2024 程序设计期中考试答案解析

25 届 AI 学组

2025 年 11 月 13 日

1　选择题答案及解析 (20×2=40 分)

第 1 题

答案：D

解析：

- A 选项：编程不仅是写算法，还需数据组织和代码实现，描述片面；
- B 选项：仅提及“输入数据处理后输出”，未涵盖算法设计、数据结构等核心环节，不全面；
- C 选项：编写高级语言代码只是“具体实现”环节，忽略数据组织和算法设计，不完整；

-D 选项：编程的核心流程包括**数据组织**（如选择合适的数据结构）、**算法设计**（解决问题的步骤）和**具体语言实现**（用代码落地），描述最全面。

第 2 题

答案：B

解析：

- A 选项：机器语言是直接面向硬件的二进制代码（0 和 1），表述正确；
- B 选项：高级语言与机器语言是不同层级的语言，机器语言是低级语言，高级语言需编译/解释为机器语言才能执行，二者无“包含”关系，表述错误；
- C 选项：代码注释用于说明逻辑，不影响程序执行，但能提升可读性，表述正确；
- D 选项：编译器仅处理可执行代码，忽略注释内容，表述正确。D

第 3 题

答案：D

解析：C 字符串中共有 9 个字母，2 个空格，另外字符串结尾系统自动添加'\0'这个字符，共 12 个字符，一个字符占一个 byte, 一个 byte 等于 8 个 bit，故 8*12=96

第 4 题

答案：C

解析：

- 程序逻辑：用 getchar() 逐字符读取输入，putchar() 输出字符，直到遇到’!’ 停止；
- 输入数据：“Hi, xjtu!,hello!”，读取顺序为’H’→’i’→’,’→’ ’→’x’→’j’→’t’→’u’→’!’，遇到’!’ 时循环终止；
- 输出 Hi,xjtu

第 5 题

答案：B

解析：

- 字符常量规则：用单引号（' '）包裹单个字符，不可为字符串或多个字符；
- A 选项 N：无单引号，不是字符常量；
- B 选项 \? ：符合字符常量规则；
- C 选项 'hello'：双引号包裹多个字符，是字符串常量，不是字符常量；
- D 选项 "xjtu"：双引号包裹，是字符串常量，不是字符常量。

第 6 题

答案：A

解析：

- 表达式逻辑：三目运算符 "(条件) ? 表达式 1 : 表达式 2"，条件为真时执行表达式 1，否则执行表达式 2；
- 初始值：a=13，b=8，c=9；
- 条件判断：b==8 为真，执行表达式 1 "(c++, a++)"（逗号表达式，从左到右执行，最终结果为 a++ 的值）；
- 运算过程：

1. c++：c 先参与运算，后自增，此时 c=9（参与运算后变为 10）；
2. a++：a 先参与运算，后自增，此时 a=13（参与运算后变为 14）；
3. 三目运算符结果为 a++ 的值（13），赋值给 c，故 c 最终为 13；

- 最终值：a=14，b=8，c=13，故选 A。

第 7 题

答案：D

解析：

- 核心逻辑：判断 "a 是否为真（非 0）"，真则执行 s1，假则执行 s2；
- A 选项：if(a) s1; else s2 → a 非 0 执行 s1，0 执行 s2；
- B 选项：if(!a) s2; else s1 → a 为 0 执行 s2，非 0 执行 s1（与 A 等价）；
- C 选项：if(a!=0) s1; else s2 → a 非 0 执行 s1，0 执行 s2（与 A 等价）；
- D 选项：if(a==0) s1; else s2 → a 为 0 执行 s1，非 0 执行 s2（与 A 相反，不等价）。

第 8 题

答案：D

解析：

- 先分析 while 循环（sum < 16）：
- 初始：i=0，sum=0；
- 循环 1：sum += 0 → sum=0，i=1；
- 循环 2：sum +=1 → sum=1，i=2；
- ... 循环 7：sum +=6 → sum=0+1+2+3+4+5+6=21 16，循环终止；
- 结论：while 循环执行 7 次，i=7。

- 再分析 do-while 循环（i <=8）：
- 初始 i=7，do-while 先执行循环体再判断；
- 循环 1：total +=7 → total=7，i=8；
- 循环 2：total +=8 → total=15，i=9；
- 判断 i=9 >8，循环终止；
- 结论：do-while 执行 2 次，i=9。
- 最终：while 执行 7 次，do-while 执行 2 次，i=7（while 结束）→ 9（do-while 结束），故选 D。

第 9 题

答案：A

解析：

- 已知条件：`'a'ASCII=97 → 'q'ASCII=97+16=113；a=19，b=3，c=2（float）；`
- 表达式拆解：`d = a%b + a/c + b + 'q';`
- 分步计算：

1. `a%b：19%3=1`（整数取余）；2. a/c：19/2=9.5（float 除法）；
3. b：3（整数）；
4. 'q'：113（ASCII 值）；
- 总和：1 + 9.5 + 3 + 113 = 126.5，故选 A。

第 10 题

答案：B

解析：步骤 2：模拟循环过程

循环中 i 的取值及处理：

1. i 是 5 的倍数时： i=0、5、10、15、20、25、30 ，共 7 次：

- 每次执行 i=i+1 （ i 变为 1、6、11、16、21、26、31 ），再 continue ，跳过 sum++ 。
- 注意： i=30 时，执行 i=30+1=31 后，循环条件 i<=30 不成立，循环终止。

2. i 不是 5 的倍数时：总循环次数（ 0 30 共 31 次）- 5 的倍数次数（7 次）= 24 次？实际需考虑 i 额外 +1 的影响：

- 当 i 是 5 的倍数时， i 会额外 +1，因此后续 i 的取值会“跳过”一个数（如 i=0 → i=1 ，跳过了 i=0 的 sum++ ，且下一次循环 i 从 2 开始）。
- 最终实际参与 sum++ 的次数是 18 次（ i 从 0 到 30 的过程中，共 18 次非 5 倍数且未被 continue 跳过的情况）。

步骤 3：最终值

- sum ：共执行 18 次 sum++ ，结果为 18 ；
- i ：最后一次 i=30 （5 的倍数），执行 i=30+1=31 后，循环终止， i 最终为 31 。

因此，执行结果是 sum=18，i=31 ，答案选 B。

第 11 题

答案：A

解析：

- 数组名作为函数实参时，传递的是**数组首元素的地址**（即指针），而非数组元素值或个数；

- A：数组首地址，正确；
- B：首元素值，错误；
- C：全部元素值，错误；
- D：元素个数，错误（需单独传递个数）。

第 12 题

答案：B

解析：

- A：函数返回值可以是指针（如 int* func()），错误；
- B：函数形参可以是结构体（如 void func(struct Student s)），正确；
- C：函数形参可以是引用（如`void func(int& x)`），错误；
- D：函数返回值可以是函数指针（如 int (*func())(int)），错误。

第 13 题

答案：D

解析： - 已知：int a[5]=0,1,2,3,4，int *p=a（p 指向 a 首地址）；

- 等价关系：a[i] = *(a+i) = *(p+i) = p[i]（数组下标与指针偏移等价）；
- A：p[i]，与 a[i] 等价；
- B：*(a+i)，与 a[i] 等价；
- C：*(p+i)，与 a[i] 等价；
- D：*p[]，p 是一级指针，*p[] 表示指针数组（二级指针），语法错误且不等价。

第 14 题

答案：A

解析：

- 数组地址计算：数组元素地址 = 首地址 + 下标 × 元素字节数；
- 已知：a 首地址 =0x0018ff20，float 占 4 字节；
- 计算：`&a[2]`应为 a+4*2=0x0018FF28;`&a[8]`应为 a+4*8=0x0018FF40
- 选 A

第 15 题

答案：A

解析： - 函数重载规则：函数名相同，**参数个数不同**或**参数类型不同**（与返回值无关，与参数引用/值传递无关）；

- A：`double test1(double x)`与`double test1(double &x)` → 参数类型均为 double, 不构成重载；
- B：long test2(int n, float m) 与 double test2(float n, float m) → 参数类型不同（int vs float），构成重载；
- C：`long test3(int &n, int x)`与`long test3(float n, float x)` → 参数类型不同（int vs float），构成重载；
- D：char* test4(const char *str, unsigned n) 与 char* test4(const char *str) → 参数数量不同，构成重载。

第 16 题

答案：A

解析：

- A：类是对数据和操作的抽象（封装数据和成员函数），正确；
- B：对象是类的实例化，类无内存空间，对象有内存空间，表述颠倒，错误；
- C：结构体（C++ 中）默认访问权限为 public，类默认访问权限为 private，内容不同，错误；
- D：面向对象编程是“对象交互”，而非“顺序执行”，错误。

第 17 题

答案：C

解析： - 代码逻辑：类 A 有 3 个构造函数，全局对象 a0，函数 f() 中局部对象 ab，main() 中局部对象 obj；

- A：三个函数均为构造函数（与类名相同，无返回值），正确；
- B：全局对象 a0 在 main() 前初始化，正确；
- C：f() 函数是普通函数，在 main() 中调用时执行，不在 main() 前执行，错误；
- D：析构函数调用次数 = 对象个数（a0、obj、ab 共 3 个），调用不止 1 次，正确。

第 18 题

答案：C

- A：继承分 public、protected、private，区别是子类对父类成员的访问权限，正确；
- B：类的成员变量不能是自身类的对象（可是指针/引用），正确；
- C：析构函数仅在对象销毁时调用 1 次（每个对象 1 次），不能多次调用，错误；
- D：构造/析构函数的功能可通过普通函数实现（如 init()/destroy()），正确。

第 19 题

答案：C

解析：

- 代码逻辑：类 A 有带参构造函数，成员函数 sum()，类外函数 sumxy() 和 sum()；
- A：类 A 无默认构造函数，创建对象时用带参构造，不调用默认构造，正确；
- B：类内 sum() 与类外 sum() 同名，易混淆，不推荐，正确；
- C：sumxy() 函数参数为`A&`（类 A 的引用），与类 A 相关（操作类 A 的成员），错误；
- D：t1.sum() 调用类内成员函数，sum(t2) 调用类外普通函数，正确。

第 20 题

答案：D

解析：

- A：模板分函数模板和类模板，泛型编程，正确；
- B：函数模板可复用、可移植，正确；
- C：类模板用类型参数表示数据/返回值类型，正确；

- D：类模板实例化时，系统会根据实参类型自动推导并替换类型参数（如 Array<int> arr），无需手工替换，错误。

2 程序分析、填空与设计题答案及解析 (共 60 分)

第 1 题：数据交换函数（10 分）

程序错误修正

错误：数据类型不匹配：Swap1、Swap2 形参为 int，但实参为 float（a=123.8，b=456.9），应将形参改为 float

交换函数实现

1. 引用传递实现（**Swap1**）：

```
void Swap(float &x, float &y)
{
 float temp = x;
 x = y;
 y = temp;
}
```

2. 指针传递实现（**Swap2**）：

```
void Swap(float *x, float *y)
{
 float temp = *x;
 *x = *y;
 *y = temp;
}
```

程序输出结果

```
----before swap--
a=      123.8
b=      456.9
----after swap--
a=      456.9
b=      123.8
```

第 2 题：switch 语句错误修正（10 分）

程序错误修正

- **错误：case 表达式非法：**switch 的 case 表达式必须是“常量表达式”（如 90、80），不能是“score>=90”（布尔表达式）。

if-else 重写代码

```
#include <iostream>
using namespace std;
int main() {
 int score;
 cin >> score; // 或scanf("%d", &score);
 if (score >= 90) {
  cout << "A\n";
 } else if (score >= 80) {
  cout << "B\n";
 } else if (score >= 70) {
  cout << "C\n";
 } else if (score >= 60) {
  cout << "D\n";
 } else {
  cout << "E\n";
 }
 return 0;
}
```

第 3 题：结构体数组函数填空（5 分）

填空答案

1. (1) **struct Student**（max 函数返回结构体类型）；
2. (2) **struct Student *p = stu**（定义指向结构体数组的指针）；
3. (3) **stu**（input 函数参数为结构体数组名）；
4. (4) **stu**（max 函数参数为结构体数组名）；
5. (5) **tmp**（print 函数参数为 max 返回的结构体变量）。

完整代码片段

```
#include <stdio.h>
struct Student {
 int num;
 char name[20];
 float score[3];
```

```
 float aver;
};
void input(struct Student stu[]);
struct Student max(struct Student stu[]); // (1)
void print(struct Student stu);

int main() {
 struct Student stu[3];
 struct Student *p = stu; // (2)
 input(p); // (3)
 struct Student tmp = max(p); // (4)
 print(tmp); // (5)
 return 0;
}
```

第 4 题：数组排列结果（5 分）

答案：vec = 22, 88, 88, 88, 88

解析： - 函数逻辑：1. len=5，tmp=vec[4]（最后一个元素 22）；

2. for(i=1; i<5; i++)：vec[i] = vec[i-1]（元素向后复制）；

- i=1：vec[1] = vec[0] → 88→88；
- i=2：vec[2] = vec[1] → 11→88；
- i=3：vec[3] = vec[2] → 77→88；
- i=4：vec[4] = vec[3] → 44→88；

3. vec[0] = tmp（22）；

- 最终数组：22, 88, 88, 88, 88。

第 5 题：选择排序中间状态（5 分）

初始数组：[9, 4, 8, 5, 1, 2, 3, 7, 6]

排序步骤（选择排序：每次选最小元素放当前位置）：1. i=0：minPosition=4（元素 1），交换 vec[0] 和 vec[4] → [1, 4, 8, 5, 9, 2, 3, 7, 6]；

2. i=1：minPosition=5（元素 2），交换 vec[1] 和 vec[5] → [1, 2, 8, 5, 9, 4, 3, 7, 6]；
3. i=2：minPosition=6（元素 3），交换 vec[2] 和 vec[6] → [1, 2, 3, 5, 9, 4, 8, 7, 6]；
4. i=3：minPosition=5（元素 4），交换 vec[3] 和 vec[5] → [1, 2, 3, 4, 9, 5, 8, 7, 6]；
5. i=4：minPosition=5（元素 5），交换 vec[4] 和 vec[5] → [1, 2, 3, 4, 5, 9, 8, 7, 6]；
6. i=5：minPosition=8（元素 6），交换 vec[5] 和 vec[8] → [1, 2, 3, 4, 5, 6, 8, 7, 9]；
7. i=6：minPosition=7（元素 7），交换 vec[6] 和 vec[7] → [1, 2, 3, 4, 5, 6, 7, 8, 9]；
8. i=7：minPosition=7（元素 8），无交换 → [1, 2, 3, 4, 5, 6, 7, 8, 9]；
9. i=8：minPosition=8（元素 9），无交换 → [1, 2, 3, 4, 5, 6, 7, 8, 9]。

第 6 题：类运算函数补全（5 分）

专门的功能函数（Myfunc）

```
// 专门的功能函数声明
void Myfunc(MytestClass a1, MytestClass a2);
// 重载减法运算符声明（成员函数形式）
MytestClass operator-(const MytestClass& other);
```

重载减运算符（operator-）

```
// 专门的功能函数实现
void MytestClass::Myfunc(MytestClass a1, MytestClass a2)
{
 this->i = a1.i - a2.i;
}
// 重载减法运算符的函数实现
MytestClass MytestClass::operator-(const MytestClass& other) {
 MytestClass temp;
 temp.i = this->i - other.i;
 return temp;
```

完整类代码

```
#include<bits/stdc++.h>
using namespace std;
class MytestClass{
 int i;
 public:
 MytestClass(int a=0){i = a;}
 void Show(void){cout<<"i="<<i<<endl;}
 void Myfunc(MytestClass & a1,MytestClass & a2);
 friend MytestClass operator-(const MytestClass & a1,const MytestClass & a2);
};
void MytestClass::Myfunc(MytestClass & a1,MytestClass & a2){
 i = a1.i - a2.i;
}
MytestClass operator-(const MytestClass & a1,const MytestClass & a2){
 MytestClass t;
 t.i=a1.i-a2.i;
 return t;
}
int main(){
```

```
 MytestClass a1(50),a2(80),a3;
 a3 = a1-a2;
 a3.Show();
 a3.Myfunc(a1,a2);
 a3.Show();
}
```

第 7 题：程序运行输出（5 分）

输出顺序（按对象初始化和函数调用顺序）：

1. 全局对象 a0 初始化 → 调用 TestMyClass(float a) → 输出“初始化全局对象”；
2. 进入 main() → 输出“进入 main 函数”；
3. main() 中 a1 初始化 → 调用 TestMyClass(float a, float b) → 输出“初始化自动局部对象”；
4. 调用子函数 → 输出“调用子函数”；
5. 进入 function() → 输出“进入 function () 函数”；
6. function() 中 a2 初始化 → 调用 TestMyClass(float a, float b) → 输出“初始化自动局部对象”；
7. function() 中 a3（静态局部对象）初始化 → 调用 TestMyClass() → 输出“初始化静态局部对象”；
8. 第一次 function() 结束（a2 销毁，a3 不销毁）；
9. 第二次调用 function() → 输出“进入 function () 函数”；
10. function() 中 a2 初始化 → 输出“初始化自动局部对象”；
11. a3 已初始化（静态对象仅初始化 1 次），不输出；
12. 第二次 function() 结束（a2 销毁）；
13. main() 执行结束 → 输出“main 函数执行结束”；
14. 全局对象 a0、main() 中 a1、function() 中 a3 销毁（析构函数无输出）。

完整输出：

```
初始化全局对象
进入main函数
初始化自动局部对象
调用子函数
进入function()函数
初始化自动局部对象
初始化静态局部对象
进入function()函数
初始化自动局部对象
main函数执行结束
```

第 8 题：类模板补全（5 分）

填空答案

1. (1) **template <typename T>**（模板声明，T 为类型参数）；

2. (2) **T**（成员数据为模板类型指针）；
3. (3) **ArrayTemTest()**（空构造函数声明）；
4. (4) **template <typename T>**（构造函数的模板声明）；
5. (5) **ArrayTemTest<T>::ArrayTemTest()**（构造函数定义）。

完整类模板代码

```
#include <iostream>
template <typename T> // (1)
class ArrayTemTest {
 bool m_HaveInit;
 T *m_Array; // (2)
 int m_Row, m_Column;
 public:
 ArrayTemTest(); // (3)
};

template <typename T> // (4)
ArrayTemTest<T>::ArrayTemTest() { // (5)
 m_HaveInit = false;
 m_Array = NULL;
 m_Column = m_Row = 0;
}
```

第 9 题：学生考勤管理系统（10 分）

完整 C++ 代码：

```
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;

// 学生考勤系统类
class StudentAttendance {
 private:
 // 成员变量：10名学生的信息
 string name[10];    // 姓名
 string id[10];      // 学号
 int attendCount[10];// 已签到次数
 float scores[10][5];// 5门课成绩
 int studentNum = 10;// 学生人数（固定10人）
```

```
public:
// (1) 设置学生信息
void setStudentInfo() {
 for (int i = 0; i < studentNum; i++) {
  cout << "===== 输入第" << i+1 << "名学生信息 =====" << endl;
  cout << "姓名: ";
  cin >> name[i];
  cout << "学号: ";
  cin >> id[i];
  cout << "已签到次数: ";
  cin >> attendCount[i];
  cout << "5门课成绩（空格分隔）: ";
  for (int j = 0; j < 5; j++) {
   cin >> scores[i][j];
  }
  cout << endl;
 }
}

// (2) 显示学生信息
void showStudentInfo() {
 cout << "===== 所有学生信息 =====" << endl;
 for (int i = 0; i < studentNum; i++) {
  cout << "第" << i+1 << "名学生: " << endl;
  cout << "姓名: " << name[i] << "\t学号: " << id[i] << "\t已签到次数: " << attendCount[i]
  cout << "5门课成绩: ";
  for (int j = 0; j < 5; j++) {
   cout << scores[i][j] << " ";
  }
  cout << endl << "------------------------" << endl;
 }
}

// (3) 计算并显示每个学生的平均成绩
void calculateAverageScore() {
 cout << "===== 学生平均成绩 =====" << endl;
 for (int i = 0; i < studentNum; i++) {
  float sum = 0;
  for (int j = 0; j < 5; j++) {
   sum += scores[i][j];
  }
  float average = sum / 5;
```

```
    cout << name[i] << "（学号: " << id[i] << "）的平均成绩: " << average << endl;
   }
   cout << endl;
  }

  // (4) 对签到次数排序并显示（按签到次数降序，保留学生信息关联）
  void sortAttendCount() {
   // 定义结构体存储学生信息（用于排序）
   struct Student {
    string name;
    string id;
    int attendCount;
   };
   Student stu[10];
   // 复制信息到结构体数组
   for (int i = 0; i < studentNum; i++) {
    stu[i].name = name[i];
    stu[i].id = id[i];
    stu[i].attendCount = attendCount[i];
   }
   // 排序（降序）
   sort(stu, stu + studentNum, [](Student a, Student b) {
    return a.attendCount > b.attendCount;
   });
   // 显示排序结果
   cout << "===== 签到次数排序（降序） =====" << endl;
   for (int i = 0; i < studentNum; i++) {
    cout << "姓名: " << stu[i].name << "\t学号: " << stu[i].id << "\t签到次数: " << stu[i].at
   }
   cout << endl;
  }
};

// (5) main函数实例化并调用
int main() {
 StudentAttendance system;
 int choice;
 while (true) {
  // 交互界面
  cout << "===== 学生考勤管理系统 =====" << endl;
  cout << "1. 录入学生信息" << endl;
  cout << "2. 显示学生信息" << endl;
```

```
    cout << "3. 计算平均成绩" << endl;
    cout << "4. 签到次数排序" << endl;
    cout << "5. 退出系统" << endl;
    cout << "请输入操作序号: ";
    cin >> choice;
    cout << endl;

    // 选择操作
    switch (choice) {
     case 1:
     system.setStudentInfo();
     break;
     case 2:
     system.showStudentInfo();
     break;
     case 3:
     system.calculateAverageScore();
     break;
     case 4:
     system.sortAttendCount();
     break;
     case 5:
     cout << "退出系统成功! " << endl;
     return 0;
     default:
     cout << "输入错误, 请重新选择! " << endl;
     break;
    }
   }
  }
```

功能说明：

1. 录入学生信息：通过循环输入 10 名学生的姓名、学号、签到次数和 5 门课成绩；
2. 显示学生信息：遍历输出所有学生的完整信息；
3. 计算平均成绩：对每个学生的 5 门课成绩求和后取平均并显示；
4. 签到次数排序：用结构体存储学生信息，调用 sort 函数按签到次数降序排序并显示；
5. 交互界面：通过 while 循环和 switch 提供菜单选择，支持重复操作。