C++primer 第五版 第二章 答案

2.1 答：<1>他们在不同的系统中其存储空间是有所差别，在 32 位系统中 short、int、long、long long 分别占 2、4、4、8 个 byte（字节）。C++语言规定一个 int 型至少和一个 short 型一样大，一个 long 型至少和一个 int 一样大，一个 long long 至少和一个 long 型大，其中 long long 是在 c++ 11 中新定义的。double、float 都是浮点型，double（双精度型）比 float（单精度型）存的数据更准确些，占的空间也更大。在 32 位系统中 float 占 4 个字节（byte），double 占 8 个字节。

<2>带符号可以表示正数、负数、0，无符号只能用来表示大于 0 的值。

2.2 答：对于利率、本金和付款选择数据类型根据数据的精度需要来选择。一般选择利率选择 float（一般保留小数点后 2 位数，单精度足够），本金选择 long (现在都是有钱人，10 位数满足不了那些土豪们就用 long long 吧，呵呵)，付款额一般为实数,可以选择 double 类型。

2.3 答：32 、2^32-32、32、-32、0、0

2.5 答：（a）'a'为 char 型字面值，L'a'为 wchar_t 型字面值，"a"为字符串型字面值，L"a"为宽字符串型字面值。

（b）10 为 int 型字面值， 10u 为 unsigned 型字面值， 10L 为 long 型字面值，10uL 为 unsigned long 型字面值， 012 为八进制表示的 int 型字面值， 0xC 为十六进制表示的 int 型字面值。

（c）3.14 为 double 型字面值，3.14f 为 float 型字面值，3.14L 为 long double 型。

（d）10 为 int 型字面值，10u 为 unsigned 型字面值，10.为 int 字面值，10e-2（0.1）为 foat 型字面值。

2.6 答：int month=9，day=7；与 int month=09，day=07；都是 int 型但是第二种方法是用八进制，所 09 会出错，八进制范围为 0~7；可以改为 int month=011；

2.7 答：（a）145 表示字符 e，012 表示换行符 （b）31.4 为 long double 型

（c）1024f 为 float 型 （d）3.14L 为 long double

2.8 答：cout << "2\115\n" << endl; cout << 2 << endl; cout << "\115" << endl;

2.9 答：（a）错误，应该先定义。（b）这种方式不能进行转换

（c）正确（d）可以，转换成功过后小数部分丢失。

2.10 答：global_str 、local_str 为空字符串，global_int 默认初始化为 0，local_int 没有初值。

2.11 答：extern int ix=1024；定义 int iy；声明并定义 extern int iz；声明

2.12 答：（a）关键字不能作为变量名（b） int _i;在函数体外这种写法是错误的，在函数体内可以用下划线开头的变量名。（c）int catch-22；错误。（d）正确，可以用下划线分割多个单词（e） 正确，变量名区分大小写；

2.13 答：i=j=100；

2.14 答：合法，输出结果为 i 的值为 100,和为 45。for 语句中定义的变量 i，其作用域仅限于 for 语句内部，输出的 i 值是 for 语句之前所定义的变量 i 的值。

2.15 答：合法、非法、合法、非法。引用必须初始化，引用是对对象的引用，不能绑定一个字面值。引用并非对象，相反的它只是为一个已经存在的对象所起一个别名（绰号）。

2.16 答：都合法。

2.17 答：输出结果为：i 和 r1 的值都为 10.

2.18 答：#include<iostream>

using namespace std;

int main()

```
{
    int *p = 0;
    int ival1= 1;
    int ival2 = 2;
    p = &ival1;
    cout << *p << endl;
    p = &ival2;//改变指针指向的对象
    cout << *p << endl;
    ival2 = 3;//改变指针指向的对象的值
    cout << *p <<endl<< ival2 << endl;
}//输出结果为：1、2、3、3.
```

2.19 答：指针和引用都能提供对其他对象的间接访问，然而在具体实现细节上二者有很大的不同，其中最重要的一点就是引用本身并非一个对象，一旦定义了引用，就无法令其再绑定到另外的对象上去，之后每次使用这个引用都是访问它最终绑定的那个对象。

指针和它存放的地址之间就没有这种限制了。和其他任何变量（只要不是引用）一样，给指针赋值就是令它存放一个新的地址，从而指向一个新的对象。

2.20 答：指针指向对象 i，然后把指针指向的对象 i 的值翻倍。输出 i 的值为 42 的平方。

2.21 答：（a）和（b）非法。请查看相关的类型转换规则。（c）正确

2.22 答：if（p）判断指针 p 本身是否合法，if（*p）判断指针 p 所指向的内容。

2.23 答：不能，要使指针指向有效的对象是程序员所做的事情，如果让它指向一个有效的对象,它就是有效的，如果它指向的对象已经无效，则将其置为 NULL；而在使用这个指针之前，应该判断它是否为 NULL！。

2.24 答：void*是一种特殊的指针类型，可以用于存放任意类型对象的地址，而 long*型不能用来存放 int 类型的地址。

2.25 答：（a）ip 是未初始化 int 型指针，i 是未初始化 int 的变量，而 r 是对象 i 的引用。

（b）i 是未初始化 int 型的变量，ip 是一个初始化为 int 型的空指针。

（c）ip 是未初始化的 int 型指针，ip2 是未初始化的 int 的型变量。

2.26 答：（a）未初始化（b）定义并初始化 cnt（c）用 cnt 初始化 const int 型 sz（d）++cnt 合法，++sz 非法，因为 sz 是 const int 型常量。

2.27 答：（a）int i= -1 合法，&r=0；非法；（b）合法；（c）const int I =-1；合法；const int &r=0；合法（d）合法；（e）合法；（f）未初始化，不合法；（g）合法

2.28 答：（a）cp 未初始化；（b）p1 未初始化的指针，p2 为 const 指针，未初始化；（c）合法；（d）未初始化；（e）未初始化；

2.29 答：（b）非法，指针类型不一样；（c）非法，p1 为非 const 指针，ic 为 const 对象；（d）非法 P3 是 const 型指针，其值不能被改变；（e）非法，p2 为 const 型指针，其值不能改变（f）非法，ic 的值不能改变。

2.30 答：cosnt int v2=0；顶层 int v1=v2；合法 v2 是一个顶层 const；

const int *p2 = &v2；p2 是底层 const，v2 是一个顶层 cosnt；

const int *cosnt p3=&i；靠右的 const 是顶层 const，靠左的是个底层 cosnt；

const int &r2 =v2；是个底层 const；

2.31 答：r1=v2；合法，可以用 cosnt int v2 用来初始化 r1；

P1=p2；非法，p2 包含一个底层 const，不能忽略；

P2=p1；合法，p1 不包含 cosnt；

P1=p3；非法，p3 既包含底层又包含顶层 const，p1 不含底层 const；

P2=p3；合法，都包含底层 const，可以忽略掉 p2 顶层 const；

2.32 答：int null=0；*p=null；这是非法的，因为这里 null 是一个 int 变量，不能把 int 变量直接赋值给指针，哪怕这 int 变量恰好等于 0 也不行；应该改成 int*p=nullptr；或 int*p=0；

2.33 答：d 是一个整形指针，运行错误；e 是一个指向整型常量的指针，运行错误；g 是一个整型常量的引用，运行错误；

2.34 答：参考书上给伪代码测试就可以。

2.35 答　auto j = i;//j 是 cosnt int 型

const auto &k = i;//k 是 const int &型

auto *p = &i;//p 是 int *型

const auto j2 = i, &k2 = i;//j2 是 const int，k2 是 const int &型

2.36 答：运行结果为：a=4；　b=4；c=4；d=4；

2.37 答：a=3；b=4；c=3；d=4；

2.38 答：const std::vector<int> ival(100);

auto a = ival[0];// a 为 int 类型

decltype(ival[0]) b =1;

// b 为 const int& 类型，std::vector<int>::operator[]（size_type）const 的返回类型

auto c = 1;// c 为 int 类型

auto d = c;// d 为 int 类型

decltype(c) e;// e 为 int 类型，c 为 int 型

decltype((c)) f = e;// f 为 int& 型，因为（c）是左值

decltype(0) g;// g 为 int 类型，因为 0 是右值

2.39 答：编译器会提示分号遗漏；

2.40 答：class Sales_data

```
{
public :
    string bookNo;
    unsigned units_sold=0;
    double revenue=0.0;
};
```

2.41 答：

重写 1.21 输出两个对象的和：

```
#include<iostream>
#include<string>
using namespace std;
class Sales_data
{
public :
    string bookNo;
    unsigned units_sold=0;
    double revenue=0.0;
};
int main()
{
    Sales_data data1, data2;
```

```
    double price = 0;
    std::cin >> data1.bookNo >> data1.units_sold >> price;
    data1.revenue = data1.units_sold * price;
    std::cin >> data2.bookNo >> data2.units_sold >> price;
    data2.revenue = data2.units_sold * price;
    if (data1.bookNo == data2.bookNo)
 {
        unsigned totalCnt = data1.units_sold + data2.units_sold;
        double totalRevenue = data1.revenue + data2.revenue;
        std::cout << data1.bookNo << " " << totalCnt<< " " << totalRevenue << " ";
        if (totalCnt != 0)
            std::cout << totalRevenue / totalCnt << std::endl;
        else
            std::cout << "(no sales)" << std::endl;
        return 0;
    }
else
 {
        std::cerr << "Data must refer to the same ISBN"<< std::endl;
        return -1;
}
}
```

重写 1.22 读取多个相同的 ISBN 的销售记录，输出所有的和

```
#include <iostream>
#include <string>
using namespace std;
struct Sales_data
{
    string bookNo;
    unsigned units_sold = 0;
    double revenue = 0.0;
};

int main()
{
    double price= 0;
    Sales_data data1;

    if (cin >> data1.bookNo >> data1.units_sold >> price)
    {
        data1.revenue = data1.units_sold*price;
        Sales_data data2;
        while (std::cin >> data2.bookNo >> data2.units_sold >> price)
        {
```

```
            if (data1.bookNo == data2.bookNo)
            {
                data1.units_sold +=    data2.units_sold;
                data1.revenue = data1.revenue + data2.units_sold*price;
            }
        }
    }
    cout << data1.units_sold << endl;
    cout << data1.revenue << endl;
    return 0;
}
```

重写 1.23：读取多条销售记录，统计每个 ISBN 有几条销售记录

```
#include <iostream>
#include <vector>
#include <string>
using namespace std;
struct Sales_data
{
    string bookNo;
    unsigned units_sold = 0;
    double revenue = 0.0;
};

int main()
{

    Sales_data data1;
    Sales_data data2;
    if (cin >> data1.bookNo)
    {
        int cnt = 1;
        while (cin >> data2.bookNo)
        {

            if (data1.bookNo == data2.bookNo)
            {
                ++cnt;
            }
            else
            {
                cout << data1.bookNo <<"卖了"<< cnt <<"本"<< endl;
                data1.bookNo = data2.bookNo;
                cnt = 1;
            }
```

```
            }
            cout << data1.bookNo << "卖了" << cnt << "本" << endl;
        }
        return 0;
}
```

2.42 答：就是把 sales_data 类放在头文件中，这么简单这个不用写出来吧。