2020 年 C++ 程式设计語言等級考試

試題冊

考生注意

1．題目中所有來自 C++ 標準樣板程式庫（C++ Standard Template Library）的名稱都假設已經引入了對應的標頭檔；

2．本次考試不考察執行緒和網路程式設計相關內容，考生答題時可以不考慮因執行緒或網路程式設計導致的問題；

3．本次考試的涉及範圍內容止步於 C++ 11，考生不必考慮 C++ 11 之後的任何用法；

4．考生需要將答案寫在答題紙上，所有寫在試題冊及草稿紙上的答案無效；

5．本場考試為閉卷筆試，考試時間為 4 個小時；

6．考試開始前考生禁止以任何方式打開試題冊．

(一) 選擇題. 1 ~ 8 題, 每小題 2 分, 共計 16 分. 每個選擇題給出 (**A**), (**B**), (**C**), (**D**), (**E**), (**F**) 和 (**G**) 七個選項, 每一題至少有一個正確選項, 至多有七個正確選項. 多選、少選或錯選均不得分.

1. 本試卷考察的內容是 (　　)
(**A**) C++　　(**B**) F#　　(**C**) C#　　(**D**) Objective-C
(**E**) Fortran　　(**F**) 機器語言　　(**G**) PHP

2. 在沒有運算子多載的情況下, 下列運算子存在多個含義的有 (　　)
(**A**) &　　(**B**) <　　(**C**) new　　(**D**) ~
(**E**) sizeof...　　(**F**) .　　(**G**) typeid

3. 下列選項中, 不支援巢狀的有 (　　)
(**A**) class / struct　　(**B**) enum　　(**C**) 函式　　(**D**) /* */
(**E**) 不具名名稱空間　　(**F**) 指示連結　　(**G**) union

4. 設變數 i 的型別為 int 且被 constexpr 標識, 下列宣告中正確的有 (　　)
(**A**) decltype(((i))) i2;　　(**B**) decltype(i) &&i3 {4};
(**C**) int i4(i);　　(**D**) void *p = &i;　　(**E**) static int &i5 {i};
(**F**) const char &&c = i;　　(**G**) enum {i, j, k};

5. 下列選項中, 由 C++ 11 引入的關鍵字有 (　　)
(**A**) char32_t　　(**B**) register　　(**C**) export　　(**D**) explicit
(**E**) auto　　(**F**) bitor　　(**G**) char8_t

6. 下列選項中, 可能或一定無法通過編碼器編碼的有 (　　)
(**A**)
```
auto empty_pointer = reinterpret_cast<int *>(nullptr);
```
(**B**)
```
class Foo {
    public:
        Foo();
    private:
        void f() = delete;
    };
```
(**C**)
```
auto p = new auto {0};
```
(**D**)
```
void f() noexcept {
        throw;
    }
```
(**E**)
```
auto arr[] = {1, 2, 3};
```
(**F**)
```
template <typename T>
    T f() {
        return {};
    }
```
(**G**)
```
constexpr int a = {};
    int main(int argc, char *argv[]) {
    #ifdef a
        ++argc;
    #endif
    }
```

7. 下列選項中, 可能會發生未定行為的是 (　　)

(**A**)
```
std::vector<int> vec(0, 1, 2, 3, 4, 5, 7);
```

(**B**)
```
std::vector<int> vec(63, 42, std::allocator<int> {});
for(auto c : vec) {
    vec.insert(vec.crbegin() + c, 42);
}
```

(**C**)
```
template <typename T>
typename std::add_pointer<T>::type alloc() {
    using rtype = typename std::add_pointer<T>::type;
    return static_cast<rtype>(::operator new(sizeof T {}));
}
```

(**D**)
```
delete this;
```

(**E**)
```
template <typename T, typename U>
bool compare(const T &t, const U &) {
    return t < u;
}
```

(**F**)
```
int main() {
    class Base {
        ~Base() {}
    };
    class Derived : Base {
        ~Derived() {}
    };
    Derived d;;
}
```

(**G**) 設 p 和 q 都是 int * 型別的指標, 且 p == q 成立. 先執行 new (p) int(1); 再令 *q = 0;

8. 下列說法正確的是 (　　)

(**A**) std::vector 與 std::string 多載的陣列注標運算子不可以用於添加新的元素.

(**B**) std::map 與 std::set 多載的陣列注標運算子可以用於添加新的元素.

(**C**) 函式可以回傳陣列型別.

(**D**) 設 arr 是 int [N] 型別的陣列, 其中 N 是一個整型的常數表達式. 對陣列 arr 使用 decltype 及 auto 進行推導, 得到的型別結果不相同.

(**E**) const_cast 運算子對頂層 const 無效.

(**F**) 可以使用 std::initializer_list 初始化 std::priority_queue.

(**G**) 不具名 union 存在成員丟失, 即不具名 union 宣告之後無法再使用其成員.

(二) 填空題. 9 ~ 14 題, 每小題 3 分, 共計 18 分. 對於多個答案的填空題, 錯填或者漏填不得分.

9. 設在某個函式之內有程式碼片段

```
int i = 0;
auto &i2 = i;
```

請使用 decltype 宣告變數 i3. 其中, i3 的型別為 int, 推斷自 i2, 且 i3 不可以被初始化 :

_____.

10. 設有

```
template <typename ...Ts>
struct meta_f {
   constexpr static auto value = true;
};
template <>
struct meta_f<void ()> {
    constexpr static auto value = false;
};
struct Foo {
    constexpr const void *volatile f(int &, char, ...) const
         volatile && noexcept(noexcept(meta_f<typename
              std::add_lvalue_reference<void>::type>::value)) {
       return nullptr;
    }
};
```

宣告成員函式指標 p, 指向 Foo::f. 不可以借助 auto, decltype 以及 template :

_____.

11. 寫出哪一些情況下函式會被宣告為隱含的 inline 函式 :

_____.

12. 在參數型別不可以使用任何推導和回傳型別不可以使用 template 推導的情況下, 寫出函式 f 的宣告. 函式 f 僅接受一個陣列的參考, 該陣列可以是任意型別且陣列中元素的數量不固定. 另外, 函式 f 的回傳型別從陣列的首個元素推導 :

_____.

13. 設有程式碼 :

```
#include <iostream>
int main(int argc, char *argv[]) {
    std::string str = "123";
    std::cout << str << std::endl;
    std::cin >> str;
    std::cout << str << std::endl;
}
```

若編碼器只能看到 std::string 中的成員函式和成員變數, 那麼上面的程式碼會產生編碼錯誤. 我們可以通過某個提前宣告來避免這個編碼錯誤, 這個宣告是 :

_____.

14. 設有函式宣告 :

```
    void f(unsigned short *);
    void f(unsigned);
    void f(long long);
    void f(...);
```

函式呼叫 f(NULL) 可能選中的函式有 (若存在多個, 那麼一行僅寫一個, 一行寫多個不給分) :

_____

_____

_____

_____.

(三) 論述題. 15 ~ 20 題, 每小題 2 分, 共計 12 分.

15. 簡述 C++ 標準樣板程式庫中的 I/O 物件不可被複製的原因.

16. 簡述移動建構子或移動指派運算子應儘量標識 noexcept 的原因.

17. 簡述物件被移動之後就不再建議使用的原因.

18. 簡述 std::list 與 std::forward_list 有專屬排序函式的原因.

19. 簡述 C++ 標準樣板程式庫中的演算法通常要求傳入疊代器而非容器的好處.

20. 在不使用 auto, decltype 和 template 對下列函式的型別進行任何推導的情況下, 寫出可以產生下列函式的可呼叫物件的五種不同方法. 其中, 函式原型為

```
const Foo ::operator==(Bar &, Baz *) noexcept;
```

(四) 改錯題. 21 ~ 23 題, 每小題 5 分, 共計 15 分. 每個改錯題的錯誤包括但不限於編碼錯誤, 未定行為, 錯誤結果和不正確的實作, 指出可能存在錯誤的地方並且在不刪除任何程式碼的情況下更改程式碼使得其可以正確運作. 初始為 5 分, 漏改一處扣 1 分, 改錯一處扣 2 分, 得分不低於 0 分.

21.
```
#include <string>
struct Foo {
    int a(0);
    int override {1};
    static int c {3};
    bool operator==(Foo *) = default;
    Foo(const Foo &) = delete;
    Foo(Foo &&) noexcept = default;
    virtual ~Foo() = default;
};
struct Bar : Foo {
    std::string str;
    Bar(const string str = "0") : str(str) {}
};
int main(int argc, char *argv[]) {
    Bar b1("hello");
    Bar b2 = static_cast<Bar &&>(b1);
}
```

22.
```
void f(char &) noexcept;
int main(int argc, char *argv[]) {
    decltype(f) fp;
    fp = f;
    char c = 0;
    fp(c);
    auto *p = new int;
    const int *cp = nullptr;
    const_cast<int *>(cp) = p;
    struct Foo {
        int *p;
        struct Bar {
            auto i = 0;
            bool operator(char c)() {
                if(this->i == c) {
                    return false;
                }
                fp(c);
                return true;
            }
        };
    };
}
```

23. 在某一次競賽中，有一試題為：計算 $\frac{1}{1!}-\frac{1}{3!}+\frac{1}{5!}-\ldots+\frac{(-1)^{n+1}}{(2n-1)!}$。其中，$n$ 的值由用戶給定。某位同學給出的程式碼如下：

```
long long fac(int n, int r = 1) {
    return fac(--n, r * n);
}
long double cal(int n) {
    bool symbol = true;
    long double r = 0.0;
    for(int d = 1; d <= n; d += 2) {
        r += []() mutable {
            if(symbol) {
                return 1 / static_cast<long double>(fac(d));
            }
            return -1 / static_cast<long double>(fac(d));
        }
        symbol = -symbol;
    }
    return r;
}
```

顯然，上述程式碼存在錯誤。在不更改函式 fac 與 cal 型別，不刪除任何函式內的變數，不修改函式內已有變數的型別及保持 Lambda 表達式的情況下修改上述函式，使得函式呼叫 cal(n) 可以得到正確的結果。

(五) 程式設計題. 24 ~ 29 題, 共計 39 分. 除了有額外要求的情形, 要求全程自行設計, 不可使用類似於 C++ 標準樣板程式庫等中的任何名稱.

24. 特性創造題. ① ~ ③ 題, 每題 2 分. 下面程式碼中存在未被 C++ 11 引入的特性, 請閱讀這些程式碼, 寫出這些特性並且介紹特性的用處.

① 設有函式 Mat visual::threshold(Mat, int, int, int). 多數函式呼叫者在呼叫函式時, 呼叫的形式為 :

```
visual::threshold(matrix, default, default, 188);
```

② 設 q 為 int * 型別的指標, 使用者的程式碼 :

```
if(auto p = q; q) {
    //do something...
}
```

③ 設下列程式碼位於某個類別內, 類別內有成員變數 p 與 f, 它們都可以隱含地轉型為 bool 型別 :

```
auto lambda = [= this](auto a, auto b) noexcept -> auto {
    if(this->p) {
        return true;
    }
    return this->f;
};
```

25. 資料結構實作題. (6 分) 堆疊 (stack) 是一種滿足先進後出 (FILO) 的資料結構. 每一次向堆疊中插入一個元素都只能將這個元素放到堆疊的尾部 (頂部), 每一次從堆疊中移除元素都是直接從堆疊的尾部 (頂部) 移除一個元素. 現在要求使用 class 實作一個堆疊, 它至少存在插入元素 (push), 移除元素 (pop), 初始化 (給定一個大小 size, 使得堆疊中有 size 大小的預留空間. 當 size 未指定時, 預設大小為 64), 清空元素 (clear), 判定堆疊是否為空 (empty), 取堆疊尾部 (頂部) 的元素 (top). 對於出現的邊界情況, 考慮擲出字面值字串為提示的例外情況. 任何成員變數都不應該對外公開.

26. 物件導向設計題. (6 分) 生物 (creature) 是地球 (Earth) 上迄今為止最為神奇的物質, 也是大自然 (nature) 的饋贈 (gift). 萬物生之於大自然, 止之於大自然. 生物有生命 (lifetime), 呼吸 (breathe), 身長 (height) 和體重 (weight) 等生物特徵 (biometric). 生物不斷進化 (evolution), 演化出了靈長類 (primate). 靈長類擁有眼 (eyes), 鼻 (nose), 嘴 (mouse), 耳 (ears), 手(hand) 和腳 (foot). 除此之外, 靈長類還進化出了爬行 (crawl) 等能力 (ability). 靈長類和我們息息相關, 靈長類不斷進化便有了猴子 (monkey) 和其餘物種 (other creature). 猴子除了擁有靈長類的基本特徵之外, 還有了尾巴 (tail) 和體毛 (hair) 特徵和跳躍 (jump) 等能力. 猴子不斷進化, 就有了古猿 (acient ape), 它的尾巴退化 (degeneration) 了. 古猿的進化有分化, 一部分古猿進化成了猿 (ape), 還有一部分進化成了原始人 (primitive), 它們都擁有直立行走 (walk) 和跑 (run) 的能力. 猿和原始人共同進化, 就有了智慧 (wisdom) 人 (human). 人類擁有智慧之後, 就有了獨立思考 (think), 語言 (language) 表達 (speak) 的能力, 創造 (create) 了很多東西 (things), 開始給自己取名字 (name), 創造出了社會 (society)... 直到現代世界 (modern world). 請根據下列要求將上述描述轉化為程式碼 :

① 每一個動作可以使用一個輸出陳述式替換, 不可以出現僅僅宣告而不實作的函式. 輸出陳述式中, 可以使用的 C++ 標準樣板程式庫物件為 std::cin 和 std::cout.

② 生物的任何屬性不應對外公開.

③ 若描述中出現 "等" 這個字樣, 那麼需要實作出一些描述中未出現的特徵或者動作.

27. 遞迴設計題.（6 分）河內塔（Hanoi Tower）問題. 傳說越南河內某間寺院有三根銀棒，第一根銀棒上有 64 個金盤. 寺院裡的僧侶依照一個古老的預言，以下面列出的規則移動 64 個金盤，預言說當這些盤子移動完畢，世界就會滅亡 : 每一次只能移動一個金盤，要求金盤全程由小到大排列（即不能出現大的金盤放置在小的金盤上），所有金盤最終需要按順序放置在第三根銀棒上. 設計程式，輸出每一次移動的過程（例如從第一根銀棒移動到第三根銀棒），統計總共移動的次數. 有如下要求 :

① 第一根銀棒上的金盤數量由用戶輸入自訂.
② 可以使用 C++ 標準樣板程式庫的物件有 std::cin 和 std::cout.
③ 要求使用過程導向的程式設計方式.
④ 要求使用遞迴的方式實作.

28. 過程導向設計題.（6 分）① ~ ④ 題.

①（2 分）設有函式 :

```
template <typename ...Args>
void f(const Args &...args);
```

函式 f 能夠對引數包 args 中的所有引數進行排序並且輸出排序之後的效果，且型別包 Args 中的所有型別都可以向 int 型別轉型. 現要求實作函式 f. 可以使用的 C++ 標準樣板程式庫物件為 std::cout.

②（1 分）設函數（function）$f(x) = a\,x^n$，它的導數（derivative）表達式為 $f(x) = n \cdot a\,x^{n-1}$. 設計函式計算任意給出的 $f(x)$ 的導數，若給定 x，則需要回傳計算結果. 可以使用的 C++ 標準樣板程式庫物件為 std::pow.

③（1 分）設函數（function）$f(x) = a\,x^n$，它的原函數（primitive）表達式為 $F(x) = \frac{a}{n}x^{n+1}$，$f(x)$ 在區間 $[a, b]$ 上的定積分（definite integral）的表達式為 $\int_a^b f(x)dx = F(b) - F(a)$，沒有給出區間的積分（integral）稱為不定積分，它就是結果就是原函數加上一個任意常數（本題中省略這個常數）. 請設計函式計算給定函數 $f(x)$ 的不定積分. 若給出區間，則需要回傳定積分的計算結果. 可以使用的 C++ 標準樣板程式庫物件為 std::pow.

④（2 分）廣義表是 Lisp 程式設計語言中內建的資料結構. C++ 雖然沒有內建廣義表，但是可以通過自行實作的方式來實作廣義表. 廣義表中的元素既可以是單個的，又可以仍然是一個廣義表. 設使用 C++ 實作的廣義表（generalized_list）有如下操作 : 取頭部元素（front），取尾部元素（back），取第 n 個元素（operator[n]），移除第 n 個元素（erase(n)），判定第 n 個元素是否為廣義表（is_glist(n)），計算廣義表的元素數量（size），判定廣義表是否為空（empty）. 如果某個元素不是廣義表而是單個元素，它將仍然以廣義表的形式儲存，只不過這個廣義表可以向對應的型別進行隱含型別轉化. 設計一函式展開廣義表，其宣告為 :

```
std::vector<int> unfold(const generalized_list<int> &) noexcept;
```

例如某個廣義表為 (1, (1, 2, 3), (1, (2, 3), (3, (4, 5, ())))), 展開後回傳的 std::vector<int> 應該是 {1, 1, 2, 3, 1, 2, 3, 3, 4, 5}. 可以使用的 C++ 標準樣板程式庫中的物件有 std::vector 和 std::stack，不可以使用遞迴的方式實作.

29. 記憶體控制題.（6 分）現在有一聯合開發的項目，項目使用的程式庫是舊式 C++ 標準. 而這個項目要求使用 C++ 標準樣板程式庫中的智慧指標 std::unique_ptr. 在項目使用的程式庫中，某一個函式要求給定一個型別為 T ** 的引數. 但是 std::unique_ptr 本身並不提供一個成員函式來回傳一個指標的參考. 因此，某個同學想到了使用 reinterpret_cast 來實作 :

```
#include <memory>
void f(T **);
int main(int argc, char *argv[]) {
    std::unique_ptr<T> p = new T();
    f(reinterpret_cast<T **>(&p));
}
```

在某些情況下，這種方法確實可行. 但是這樣的程式碼容易產生未定行為甚至嚴重錯誤，程式碼的可攜性與相容性也得不到保障. 現在需要設計一個類別，對上述需求進行包裝，提供易用性，安全性和可攜性. 要求如下 :

① 可使用的 C++ 標準樣板程式庫物件 : std::unique_ptr, std::exchange 和 std::default_delete. 其中, std::exchange 可能的實作為

```
template<class T, class U = T>
T std::exchange(T &obj, U &&new_value) {
    T old_value {std::move(obj)};
    obj = std::forward<U>(new_value);
    return old_value;
}
```

② 不可以重新設計或者改變 std::unique_ptr, 只能設計一個輔助類別.

2020 年 C++ 程式设计語言等級考試

封頁

考生注意

考試開始前考生禁止以任何方式打開試題冊