2020 年全国 C++ 等级考试

试题册

考生注意

1. 题目中所有来自 C++ 程式库 (C++ Standard Template Library) 的名称都假设已经包含了对应的头文件

2. 本次考试不考察多线程和网络编程相关内容, 考生答题时可以不考虑因多线程或网络编程导致的问题

3. 所有写在试题册及草稿纸上的答案无效

4. 本场考试方式为闭卷笔试, 考试时间为 4 个小时

5. 考试开始前不可打开考生试题册

(一)、选择题 : 1 ~ 8 题, 每小题 2 分, 共计 16 分. 每个选择题给出 A、B、C、D、E、F 和 G 七个选项, 每一题保证至少有一个正确选项, 至多有七个正确选项. 多选、少选或错选均不得分.

1. 本试卷考察的内容是 (　　)

A、C++　　B、F#　　C、C#　　D、Objective-C　　E、Fortran
F、机器语言　　G、PHP

2. 在没有运算符重载的情况下, 下列运算子中有多个含义的是 (　　)

A、&　　B、<　　C、new　　D、~　　E、sizeof...
F、.　　G、typeid

3. 下列选项中, 不支持嵌套的是 (　　)

A、class / struct　　B、enum　　C、函数　　D、/*　*/
E、未命名的命名空间　　F、链接指示　　G、union

4. 设 i 是一个 int 类型且被 constexpr 标识的变量, 下列定义中正确的是 (　　)

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. 下列关键字从 C++ 11 才引入的是 (　　)

A、char32_t　　B、register　　C、export　　D、explicit　　E、auto
F、bitor　　G、char8_t

6. 下列选项中, 可能或一定无法通过编译器编译的是 (　　)

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. 下列选项中，可能会发生未定义行为的是（　）

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() {
     return static_cast<typename std::add_pointer<T>::type>(::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、设 p 和 q 是一个 int * 类型的指针，且 p == q，有 new (p) int(1); 再令 *q = 0;

8. 下列说法正确的是（　）

A、std::vector 与 std::string 重载的数组下标运算符不可以用于添加新的元素

B、std::map 与 std::set 重载的数组下标运算符可以用于添加新的元素

C、函数可以返回数组类型

D、设变量 arr 是 int [N] 类型的数组，其中 N 是一个整型常量表达式. 对数组 arr 使用 decltype 及 auto 进行推导，得到的类型结果不相同

E、const_cast 运算符对顶层 const 无效

F、可以使用 std::initializer_list 初始化 std::priority_queue

G、匿名 union 存在成员丢失（即匿名 union 定义之后无法再使用其成员）

(二)、填空题 : 9 ~ 14 题, 每小题 3 分, 共计 18 分. 每个填空题至少有一个正确答案, 对于多个答案的填空题, 全填对得 3 分, 漏填得 1 分, 存在错填不得分.

9. 设有

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

请使用 decltype 声明变量 i3, 其类型推断自 i2, 且 i3 不可以被初始化 :

_____.

10. 设有

```
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
            add_lvalue_reference<void>::type>::value)) {
       return nullptr;
   }
};
```

写出一指针 p 指向 Foo::f, 不可使用 auto、decltype 以及 template :

_____.

11. 写出隐含的 inline 函数的情况 :

_____.

12. 写出函数 f 的声明, 使得其接受仅接受参数为任意大小的 T 类型数组的参考, 返回类型从数组的第 0 个元素推断 (不可以使用 template 推导数组的引用和数组第 0 个元素的类型) :

_____.

13. 设有代码 :

```
#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;
}
```

若编译器对上述代码的名称不作特殊处理, 那么要使得上述程式码通过编译器的变异, 应该额外写出的语句是 :

_____.

14. 设有函数声明 :

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

函数调用 func(NULL) 选中的函数是 :

_____.

(三)、改错题 : 15 ~ 18 题, 每小题 5 分, 共计 15 分. 每个改错题的错误包括但不限于编码错误、未定行为、错误结果和不正确的实作, 指出可能存在错误的地方并且在不删除程式码的情况下更改程式码使得你认为其可以正确运作. 初始为 5 分, 漏改一处扣 1 分, 改错一处扣 2 分, 得分不低于 0 分

15.
```
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);
}
```

16.
```
template <typename T>
struct add_lvalue_reference {
      using type = T &;
};
```

17.
```
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;
              }
        };
  };
}
```

18. 在某一次竞赛中，试题卷中其中一试题为 ：计算 $\frac{1}{1!}-\frac{1}{3!}+\frac{1}{5!}-\ldots+\frac{(-1)^{n+1}}{(2n-1)!}$. 其中，n 的值由用户给定. 某位同学给出的程式码如下 ：

```
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;
}
```

上述代码存在错误，在不更改函数 fac 与 cal 类型、不删除任何函数内的变量、不修改函数内已有变量的类型及保持 lambda 表达式的情况下修改上述函数，使得其 cal(n) 可以得到正确的结果.

(四)、论述题 : 19 ~ 24 题, 每小题 2 分, 共计 12 分.

19. 简述 C++ 标准库中的 I/O 对象不可被复制的原因.

20. 简述移动构造函数或移动赋值运算符应尽量标识 noexcept 的原因.

21. 简述被移动的对象不建议使用的原因.

22. 简述 std::list 与 std::forward_list 有自己的排序函数的原因.

23. 简述 C++ 标准库中的算法通常要求传入迭代器而非容器的好处.

24. 使用 5 种不同的方法产生可调用对象, 其中函数原型为

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

(五)、程式设计 : 25 ~ 30 题, 共计 39 分. 本题除了小题有额外的要求之外, 要求全程自行设计, 不可使用类似于 C++ 标准样板程式库等中的任何名称.

25. 特性创造题 : ① ~ ③ 题, 每题 2 分. 下列都是未被 C++ 11 引入的特性, 请阅读这些代码, 写出这些特性并且介绍特性的用处.

①. 设有函数原型 : `Mat visual::threshold(Mat, int, int, int);`
多数使用者在使用函数时, 调用的形式为 :
```
visual::threshold(matrix, default, default, 188);
```

②. 设 q 为 int * 类型的指针, 使用者的代码 :
```
if(auto p {q}; q) {
    //do something...
}
```

③. 设下列代码位于某个类内, 类内有成员变量 p 与 f, 它们都可以隐含地转型为 bool 类型 :
```
auto lambda {[this](auto a, auto b) noexcept -> auto {
    if(this->p) {
        return true;
    }
    return this->f;
}};
```

26. 数据结构实现题 : 6 分. 栈 (stack) 是一种满足先进后出 (FILO) 的数据结构. 每一次入栈都是将元素放入栈的顶部, 每一次出栈都是将栈顶部的元素从栈中删除. 要求使用 class 实现一个栈, 它至少满足元素入栈 (push), 元素出栈 (pop), 栈的初始化 (给定一个 size, 使得栈的大小为 size, 当 size 未指定, 则默认大小为 64), 栈的清空 (clear), 判定栈是否为空 (empty), 取栈顶部的元素 (top). 对于出现的边界情况, 考虑抛出异常.

27. 面向对象设计题 : 6 分. 生物 (creature) 是地球 (Earth) 上迄今为止最为神奇的一类, 也是大自然 (nature) 的馈赠 (gift), 万物生之于大自然, 止之于大自然. 生物有生命 (lifetime)、呼吸 (breathe)、身长 (height) 和体重 (weight) 等生物特征 (biometric). 生物不断进化 (evolution), 演化出了灵长类 (primate). 灵长类拥有眼 (eyes)、鼻 (nose)、嘴 (mouse)、耳 (ears)、手(hand)、脚 (foot), 除此之外, 灵长类还进化出了爬行 (crawl) 等能力. 灵长类和我们息息相关, 灵长类不断进化有了猴 (monkey) 和其余的物种 (other creature). 猴子除了拥有灵长类的基本特征之外, 还有了尾巴 (tail) 和体毛 (hair) 等特征, 还拥有了跳跃 (jump) 等的能力 (ability). 猴子不断进化, 就有了古猿 (acient ape), 它的尾巴退化 (degeneration) 了. 古猿的进化有分化, 一部分古猿进化成了猿 (ape), 还有一部分进化成了原始人 (primitive), 它们都拥有直立行走 (walk) 和跑 (run) 等的能力. 猿和原始人共同进化, 就有了智慧 (wisdom) 人 (human). 人类拥有智慧之后, 就学会了自我独立思考 (think), 语言 (language) 表达 (speak) 等的能力, 创造 (create) 了很多东西 (things), 开始给自己取名字 (name), 创造出了社会 (society), ..., 直到现代世界 (modern world). 请根据下列要求, 设计类别 :

①可以使用的 C++ 标准库对象 : std::string, std::cin, std::cout, std::endl

②每一个动作可以使用一个输出语句替换, 不可以出现仅仅声明而不实现的函数

③生物属性不应向外公开

28. 递归设计题 : 6 分. 汉诺塔 (Hanoi Tower) 问题 : 传说越南河内某间寺院有三根银棒, 上串 N (N 由使用者输入) 个金盘. 寺院里的僧侣依照一个古老的预言, 以下面列出的规则移动 64 个盘子. 预言说当这些盘子移动完毕，世界就会灭亡. 设初始这些金盘按顺序被放置在第一根银棒 X 上, 设第二根银棒是 Y, 设第三根银棒为 Z. 每一次只能移动一个金盘, 要求金盘全程由小到大排列 (即不能出现大的金盘放置在小的金盘上), 所有金盘最终需要按顺序放置在银棒 Z 上. 设计程序, 输出每一次移动的过程 (例如从 X 移动到 Y), 统计总共移动的次数, 并且输出最终银棒 Z 上的金盘情况. 有如下要求 :

①可以使用 C++ 标准库的对象 : std::vector, std::string, std::deque, std::cin, std::cout, std::endl

②不可使用 std::vector 及 std::string 的下标运算符

③要求使用面向过程的程序设计方式

④要求使用递归的方式实现

29. 面向过程设计题 : 6 分, ① ~ ④ 题.

①(2 分) 设有函数 :

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

设包 Args 中的类型可以向 int 转型. 设计程序, 写出函数 f 对包 Args &&... 中的参数进行排序后, 输出这些参数的整型数据. 可以使用的 C++ 标准库对象 : std::cout, std::endl, std::forward

②(1 分) 设函数 (function) $f(x) = a\,x^n$, 它的导数 (derivative) 表达式为 $f(x) = n \cdot a\,x^{n-1}$. 设计函数计算任意给出的 $f(x)$ 的导数, 若给定 x, 则需要返回计算结果. 可以使用的 C++ 标准库对象 : std::pow

③(1 分) 设函数 (function) $f(x) = a\,x^n$, 它的原函数 (primitive) 表达式为 $F(x) = \frac{a}{n}x^{n+1}$, $f(x)$ 在区间 $[a, b]$ 上的定积分 (definite integral) 的表达式为 $\int_b^a f(x)dx = F(b) - F(a)$, 没有给出区间的积分 (integral) 称为不定积分, 它就是结果就是原函数加上一个任意常数 (本题中省略这个常数). 请设计函数计算任意给定的 $f(x)$ 函数的不定积分. 若给出区间, 则需要返回定积分的计算结果. 可以使用的 C++ 标准库对象 : std::pow

④(2 分) 广义表是 Lisp 中内置的数据结构, C++ 中广义表非内置, 但是可以通过自行实现的方式获得广义表. 广义表中的元素既可以是单个的, 又可以仍然是一个广义表. 设使用 C++ 实现的广义表 (generalized_list) 有如下操作 : 取头部元素 (top), 取尾部元素 (back), 取第 n 个元素 (operator[]), 移除第 n 个元素 (erase(n)), 判定某个元素是否为广义表 (is_glist(n)), 计算广义表中的元素数量 (size), 判定广义表是否为空 (empty), 如若某个元素不是广义表而是单个元素, 那么它仍然以广义表存储, 只不过他可以向对应的类型进行隐含转化, 单个元素也可以向广义表进行隐含转化. 设计一函数展开广义表 :

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

例如某个广义表为 (1, (1, 2, 3), (1, (2, 3), (3, (4, 5, ())))), 展开后返回的 std::vector 中应保存 {1, 1, 2, 3, 1, 2, 3, 3, 4, 5}. 可以使用的 C++ 标准库中的对象 : std::vector, std::stack, 不可以使用递归的方式实现

30. 内存控制题 : 6 分. 现在有一联合开发的项目, 项目使用的程序库使用的是旧式 C++ 标准, 而这个项目要求使用 C++ 标准库的智能指针 std::unique_ptr. 在项目使用的程序库中, 某一个函数要求给定一个类型为 T ** 的参数, 但是 std::unique_ptr 本身并不提供一个成员函数来返回一个指针的饮用. 某个同学想到使用 reinterpret_cast 来实现 :

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

在某些情况下, 这种方法确实可行, 但是这样容易产生为定义行为甚至严重错误, 程式码的可移植性与兼容性也得不到保障. 现在需要设计一个类, 对上述需求进行包装, 提供易用性、安全性和可移植性. 有如下要求 :

①可使用的 C++ 标准库对象 : std::unique_ptr, std::default_delete, std::exchange. 其中, std::exchange 可能的实现为

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

②不可以重新设计 std::unique_ptr 或着更改 std::unique_ptr, 只能设计一个辅助类型的类