1 C 语言数据和运算基础部分参考题

1.1 填空题

1. 假设用 1 个字节存放一个整数，那么 21 的原码、反码和补码分别是（$(00010101)_2$）、（$(11101010)_2$）和（$(00010101)_2$），-21 的补码为（$(11101011)_2$）。

2. 已知代码段`float f1 = 0.709394709394; float f2 = 0.709394709395; printf("f1 == f2:  %d", f1 == f2);`输出 1（真），原因是（*float 的表示精度不够*）。

1.2 选择题（每题恰好有一个选项符合要求）

1. 以下说法不正确的是（*D*）

   A C 语言的数据类型所占的字节越多，能表示的数的范围越大

   B 字符型文字（Literal）用单引号夹住

   C `char a=52,b=73;char c=a+b;printf("%d",c);`输出 125

   D Visual C++ 6.0 开发环境支持`long long`型

2. 以下类型转换，不安全的是（*D*）。

   A `float a = (float)10;`

   B `int a = 5; long b = (long)a;`

   C `float a = 12.5; double b = (double)a;`

   D `int a = 1000; char c = (char)a;`

3. 以下说法不恰当的是（*B*）。

   A 如果`char`的表示范围是 $-2^7 \sim 2^7-1$，那么`unsigned char`的表达范围是 $0 \sim 2^8-1$。

   B `long`型的表达范围超过`int`型。

   C `short`型的表达范围不超过`int`型。

   D 如果`int`的表示范围是 $-2^{31} \sim 2^{31}-1$，那么`unsigned int`的表达范围是 $0 \sim 2^{32}-1$。

4. 已知 INT_MAX 是在 limits.h 头文件中定义的int型变量可表示的最大值，long型变量所占字节数大于int型变量所占字节数。设 t 为int型，并且 t 的值为 47。以下代码不发生溢出的是 (*D*) 。

   A `int a = INT_MAX - t + 50;`

   B `long b = INT_MAX - t + 50;`

   C `long long b = INT_MAX - t + 50;`

   D `int d = INT_MAX - t + (long long)50 - INT_MAX;`

5. 若`sizeof(long long) = 8`，那么，`long long`型的表达范围是 (*C*) 。

   A $-2^8 \sim 2^8 - 1$

   B $-2^7 \sim 2^7 - 1$

   C $-2^{63} \sim 2^{63} - 1$

   D $-2^{64} \sim 2^{64} - 1$

6. 已知`sizeof(int) = 4`，那么，恰有 () 个int型数字可以安全地转换为`unsigned int`型；同时又恰有 () 个`unsigned int`型数字可以安全地转换为int型。正确选项为 (*B*) 。

   A 2^{32}，2^{32}

   B 2^{31}，2^{31}

   C 2^{31}，2^{32}

   D 2^{32}，2^{31}

7. 以下说法恰当的是 (*C*) 。

   A limits.h 包含char型和float型的最大值

   B 如果int型变量能表示某个整数 n，那它也一定能表示 $-n$

   C 凡是float型变量能表示的数，double型变量都能表示

   D `unsigned int`能表示的整数的个数，比int能表示的要少

8. 以下哪一个宏在 limits.h 中无定义 (*A*) 。

   A UINT_MIN

B `CHAR_MAX`

C `INT_MIN`

D `ULLONG_MAX`

9. 在 `float.h` 中，反映数据类型精度的是 (*D*) 。

A `FLT_MAX`

B `DBL_MIN`

C `LDBL_MAX`

D `DBL_EPSILON`

10. 以下说法不恰当的是 (*A*) 。

A `printf("%d", 'A' - 'a');` 输出 `32`。

B `printf("%d", '5' - '4');` 输出 `1`。

C `printf("%d", 'z' - 'a');` 输出 `25`。

D `printf("%c", 'a' + 1);` 输出`'b'`。

11. 在 C 语言的语法中，以下变量命名，合法的是 (*B*) 。

A `3age`

B `book_cost`

C `maths-book`

D `student id`

12. 以下关于变量名 `DataBaseUser`, `myFirstName`, `student_id` 的说法，不恰当的是 (*D*) 。

A 前两个使用驼峰命名法，后一个使用下划线命名法

B 第一个使用大驼峰法，第二个使用小驼峰法

C 小驼峰法常用于命名变量

D 大驼峰法常用于命名函数

13. 以下运算符用于获取地址的是 (*A*) 。

A &

B *

C %

D /

14. 假设int型变量占据 4 个字节的存储空间，并且int型变量 a 的地址值为 0028FE10，那么，以下 4 个地址对应的内存单元，不被 a 占据的是 (*D*) 。

A 0028FE11

B 0028FE12

C 0028FE13

D 0028FE14

15. 假设int型和float型变量占 4 个字节，并且char型变量占 1 个字节。已知int a; ··· float s; ··· char ch;，并且 ch 的地址为 0028FF17，那么 s 和 a 的地址可能是 (*D*) 。

A 0028FF17, 0028FF1B

B 0028FF18, 0028FF1A

C 0028FF18, 0028FF1B

D 0028FF18, 0028FF1C

16. 已知int n; 则关于 n = 1234; 的执行结果的说法，恰当的是 (*D*) 。

A n 的地址可能发生改变

B n 在内存中的位置可能发生改变

C n 占据空间的大小可能发生改变

D n 所占据内存的 0-1 组合可能发生改变

17. 关于代码段int n; scanf("%d", &n); 的执行结果的说法，恰当的是 (*B*) 。

A n 的地址可能发生改变

B n 所占据内存的 0-1 组合可能发生改变

C n 占据空间的大小可能发生改变

D n 的数据类型可能发生改变

18. 关于`short n = 1234; short m = n;`的说法，恰当的是（*D*）。

A m 和 n 的地址都可能发生改变

B m 的地址可能发生改变，n 的地址不可能发生改变

C m 的地址不可能发生改变，n 的地址可能发生改变

D m 和 n 的地址都不可能发生改变

19. 代码段`int a = 2.3; float b = 5; int c = -3.6; printf("a:%d, b:%.0f, c:%d", a, b, c);`输出的结果是（*A*）。

A 2，5，-3

B 2.3，5.0，-3.6

C 2.3，5，-4

D 2.3，5，-3.6

20. 已知`sizeof(char) = 1`。以下四个选项的值中，在不发生赋值溢出的前提下，`char ch;`中的 ch 可以被赋予的最大值为（*D*）。

A 130

B 129

C 128

D 127

21. 以下四个代码段，赋值不安全的是（*D*）。

A `int a = 20; char ch = a;`

B `float b = 5.0; int c = b;`

C `long long d = 1000; int f = d;`

D `long g = 1000; char h = g;`

22. 按照 C99 规定，表达式 `10 % 3` 和 `(-10) % (-3)` 分别得到（*B*）。

A 1，1

B 1，-1

C 1，0

D 3，1

23. 表达式 `10 / 6` 和 `(-9) / 4` 分别得到（*A*）。

A 1，-2

B 2，-3

C 2，-2

D 1，-3

24. 按照 CPU 运算时的“字节对齐”原则，以下说法不恰当的是（*B*）。

A `char`型数据总是先转为`int`型数据，才参加运算。

B `char`型数据总是先转为`short`型数据，才参加运算。

C `short`型数据总是先转为`int`型数据，才参加运算。

D `float`型数据总是先转为`double`型数据，才参加运算。

25. 以下数据类型转化，不安全的是（*D*）。

A `char`型数值转`int`型数值

B `int`型数值转`long`型数值

C `int`型数值转`double`型数值

D `double`型数值转`int`型数值

26. 考虑如下代码段。

I. `float s = 0.0; s += (-5) / 3; printf("%.4f", s);`

II. `float s = 0.0; s += (-5) / 3.0; printf("%.4f", s);`

下列说法正确的是（*A*）。

A 代码段（I）输出`-1.0000`，代码段（II）输出`-1.6666`

B 代码段（I）和（II）都输出`-1.6666`

C 代码段（I）和（II）都输出`-1.0000`

D 代码段（I）输出-1.6666，代码段（II）输出-1.0000

27. 考虑如下代码段。

I. `int n = 3; printf("%d", n++);`

II. `int n = 3; printf("%d", ++n);`

下列说法正确的是（*A*）。

A 代码段（I）输出 3，代码段（II）输出 4

B 代码段（I）和（II）都输出 3

C 代码段（I）和（II）都输出 4

D 代码段（I）输出 4，代码段（II）输出 3

28. 在C语言的头文件 math.h 中，以下关于求绝对值、求平方根的函数和舍入函数的说法，不恰当的是（*C*）。

A 求float型的值的绝对值用 fabs()

B 求float型的值的平方根用 sqrtf()

C 求double型的值的平方根用 sqrtd()

D 舍入函数的传入和传出都是double型

29. 如果要交换int型变量 m 和 n 的值，以下代码恰当的是（*A*）。

A `int temp = m; m = n; n = temp;`

B `int temp = m; n = m; m = temp;`

C `int temp = n; m = n; n = temp;`

D `m = n; int temp = m; n = temp;`

30. 一个咨询师的计费标准如下：咨询时长最低按 4 个小时收费；如果超出 4 个小时，则按实际计算（小时数为整数）。已知前面的代码声明了`int hours; scanf("%d", &hours);`，那么以下代码恰当反映上述意思的是（*B*）。

A `hours = hours <= 4 ? hours : 4;`

B `hours = hours > 4 ? hours : 4;`

C `hours = hours > 4 ? 4 : hours;`

D `hours = hours > 4 ? hours : (hours - 4);`

31. 以下关于C语言关键字`const`和`define`的说法，不恰当的是（*C*）。

A `const`定义的常量，在编译中将进行类型检查

B `define`定义的常量不存在地址

C `const`定义的常量不存在地址

D 语句`int const PEOPLE_NUM = 112;`和`const int PEOPLE_NUM = 112;`的含义相同

32. 以下说法恰当的是（*D*）。

A 在头文件 `stdlib.h` 中，`srand()` 用于生成下一个伪随机数

B 在头文件 `stdlib.h` 中，`rand()` 用于设置随机数种子

C 在头文件 `stdlib.h` 中，`rand()` 获取的数可以是负数

D 在头文件 `time.h` 中，`time(NULL)` 返回从 UTC 时间 1979 年 1 月 1 日 0 时 0 分 0 秒到当前时刻经过的秒数

33. 在娱乐游戏和科学实验中，都可能用到伪随机数。以下说法最恰当的是（*A*）。

A 在娱乐游戏中，用 `time()` 的返回值作为种子；在科学实验中，人工指定种子

B 在娱乐游戏中，人工指定种子；用 `time()` 的返回值作为种子

C 两个场景都用 `time()` 的返回值作为种子

D 两个场景都人工指定种子

34. 以下表达式可以作为逻辑上的“真”的是（*C*）。

A `0`

B `'\0'`

C `'0'`

D `0.0`