứa dấu cách**

**Câu 177. B.** //dòng chú thích

**Hoặc:** /\* đoạn chú thích \*/

**Câu 178. A.** #include “tên\_tệp\_tiêu\_đề”

**Trong C, có 2 cách khai báo tệp tiêu đề (thư viện):**

#include <tên\_tệp\_tiêu\_đề> ****:** chỉ dùng cho các thư viện chuẩ của C (**stdio.h conio.h math.h **...)**

#include “tên\_tệp\_tiêu\_đề”: dùng cho cả thư viện do người dùng tự xây dựng.

Chú ý: khai báo thư viện không có dấu ; và phải có dấu #

Câu 179. D. Phần khai báo hàm main(). Chương trình bắt đầu chạy từ hàm main()

**Câu 180. A. D:\TC\BIN\Tc.exe**

**Bao gồm cả đường dẫn đến file**

**Câu 181. C. Để chạy được mọt chương trình viết bằng ngôn ngữ lập trình bậc cao, trước hết phải biên dịch hoặc thông dịch. Biên dịch: dịch toàn bộ rồi chạy. Thông dịch: dịch từng lệnh một và chạy luôn.**

**Máy tính chỉ có thể chạy luôn một chương trình viết bằng ngôn ngữ máy. Hợp ngữ (Assebly) phải dịch ra mã máy mới chạy được.**

**Câu 182. A. Viết chương trình**

**Dịch chương trình là công việc của chương trình dịch (biên dịch, thông dịch).**

**Câu 183. A. Chương trình dịch**

Câu 184. B. int long float double

|  |  |  |
| --- | --- | --- |
| **Kiểu** | **Độ rộng bit** | **Miền giá trị** |
| int | 2 bytes | -32,768 tới 32,767 |
| long | 4 bytes | -2,147,483,648 tới 2,147,483,647 |
| float | 4 bytes | ±3.4E-38 tới ±3.4E+38 |
| double | 8 bytes | ±1.7E-308 tới ±1.7E+308 |

**Câu 185. D.** int1

char : trùng từ khóa

1First : bắt đầu bằng số

int : trùng từ khóa

(xem câu 174)

**Câu 186. A.** While ≠ while (C phân biệt chữ thường với chữ hoa)

if for : trùng từ khóa

(xem câu 174)

**Câu 187. B.** unsigned int n;

float int n;

unsigned float n=0; không có kiểu unsigned float

n: int; sai cú pháp

**Câu 188. C.** int n:=0;cú pháp khai báo không có dấu :

**Câu 189. C.** const char Hangso=’A’

Trong C, có 2 cách khai báo hằng số:

#define Tên\_Hằng Giá\_Trị (không có dấu = Không có dấu ; Không cần khai báo kiểu dữ liệu. Khai báo ở phần đầu chương trình)

#include <stdio.h>

#define CHIEUDAI 15

#define CHIEURONG 12

#define NEWLINE '\n'

int main()

{

int dientich;

dientich = CHIEUDAI \* CHIEURONG;

printf("Dien tich hinh chu nhat la: %d", dientich);

printf("%c", NEWLINE);

printf("===========================\n");

return 0;

}

Kết quả:

Dien tich hinh chu nhat la: 180

===========================

const Kiểu\_Dữ\_Liệu Tên\_Biến = Giá\_Trị;

#include <stdio.h>

int main()

{

const int CHIEUDAI = 15;

const int CHIEURONG = 12;

const char NEWLINE = '\n';

int dientich;

dientich = CHIEUDAI \* CHIEURONG;

printf("Dien tich hinh chu nhat la: %d", dientich);

printf("%c", NEWLINE);

printf("===========================\n");

return 0;

}

Kết quả:

Dien tich hinh chu nhat la: 180

===========================

#define HangSo A; Thừa dấu ; và lỗi ở A, do không xác định kiểu dữ liệu, sửa thành “A” (kiểu xâu) hoặc ‘A’ (kiểu ký tự)

#define HangSo ‘A’; thừa dấu ;

const char HangSo A; kiểu char nên phải dùng ‘A’

**Câu 190. B.** #define HangSo ‘A’

#define HangSo Asửa A thành‘A’hoặc“A”

const char HangSo A; sửa thành:

const char HangSo=’A’;

const char HangSo=A;sửa thành:

const char HangSo=’A’;

(xem bài 189)

Câu 191. A. #define HangSo 2.78

#define HangSo=2.78 thừa dấu =

const float HangSo 2.78 thiếu dấu =

const HangSo=2.78; thiếu kiểu dữ liệu

(xem câu 189)

**Câu 192. C.** const float HangSo=2.78;

#define float HangSo 2.78thừa kiểu dữ liệu

#define HangSo = 2.78thưa dấu =

const HangSo=2.78;thiếu kiểu dữ liệu

(xem câu 189)

**Câu 193. D.** #define MAX 10

#define MAX 10; thừa dấu ;

#define MAX=10 thừa dấu =

const int MAX 10; thiếu dấu =

**Câu 194. B.** Dòng 1, 2

1: #define PI 3.1415; thừa dấu ;

2: const float g 9.81; thiếu dấu =

3: int i=0; đúng

4: int j=’a’; đúng. Khi lưu 1 ký tự tức là lưu mã ASCII của nó. Ở đây, j=97 (mã ASCII của a là 97)

**Câu 195. A.** A%B . Phép toán % chỉ dùng cho 2 số nguyên

**Câu 196. A.** 1.5

3/2 là phép chia lấy phần nguyên, kết quả được 1

3.0/2 là phép chia số thực cho số nguyên, được 1.5

**Câu 197. B.** 4.5

1.5 \* (11/3)

Thực hiện phép chia trong ngoặc trước: 1.5 \* 3

Và kết quả là: 4.5

**Câu 198. A.** =

=là phép gán

<, >, >= là các toán tử quan hệ

Các toán tử quan hệ (phép so sánh):

|  |  |  |
| --- | --- | --- |
| **Toán tử** | **Miêu tả** | **Ví dụ** |
| == | Kiểm tra nếu hai toán hạng có bằng nhau hay không. Nếu bằng nhau thì điều kiện là true, khác nhau là false | (3 == 5) là false |
| != | Kiểm tra nếu giá trị hai toán hạng khác nhau không, nếu các giá trị là không bằng nhau thì điều kiện là true | (3 != 5) là true |
| > | Kiểm tra nếu toán hạng bên trái có lớn hơn toán hạng bên phải không, nếu có thì điều kiện là true | (2>6) là false |
| < | Kiểm tra nếu toán hạng bên trái có nhỏ hơn toán hạng bên phải không, nếu có thì điều kiện là true | (2 < 6) là true |
| >= | Kiểm tra nếu toán hạng bên trái có lớn hơn hoặc bằng toán hạng bên phải không, nếu có thì điều kiện là true | (3 >= 4) là false |
| <= | Kiểm tra nếu toán hạng bên trái có nhỏ hơn hoặc bằng toán hạng bên phải không, nếu có thì điều kiện là true | (3<= 4) là true |

**Câu 199. A.** == là phép so sánh bằng

!= là phép so sánh khác

= là phép gán

**Câu 200. A.** a=b;

**Câu 201. D.** Biểu thức

a + b là biểu thức, có toán tử là +, 2 toán hạng là a và b

**Câu 202. C.** a-b=0 **.** Trước phép gán (dấu =) phải là 1 biến

**Câu 203. B.**0

f=( (a==b) || (b>4) )

Ta thực hiện các biểu thức trong ngoặc trước:

(a==b) tức là (7==2) : false, nên được kết quả là 0

(b>4) tức là (2>4) : false, nên được kết quả là 0

Vậy f=(0 || 0) nghĩa là 0 hoặc 0, kết quả được 0. (giống các phép toán logic mệnh đề môn đại số)

**Câu 204. C.** 1

Phép so sánh < có thứ tự ưu tiên cao hơn gán = nên ta tính < trước

Đối với phép toán < , ta tính từ trái qua phải

1 < 7 < 5

1 < 5 1<7 là đúng (true), nên có giá trị là 1

1. 1<5 là đúng (true), nên có giá trị là 1

Đối với phép toán gán =, ta gán từ phải qua trái

a = a = 1

a = 1

**Câu 205. B.** 1

7+5 && 4 <= 1+3-2/3 || 5 < 2+1

12 && 4 <= 4 – 0 || 5 < 3

12 && 4 <= 4 || 5 < 3

12 && 1 || 0

1 || 0

1

Thứ tự thực hiện: các phép toán thông thường 🡪 các phép so sánh 🡪 các phép logic 🡪 phép gán

**Câu 206. D.** a=4

(b>c) sai nên thực hiện a=++c; (dạng tiền tố, c=c+1 trước, rồi mới gán a=c )

Vậy a=4, b=2, c=4

**Câu 207. B.** a=7

Toán tử ?: thực hiện từ trái qua phải

a = b ? c : d?e:f

b=0 (false) nên thực hiện a=d?e:f

d=6 (true) nên thực hiên a=e. Vậy a=7

Câu 208.B D đều đúng

ch==’a’||‘b’ phép so sánh == thực hiện trước (ch==’a’) || (‘b’)

‘b’=98 (true) nên biểu thức luôn true (luôn bằng 1)

ch==’a’ | ‘b’ thực hiện ch==’a’ trước. | là phép HOẶC TRÊN TỪNG BIT TƯƠNG ỨNG. ‘b’ có giá trị khác 0 nên biểu thức luôn đúng (luôn có giá trị bằng 1)

(ch==’a’)|(ch==’b’) nếu ch=’a’ thì (ch==’a’) true, có giá trị là 1; (ch==’b’) false, có giá trị là 0. Nếu ch=’b’ thì ngược lại. Ta lấy số 0 và 1 thực hiện phép OR thì vẫn được kết quả là 1.

nếu ch khác với 2 giá trị trên, cả (ch==’a’)

và(ch==’b’) đều bằng 0, biểu thức sẽ bằng 0.

**Câu 209. A.** x=13.Ngắt bỏ phần thập phân, chỉ lấy phần nguyên

**Câu 210. B.** 3

d = b / c % a

d = 7 / 2 % 5 Thực hiện từ trái qua phải

d = 3 % 5

d = 3

**Câu 211. A.** a=7, b=9, c=-1

|  |
| --- |
| int a=7,b,c; |
| b = a++ + 1; | b=a+1; a=a+1; | b=8; a=8; |
| c = --a – b++; | a=a-1; c=a-b; b=b+1; | a=7; c=-1; b=9; |

**Câu 212. C.** a=7, b=6, c=11

|  |
| --- |
| int a=5,b,c; |
| b = a++; | b=a; a=a+1; | b=5; a=6; |
| c=a++ + ++b -1; | b=b+1; c=a+b-1; a=a+1; | b=6;c=10;a=6; |

**Câu 213. B.** 6

int a=8, b=9, c=6;

d=a/b\*c+c++; //tách thành d=a/b\*c+c; c=c+1;

d = a/b \* c + c

d = 0 \* 6 + 6

d = 6

c = c + 1

c = 7

**Câu 214. A.** 200

(b>c) false nên thực hiện gán a=200;

**Câu 215. B.** 3

a=-1 ≠ 0 nên là true 🡪 thực hiện b+=2; //b=b+2

Câu 216. C. A=3, B=3, C=4

**(A>B)** true nên thực hiện **C=A--;**(hậu tố)**.** Có thể tách thành: **C=A; A=A-1;** (C=4; A=3)

**int A=4, B=3,C;**

**C=(A>B)?A--:--B;**

**Câu 217. B.** c=(a<b)?a:b

c=(a>b)?a:b Nếu a>b thì gán c=a

Nếu a≤b thì gán c=b

* gán số lớn hơn cho c

c=(a<b)?a:b Nếu a<b thì gán c=a

Nếu a≥b thì gán c=b

* gán số bé hơn cho c

c=(a!=b)?a:b Nếu a≠b thì gán c=a

Nếu a=b thì gán c=b

c=(a|b)?a:b Nếu cả a và b đều bằng 0 thì tất cả các bit của chúng là 0, kết quả phép OR trên từng bit tương ứng (a|b) là 0 (false). Thự hiện phép gán c=b

Nếu ít nhất một trong 2 khác 0, (a|b) khác 0 (true). Thực hiện phép gán c=a

**Câu 218. A.** 0

int a=5, c=2;

float b=11;

int d=3;

d=(int)b / c % 5;

Toán tử ép kiểu có quyền ưu tiên hơn phép chia, nên thực hiện (int)b trước, được kết quả là số nguyên 11.

Vậy d=11/2 % 5= 5 % 5 = 0

**Câu 219. B.** stdio.h (standard input-output header)

**Câu 220. A.** conio.h

**Câu 221.B.** %dsố nguyên hệ thập phân

%osố nguyên hệ bát phân

%cký tự

%sxâu

**Câu 222. D.** n=10

**Câu 223. A.** □3.45

%5.2f in ra số thực, với 2 chữ số sau dấu phảy (3.45). Và dành ra 5 vị trí để in, 3.45 chiếm 4 vị trí, nên in ra 1 dấu cách trước số 3 nữa.

Câu 224. A. printf(“%7.2f”,X);

□□12.35 🡪 cần 7 vị trí, trong đó 2 vị trí sau dấu phảy 🡪 %7.2f

Câu 225. C. f=1.23

%3.2f 🡪 2 chữ số sau dấu phảy (1.23) và chừa 3 vị trí để in số, nhưng cần tối thiểu 4 vị trí nên số vẫn in ra màn hình đầy đủ cả 4 vị trí.

**Câu 226. B**

**Câu 227. C.** A 66

Trong C, khi lưu ký tự vào bộ nhớ chính là lưu mã ASCII (dưới dạng số) vào bộ nhớ. Vì vậy giữa số và kí tự có thể dùng thay cho nhau.

VD: x=’A’\*2 + 3;

Vì ‘A’=65 (65 là mã ASCII của ký tự A) nên x=65\*2+3=133

printf(“%c“,65 );🡪 in ra ký tự có mã ASCII là 65 (ký tự A)

%c in ra ký tự, %d in ra số nguyên

**Câu 228. A.** 65 66

(xem câu 227)

**Câu 229. C.** f=0.000

f=1/3; ở đây, 1/3 là phép chia lấy phần nguyên, được kết quả là 0. Kết quả này được gán cho f, nên f=0.000000 (f kiểu float nên có độ chính xác 6 số sau dấu phảy)

printf(“f=%.3”,f); 🡪 in f ra với 3 chữ số sau dấu phảy 🡪 f=0.000

**Câu 230. C.** 1

2

\n là ký tự xuống dòng

Câu 231. C. Toán tử ++ áp dụng lên một hằng số. Không thể và cũng không được thay đổi giá trị của một hằng số.

Câu 232. A. printf(“%-10.2f\n%6.3f”,x,y);

Mặc định căn lề phải, muốn căn lề trái thêm dấu –

\n ký tự xuống dòng

**Câu 233. A.** { }

**Câu 234. C.** 7 13 20

|  |  |  |
| --- | --- | --- |
| a = 13/2; |  | a=6; |
| b = ++a + 6; | a=a+1; b=a+6; | a=7; b=13; |
| c=(b>12)?(b+7):(a\*a); | c=(b+7);  //vì(b>12) true | c=20; |

**Câu 235. B.** 9

Khi khai báo biến trong khối lệnh (hoặc hàm) nào thì nó chỉ tồn tại ở đó thôi.

Biến a=5 ở ngoài và biến a=7 ở trong khối lệnh là hai biến hoàn toàn khác nhau. Biến a=7 chỉ tồn tại bên trong hai dấu { } của khối lệnh.

Khi gán a=a+4, tức là dùng biến a=5 ở ngoài, giá trị mới: 9

Câu 236. A. Tất cả các câu lệnh sử dụng cấu trúc if...else đều có thể thay thế bằng biểu thức điều kiện ?:  
**Câu 237. D.** if ((‘A’<=ch)&&(ch<=’Z’))|| ((‘a’<=ch)&&(ch<=’z’))

Khi so sánh hai ký tự, tức là so sánh mã ASCII của chúng, ký tự nào có mã ASCII lớn hơn thì lớn hơn.

(‘A’<=ch)&&(ch<=’Z’): ch có mã ASCII từ 65 đến 90, hay ch là ký tự chữ cái hoa

(‘a’<=ch)&&(ch<=’z’): ch là ký tự chữ cái thường

**Câu 238. A.** 35

if(c) //c=4 (true)

if(a>b) printf(“2”);//a>b false thực hiện else

else printf(“3”); //in ra màn hình 3

printf(“5”);//in ra màn hình 5

Câu 239. C. A=1 và B=1

int A=0, B=1;

if (A<B){ //A<B true

A=B; //A=1

B=A; //B=1

}

**Câu 240. A.** a>b

**Câu 241. B.** A=1 và B=0

int A=0, B=1, C;

if (A<B) { //A<B true

C=B; //C=1

B=A; //B=0

A=C; //A=1

}

Đoạn chương trình trên đổi giá trị của A và B cho nhau.

**Câu 242. C.** Hai nhãn case có cùng giá trị là 5

**Câu 243. D.** C=10

(A>B) false nên có giá trị là 0

Rẽ nhánh tới case 0: C=A; break;

C được gán bằng 10, rồi gặp lệnh break 🡪 thoát khỏi switch **Câu 244. C.** Nhãn case P sử dụng một giá trị không phải là một hằng số

**Câu 245. B. 4**

a=5, khác với giá trị của case 3, case 4 nên rẽ tới default: a-- , gán a=a-1

**a=5**

****int a=10;****

****a=a/2;****

switch(a){

case 3: a+=2;

case 4: a++; break;

default: a--;

}

printf(“%d”,a);

a=4

Câu 246. B. A=10

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| i+=2 | 0 | 2 | 4 | 6 | 8 | 10 (thoát) |
| A | 2 | 4 | 6 | 8 | 10 |

Vòng for lặp 5 lần (i=0,2,4,6,8).

Mỗi lần lặp, A tăng thêm 2. Vậy giá trị của A là: 0+5\*2=10

**Câu 247. B.** 20

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| i+=2 | 0 | 2 | 4 | 6 | 8 | 10 (thoát) |
| sum | 0 | 2 | 6 | 12 | 20 |

Vì i=0,2,4,6,8 nên vòng for lặp 5 lần. Vòng for đầu tiên, i=0 nên sum được gán bằng 0. Các vòng tiếp theo, sum tăng i đơn vị (sum+=i). Suy ra sum=0+2+4+6+8=20

**Câu 248. B.** 45

Khi khai báo int S; biến S được gán giá trị mặc định là 0

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i++ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 (thoát) |
| S+=1 | 1 | 3 | 6 | 10 | 15 | 21 | 28 | 36 | 45 |

Để ý thấy: i chạy từ 1🡪9. Mỗi lần lặp, S tăng thêm i đơn vị. Như vậy, S=1+2+...+9=45

**Câu 249.**  **B.** A B C D

printf(“%c”,a); in ra ký tự có mã ASCII bằng a

**Câu 250. C.** 1 4 7 10

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| for(a++) | 1 | 4 | 7 | 10 | 13 (thoát) |
| printf() | 1 | 4 | 7 | 10 |  |
| (a%3) | 1 | 1 | 1 | 1 |  |
| a+=2 | 3 | 6 | 9 | 12 |  |

%i in ra số nguyên, giống %d

(a%3)Nếu a3, a%3=0🡪false

Ngược lại, a%3≠0🡪 true🡪 a+=2;

**Câu 251. A.** c e i

c=c+i : tăng mã ASCII của c lên i đơn vị

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i=i+2 | 0 | 2 | 4 | 6 (thoát) |
| c=c+i | ‘c’ | ‘e’ | ‘i’ |

a b c d e f g h i j k l ....

+2

+4

Câu 252. A. 5 3 1

printf(“%2d”,x--); có xuất hiện dạng hậu tố x-- (dùng giá trị cũ của x trước rồi giảm x sau)

Có thể tách lệnh trên thành 2 lệnh nhỏ:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| for: x-- | 5 | 3 | 1 | -1 (thoát) |
| printf() | 5 | 3 | 1 |
| x=x-1 | 4 | 2 | 0 |

for (x=5;x>0;x--){

printf(“%2d”,x);

x=x-1;

}

**Câu 253. C.** 5

for(i=0; i<5;i++);

printf(“%d”,i);

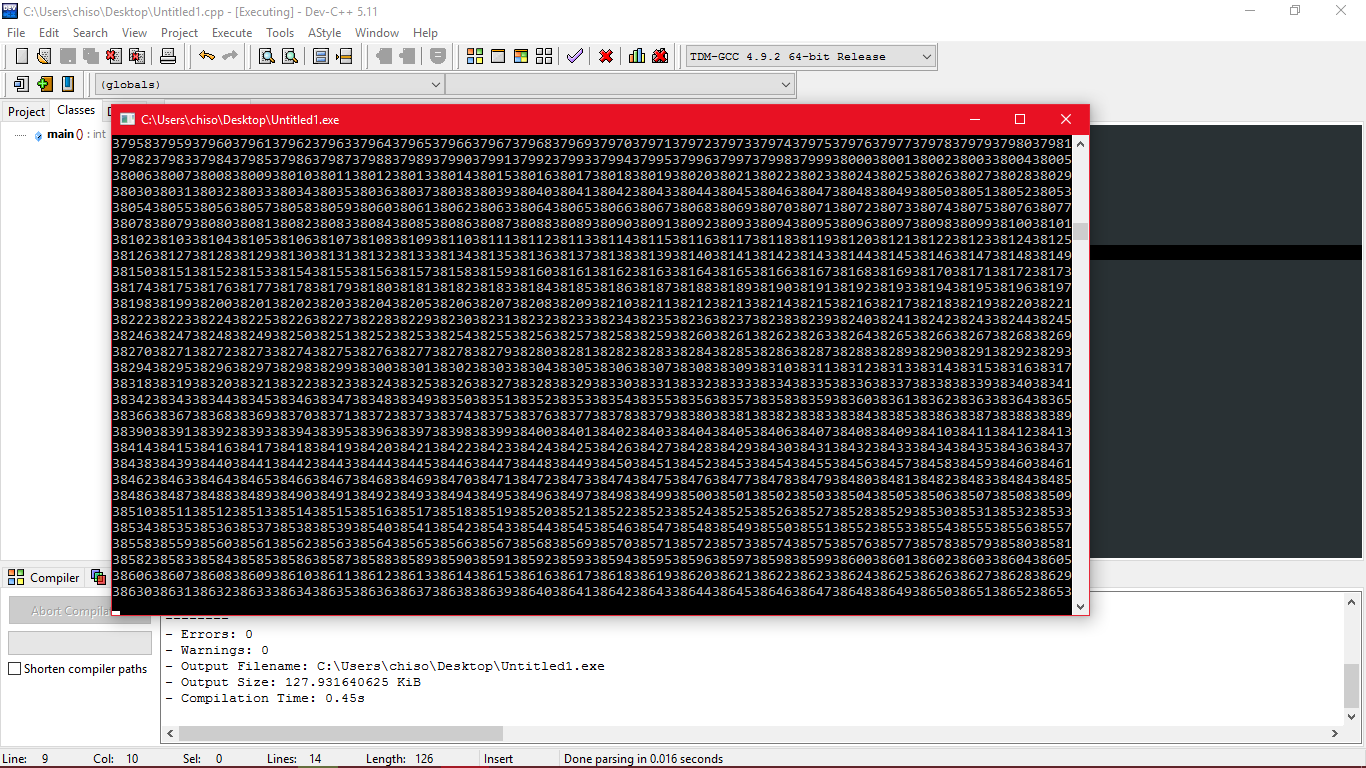
Sau for có dấu ; 🡪 vòng for lặp mà không thực hiện câu lệnh nào

Kết thúc vòng lặp, i=5. Lúc này chạy tới printf(“%d”,i); in ra màn hình số 5

**Câu 254. D.** Lỗi cú pháp vòng for

Các biểu thức trong for có thể khuyết, nhưng bắt buộc phải có 2 dấu ; ngăn cách giữa 3 phần

Nếu sửa thành: for(;i<=5;i++) kết quả in ra màn hình 0 1 2 3 4 5. Chương trình chạy với i=0 được gán ngay trên phần khai báo.

Hay thậm chí: for(;;i++) chương trình vẫn chạy, nó lặp vô hạn, do không có điều kiện dừng, và liên tục in ra màn hình các số tăng dần.

**Câu 255. A.** Vòng for nhanh hơn while

Chuyển for thành while:

|  |  |
| --- | --- |
| for(int i=0;i<=100;i++) printf(“%d”,i); | int i=0;  while (i<=100)  {  printf(“%d”,i);  i++;  } |
| for(bt1;bt2;bt3)  {//khối lệnh} | bt1;  while (bt2)  {  //khối lệnh  bt3;  } |

Trong đó:

bt1: biểu thức ban đầu, khởi tạo giá trị ban đầu của biến đếm

bt2: biểu thức điều kiện để thực hiện vòng lặp

bt3: biểu thức cập nhật giá trị biến đếm

//khối lệnh: các lệnh cần lặp

Các câu lệnh của for có thể không được lặp. VD: for(i=5;i<3;i++) Sau khi được gán giá trị là 5, i “bị” kiểm tra điều kiện i<3, vì không thỏa mãn nên chương trình không chạy vòng lặp.

**Câu 256. D**

**Câu 257. D.** 4,3,2,1,0,-1,

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| while: i>=0 | 5 | 4 | 3 | 2 | 1 | 0 | -1 |
| i=i-1 | 4 | 3 | 2 | 1 | 0 | -1 | -2 |
|  |  |  |  |  |  |  | (thoát) |
| printf() | 4 | 3 | 2 | 1 | 0 | -1 |

while(i-->=0) xuất hiện dạng hậu tố i--, ta lấy giá trị cũ của i để kiểm tra điều kiện i>=0, rồi sau đó trừ i đi 1. Điều kiện đúng thì thực hiện printf(), không thì thoát khỏi vòng lặp.

**Câu 258. C.** Cả A và B

**Câu 259. C.** 0 2 4 6 8

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| while: i<10 | 0 | 2 | 4 | 6 | 8 | 10(thoát) |
| printf() | 0 | 2 | 4 | 6 | 8 |
| i=i+2 | 2 | 4 | 6 | 8 | 10 |

Có thể chuyển thành vòng for: for(i=0;i<10;i+=2) printf(“%d “,i);

Dễ dàng nhận thấy trong vòng lặp, i nhận các giá trị từ 0, 2, 4, 8. Lệnh printf() sẽ in các giá trị này ra màn hình.

**Câu 260. C.** 2 4 6 8 10

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| while: i<10 | 0 | 2 | 4 | 6 | 8 | 10 (thoát) |
| i+=2 | 2 | 4 | 6 | 8 | 10 |
| printf() | 2 | 4 | 6 | 8 | 10 |

**Câu 261. B.** Chương trình lặp vô hạn và không dừng

Điều kiện lặp n<10 luôn đúng, n không hề bị thay đổi, vòng lặp while lặp vô hạn.

**Câu 262. D.** 1234

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| i=i+1 | 1 | 2 | 3 | 4 | 5 |
| i+1<=5 | true | true | true | true | false (thoát) |
| printf() | 1 | 2 | 3 | 4 |

while(++i +1<=5) xuất hiện biểu thức tiền tố ++i. Ở đây, i được tăng lên 1 trước i=i+1, rồi sau đó mới kiểm tra điều kiện i+1<=5 (i≤4)

**Câu 263. B.** 1,2,3,4,5,

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| i+=1 | 1 | 2 | 3 | 4 | 5 | 6 |
| i<=5 | true | true | true | true | true | false (thoát) |
| printf | 1, | 2, | 3, | 4, | 5, |

while(++i<=5) xuất hiện biểu thức tiền tố ++i. Ở đây, i được tăng lên 1 trước i=i+1, rồi sau đó mới kiểm tra điều kiện i<=5

Câu 264. C. 20

S=0, d=0

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| d<10 | true | true | true | true | true | false (thoát) |
| S+=d | 0 | 2 | 6 | 12 | 20 |
| d+=2 | 2 | 4 | 6 | 8 | 10 |

**Câu 265. C.** while(n<0||n>100)

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | A||B | A&&B |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 |

Ta cần nhập lại n khi n không thỏa mã điều kiện 0≤n≤100, tức là n<0 hoặc n>100: n<0||n>100  
 || là phép hoặc logic, && là phép và logic (A, B là các biểu thức, 0 là false, 1 là true):

Hai phép toán || và && tương tự như phép toán tuyển ˅ và phép hội ˄ trong logic mệnh đề.

Cần phân biệt với phép | (hoặc trên từng bit tương ứng) và & (và trên từng bit tương ứng).

VD:

int a=9, b=4;

printf("%d\n",a||b);

printf("%d\n",a|b);

Kết quả:

1

13

a||b : a=9 ≠ 0, được hiểu là true. b=4≠0, nên cũng được hiểu là true. True||true, được kết quả là true🡪1

a|b:

a= 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1

b= 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0

a|b= 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 =13

Câu 266. C. while(n<0||n>100)

(Tương tự câu 265)

Câu 267. A. A: n<0 B: n<0

Khi n<0 thì cần nhập lại🡪 cần xuất thông báo ra màn hình và lặp lại vòng lặp.

Câu 268. A. A=0

Vòng lặp nào cũng gặp lệnh continue; (buộc vòng lặp tiếp theo diễn ra, bỏ qua bất kỳ đoạn code nào ở giữa). Lệnh A=A+1 không được thực hiện, nên giá trị của A không đổi.

for( i=0 ; i<1000 ; i+=13 ) {

continue;

A=A+1;

}

Ban đầu, i được gán bằng 0. Sau đó, thực hiện kiểm tra i<1000, nếu đúng thì chạy lệnh continue trong tròng for, nếu sai thì thoát khỏi vòng lặp. Khi gặp lệnh continue, chương trình chuyển đến vòng tiếp theo: tăng i lên 13, rồi lại kiểm tra điều kiện i<1000....

Câu 269. B. 100

Ban đầu, i được gán bằng 0, sau đó kiểm tra điều kiện i<100 (0<100) đúng, nhảy vào vòng lặp, thực hiện lệnh A=A+100 ( A=100). Tiếp theo là gặp lệnh break. Chương trình thoát khỏi vòng lặp vô điều kiện.

for( i=0 ; i<100 ; i+=13){

A=A+100;

break;

}

Câu 270. A. Chương trình lặp vô hạn

while (i<’z’)

{

if (i==’a’) continue;

printf(“%c”,i);

}

Chương trình chạy đến lệnh while, thực hiện so sánh i<’z’ (‘a’<’z’). Khi so sánh hai ký tự, thực chất là so sánh mã ASCII của chúng với nhau. Trong bảng mã ASCII, ‘a’ có mã là 97, tiếp theo là ‘b’ với mã 98,....’z’ có mã ASCII là 122. Như vậy biểu thức i<’z’ đúng, chương trình nhảy và vòng lặp.

Tiếp theo thực hiện lệnh if với so sánh i==’a’, biểu thức này đúng, nên rẽ nhánh đến lệnh continue.

Khi gặp lệnh continue, chương trình nhảy vô điều kiện đến vòng lặp tiếp theo. Thực hiện so sánh i<’z’ (đúng), rồi lại thực hiện lệnh if....

Cứ như vậy, chương tình lặp vô hạn, không có lối thoát.

Câu 271. D. Không có kết quả nào được đưa ra màn hình

while (i<10){

i=i+2;

continue;

printf(“%d”,i);

}

Chương trình chạy đến lệnh while, thực hiện so sánh i<10 (i=0), điều kiện này đúng. Nhảy vào trong vòng lặp, thực hiện lệnh i=i+2 (i=2). Tiếp đến là thực hiện lệnh continue, chương trình nhảy vô điều kiện đến vòng lặp tiếp theo, bỏ qua lệnh printf ở dưới.

Thực hiện so sánh i<10 (i=2) đúng. Tiếp theo, i=i+2 (i=4)....  
 Cứ như vậy, đến khi i=10, không thỏa mã điều kiện lặp i<10 nữa, chương trình thoát khỏi vòng lặp.

**Câu 272. A.** 1/23/4/

int i=1;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| printf(“%d”,i) | 1 | 2 | 3 | 4 |
| i++ | 2 | 3 | 4 | 5 |
| i%3==0 | false | true | false | false |
| printf(“/”) | / |  | / | / |
| i%5==0 | false |  | false | true |

(thoát)

while(1) : biểu thức điều kiện là 1 (true) nên luôn thực hiện vòng lặp.

i%3==0 : i chia cho 3 có số dư bằng 0, hay i chia hết cho 3. Thì thực hiện lệnh continue: chạy vòng lặp while kế tiếp mà bỏ qua lệnh printf(“/”), if(i%5==0) ở dưới.

i%5==0: khi i chia hết cho 5 thì thực hiện lệnh break, thoát khỏi vòng lặp while.

**Câu 273. A.** 1 4 16 25

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| for: a++ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11(thoát) |
| a%2==0 | false | true | false | true | false | true | false | true | false | true |  |
| b+=a | 1 |  | 4 |  | 9 |  | 16 |  | 25 |  |  |
| printf() | 1 |  | 4 |  | 9 |  | 16 |  | 25 |  |  |

for( a=1 ; a<10 ; a++){

if (a%2==0) continue;

b=b+a;

printf(“%5d”,b);

}

Khi a chia hết cho 2, a%2==0 đúng, rẽ nhánh đến lệnh continue, đến vòng for tiếp theo, tăng a++...  
 Ngược lại, nếu a không chia hết cho 2, a%2==0 sai, thực hiện lệnh b=b+a và lệnh printf().

%5d : in số nguyên, và chừa ra 5 vị trí để in. Mục đích là chừa ra các khoảng trống giữa các số, nhằm dãn cách.

**Câu 274. B.**2 5 10 17 26

Tương tự câu 273.

b+=a; là dạng viết tắt của b=b+a;

“%d□“ (□ là dấu cách): in ra số nguyên cùng với 1 dấu cách ở ngay sau số đó. Mục đích là để dãn cách. Nếu chỉ dùng “%d” thì sẽ in ra: 25101726

|  |  |  |
| --- | --- | --- |
| i<10 | true | true |
| i++ | 1 | 2 |
| i%2==0 | false | true (thoát) |
| printf() | 1 |

Câu 275. B. 1

Khi chạy while, thực hiện kiểm tra điều kiện i<10 (i=0) 🡪 đúng 🡪 thực hiện vòng lặp. Trong vòng lặp, thực hiện i++: tương đương với i=i+1, tăng i lên 1 đơn vị 🡪 i=1. Tiếp theo là lệnh if, i%2=1 nên không thực hiện break. Tiếp đến là lệnh printf(), in số 1 ra màn hình. Sau đó, tiếp tục vòng lặp tiếp theo.

Kiểm tra điều kiện i<10 (i=1)🡪 đúng🡪 nhảy vào trong vòng lặp. Trong vòng lặp, thực hiện i++, lúc này i=2. Điều kiện i%2== đúng, nên thực hiện lệnh break🡪 thoát vòng lặp.

Câu 276. D. 81 36 9 0

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| n-- | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| n%3!=0 | false | true | true | false | true | true | false | true | true | false |
| printf() | 81 |  |  | 36 |  |  | 9 |  |  | 0 |
| n<3 | false |  |  |  |  |  | false |  |  | true(thoát) |

while(1) : 1≠0 nên được hiểu là true. Điều kiện luôn đúng nên luôn lặp.

n-- : chính là n=n-1, giảm n đi 1.

n%3 != 0 : n chia cho 3 có số dư khác 0, hay n không chia hết cho 3. Khi điều kiện này đúng, thực hiện lệnh continue, nhảy đến vòng lặp tiếp theo.

Câu 277. B. 1 3 5 7 9

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i<10 | true | true | true | true | true | true | true | true | true | true | false |
| i++ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| i%2==0 | false | true | false | true | false | true | false | true | false | true |
| printf() | 1 |  | 3 |  | 5 |  | 7 |  | 9 |  |

Khi i%2==0 (i chia hết cho 2) thì chạy lệnh contine: nhảy đến vòng lặp tiếp theo, bỏ qua lệnh printf().

Câu 278.