第2章 练习

2.1节

2.1

```
1. 类型占用的字节数不同，long long >= long > int > short;
2. 带符号类型最高位是符号位，可能表示的数字是无符号类型的一半
3. 表示的精度不同
```

2.2

```
double  浮点数推荐使用double类型
```

2.3 - 2.4

```
32; 2^32-32;
32; -32; 0; 0;
```

2.5

```
1. 字符；  宽字符；  字符串数组；  宽字符串数组
    > 宽字符类型，每个字符占用两个字节
2. int; unsigned; long; unsigned long; int; int
3. double; float; long double;
4. int; unsigned; double; double;
```

> 可以利用auto来字面值的数据类型

2.2节

2.9

```
1. 错；  std::cin>> intput_value;
2. 错；  花括号初始化不能发生类型转换
3. 可对可错；如果wage是已经定义过的变量就是对的，否则就是错的
4. 对
```

2.10

```
global_str   空字符串
global_int   0
local_int    未定义的值
local_str    空字符串      (string类有默认构造函数)
```

2.11

```
1. 定义
2. 定义
3. 声明
```

> int a; int a;
>
> 在C中这种写法是支持的，理解为"tentative definitions"；在C++中是不允许的，int a被认为是定义语句

2.12

```
1. 不合法;  double是关键字        2. 合法
3. 不合法;  不支持 -              4. 不合法;  以字母或下划线开头
5. 合法
```

2.13

```
j = 100; 使用的是main()函数作用域内的i初始化j
```

2.14

```
合法;
输出:  100 45
```

2.3节

2.15

```
1. 合法;        2. 不合法；不能将非常量引用绑定
3. 合法;        3. 不合法；引用必须初始化
```

2.16

```
1. 合法;        2. 合法
3. 合法;        4. 合法
```

> 引用初始化后，对它操作就相当于对它指向的对象操作，允许类型转换

2.17

```
10 10
```

2.18

```
int a = 1;
int *p = nullptr;
p = &a;      // 更改指针的值
*p = 2;      // 更改指针所指对象的值
```

2.19

```
指针是一个对象，有内存，可以为空；
引用不是一个对象，不能为空
```

2.20

```
i = i*i;
```

2.21

```
1. 不合法；  类型不匹配；
2. 不合法；  需要取地址
3. 合法；
```

> 尽管 int *p = 0;是正确的；
>
> 但是 int *p = a;(a = 0)是不对的

2.22

```
1. 如果p是一个空指针，则为false;
2. 如果p指向的对象=0，则为false;
```

2.23

```
不能；  内存方面的检验很难实现
```

> "要是能轻易做到的话C程序就不容易出内存方面的bug了" — 知乎答主

2.24

```
void* 指针可以指向任何类型的变量；
long* 指针指向的对象必须类型匹配
```

2.25

```
1．指向int类型的指针（未定义）；int(未定义)；int型引用（i)
2．int(未定义)；指向int的指针（指针值=0)
3．指向int类型的指针（未定义）；int(未定义)；
```

2.4节

2.26

```
1．不合法；常量必须被显式初始化；        2．合法
3．合法；                                4．++cnt不合法；  ++sz合法
```

2.27

```
1．int &r=0不合法；       2．合法
3．合法；                 3．合法
4．合法                   5．不合法；  &const不合法
6．合法
```

2.28

```
1．不合法；       2．不合法
3．不合法；       3．不合法
常量对象必须显式初始化
4．合法
```

2.29

```
1．合法；         2．不合法；  底层const限制
3．不合法；指针类型和对象类型不匹配      4．不合法；顶层const限制
5．不合法；顶层const限制          6．不合法；ic是个常量
```

2.30

```
顶层：v2 p3
底层：p2 p3
```

2.31

```
合法；
p1=p2不合法，类型不匹配；  p2=p1合法
p1=p3不合法，  p3底层const限制；  p2=p3合法
```

2.32

```
不合法
int null = 0 , *p = nullptr;
```

2.5节

2.33 - 2.34

```
对，a是整数；对，b是整数；  对，c是整数
错，d是整型指针；错，e是指向整数常量的指针；错，g是一个整型常量引用
```

2.35

```
j - int j;  k - const int &k; p是 const int *p;
j2 - const int j2;  k2 - const int &k2
```

2.36

```
int c = 3;
int &b = a;
c = 4
d = 4
a = 4;
b = 4;
```

2.37

```
int c = 3;
int &d = a;      // a=b是一个左值
```

2.38

```
主要的区别有两点：

1：如果使用引用类型，auto会识别为其所指对象的类型，decltype则会识别为引用的类型。

2：decltype(())双括号的差别
```

2.6节

```cpp
//
// Created by Yjyu on 2022/1/14.
//

#ifndef CHAPTER_1_SALES_ITEM_H
#define CHAPTER_1_SALES_ITEM_H

class Sales_item {
    friend std::ostream &operator<<(std::ostream &os, const Sales_item &item);
    friend std::istream &operator>>(std::istream &is, Sales_item &item);
    friend Sales_item operator+(const Sales_item item1, const Sales_item item2);
public:
    const std::string &getIsbn() const;

    void setIsbn(const std::string &isbn);

    int getNum() const;

    void setNum(int num);

    double getPrice() const;

    void setPrice(double price);



private:
    std::string ISBN_;
    int num_;
    double price_;

    double sum_profit_;

    void sum_profit() {
        sum_profit_ = num_*price_;
    }

};
```

```
39 #endif //CHAPTER_1_SALES_ITEM_H
```