#1，2，3

Sorry but I haven’t written SQL for a long long time… I forgot lots of things about SQL without the Internet…

# 4

Q: What is the purpose of ‘delete’ operator?

A: The purpose of the delete operation is to run the destructor of the object .

Q: Which members of the base class are visible in a derived class?

A: public/protected

# 5:

size\_t decimalLength(uint32\_t value) {

if (value == 0) {

return 1;

}

size\_t length = 0;

while (value > 0) {

value /= 10;

length++;

}

return length;

}

int getTopDecimal(uint32\_t value) {

while (value >= 10) {

value /= 10;

}

return value;

}

uint32\_t removeTopDecimal(uint32\_t value) {

if (value < 10) {

return 0; // If the value is a single digit, return 0

}

uint32\_t powerOf10 = 1;

while (value >= 10) {

value /= 10;

powerOf10 \*= 10;

}

return value + (value \* powerOf10); // Return the value without its top decimal

}

uint8\_t mergeThem(const uint8\_t \_a, const uint8\_t \_b) {

if (\_a > 0xF || \_b > 0xF) {

return 0;

}

return (\_a << 4) | \_b;

}

unsigned int convert\_int\_to\_bcd(unsigned char\* destination\_buffer, unsigned int bufferlength, unsigned int value) {

auto outputLength = decimalLength(value);

bool isOdd = false;

if (outputLength % 2 != 0) {

isOdd = true;

}

size\_t \_index = 0;

if (isOdd) {

destination\_buffer[\_index++] = getTopDecimal(value);

value = removeTopDecimal(value);

}

for (auto i = 0; i < outputLength; i++) {

auto \_a = getTopDecimal(value);

value = removeTopDecimal(value);

auto \_b = getTopDecimal(value);

value = removeTopDecimal(value);

destination\_buffer[\_index++] = (\_a << 4) | \_b;

}

return \_index;

}

# 6:

void sortIntVector(std::vector<int>& theVetor) {

for (auto i = 0; i < theVetor.size() - 1; i++) {

for (auto j = i + 1; j < theVetor.size(); j++) {

if (theVetor[i] < theVetor[j]) {

// swap them

auto temp = theVetor[i];

theVetor[i] = theVetor[j];

theVetor[j] = temp;

}

}

}

}

#7

size\_t ListFiles(const TCHAR\* \_inputDir) {

WIN32\_FIND\_DATA \_findFileData;

HANDLE \_handle = FindFirstFile(\_inputDir, &\_findFileData);

size\_t \_count = 0;

if (\_handle == INVALID\_HANDLE\_VALUE) {

return 0;

}

for (; FindNextFile(\_handle, &\_findFileData) != 0;) {

if (\_findFileData.dwFileAttributes & FILE\_ATTRIBUTE\_DIRECTORY) {

if (\_tcscmp(\_findFileData.cFileName, \_T(".")) != 0 && \_tcscmp(\_findFileData.cFileName, \_T("..")) != 0) {

TCHAR \_subDir[MAX\_PATH];

\_stprintf\_s(\_subDir, \_T("%s\\%s"), \_inputDir, \_findFileData.cFileName);

\_tprintf(\_T("%s\n"), \_subDir);

\_count++;

\_count += ListFiles(\_subDir);

}

} else {

\_tprintf(\_T("%s\n"), \_findFileData.cFileName);

\_count++;

}

}

FindClose(\_handle);

return \_count;

}