第二次小班课

计算机系统导论 (Class 9)

老师: 汪小林
助教: 陈东武

北京大学 信息科学技术学院

2024 年 09 月 18 日

回课补充

数据表示

- 补码表示: 模 2^n 运算.
  - 有符号数范围: 从 -2^{n-1} 到 $2^{n-1}-1$.
- 浮点数表示: 符号位 + 指数位 + 小数位.
  - 指数范围: 从 $2-2^{k-1}$ 到 $2^{k-1}-1$.
- 大端序与小端序: 多字节对象中各个字节的编号顺序.
  - 机器内部计算一般使用小端序, 网络传输使用大端序.
  - 多字节对象的地址是其中最小的地址.

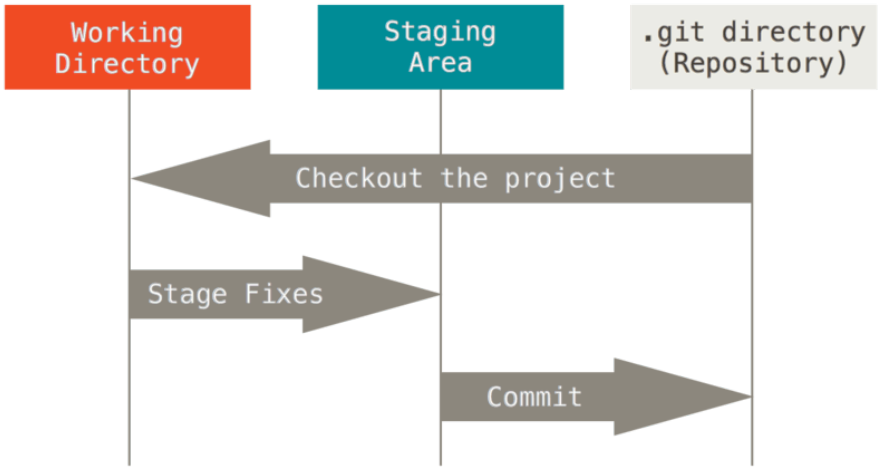
运算规则

- 类型提升规则: 在计算 C 语言表达式时, 可能进行隐式类型转换.
  - (unsigned) char 和 (unsigned) short 会被提升为 int 类型.
  - 在不同类型的运算中, 低级类型会被提升为高级类型.
  - double > float > unsigned long long > long long > unsigned > int.
- 舍入规则: 向偶数舍入.

Git 介绍

什么是 Git?

- Git 是一个**版本控制工具**, 用来管理和追踪一个软件的源文件版本.
  - 将项目的历史记录组织为**有向无环图**.
  - GitHub 与 Git 不同, 是基于 Git 的网盘代码托管平台.
- 仓库包含三个目录: **工作区**, **暂存区**和**版本库**.
  - 工作区就是项目目录, 你可以随意修改.
  - 暂存区对修改进行缓存, 直至你决定提交到版本库.



创建, 修改与提交

- 在 GitHub 上创建仓库, 跟随指引即可.
  - `git init` 在当前文件夹创建 git 仓库.
  - `git clone <path/to/repository>` 复制远程仓库.
- `git add -A` 将所有文件添加到暂存区.
  - 使用 `.gitignore` 排除临时文件和敏感文件.
- `git commit -m "提交信息"` 将暂存区的修改提交到版本库.
  - 相当于在历史记录图中创建一个结点.
- `git push origin main` 将本地的记录上传到远程仓库.
  - 需要先 `git remote add origin <server>` 添加远程仓库.
  - 还有配置 SSH 密钥, 类似 CLab 的 SSH 连接.

分支, 合并与撤销

- `git checkout -b <branch>` 创建一个新的分支并切换.
  - 没有 -b 选项就是切换到旧的分支.
- `git pull` 将远程仓库合并到本地项目.
- `git merge <branch>` 将其他分支合并到当前分支.
  - 在发生冲突时手动合并.
- `git checkout <hash>` 回退到指定历史版本.

Exercise

```c
#include <stdio.h>
int main() {
  double pi = 0.0;
  for (int k = 0; k >= 0; ++k)
    pi += (k & 1 ? -1 : 1) / (double)(2 * k + 1);
  pi *= 4;
  printf("%lf\n", pi);
}
```

- 采用 -Og 优化, 该程序的输出最接近选项中的哪个数?
  - A. 0 B. $\pi/2$ C. π D. 2π

$$\sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} = \frac{\pi}{4}.$$

```c
#include <stdio.h>
int main() {
  double pi = 0.0;
  for (int k = 0; k >= 0; ++k)
    pi += (k & 1 ? -1 : 1) / (double)(2 * k + 1);
  pi *= 4;
  printf("%lf\n", pi);
}
```

- 采用 -Og 优化, 该程序的输出最接近选项中的哪个数?
  - A. 0　　B. $\pi/2$　　C. π　　D. 2π

$$\sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} = \frac{\pi}{4}.$$

```c
#include <stdio.h>
int main(){
  int x = 33554466; // 2^25 + 34
  int y = x + 8;
  for (; x < y; ++x){
    float f = x;
    printf("%d ", x - (int)f);
  }
}
```

- 采用 -Og 优化, 该程序的输出为:

```c
#include <stdio.h>
int main(){
  int x = 33554466; // 2^25 + 34
  int y = x + 8;
  for (; x < y; ++x){
    float f = x;
    printf("%d ", x - (int)f);
  }
}
```

- 采用 -Og 优化, 该程序的输出为: `2 -1 0 1 -2 -1 0 1`

- _____ 的类型转换既可能导致溢出, 又可能导致舍入.
  - A. int 到 float　　B. float 到 int
  - C. int 到 double　　D. float 到 double

浮点数

- _____ 的类型转换既可能导致溢出, 又可能导致舍入.
  - A. int 到 float　　B. float 到 int
  - C. int 到 double　　D. float 到 double

浮点数

- _____ 的类型转换既可能导致溢出, 又可能导致舍入.
  - A. int 到 float　　B. float 到 int
  - C. int 到 double　　D. float 到 double
- 给定一个实数, 会因为该实数表示为 float 而发生误差.
  - 不考虑 NaN 和 Inf, 该绝对误差的最大值是:

浮点数

- _____ 的类型转换既可能导致溢出, 又可能导致舍入.
  - A. int 到 float　　B. float 到 int
  - C. int 到 double　　D. float 到 double
- 给定一个实数, 会因为该实数表示为 float 而发生误差.
  - 不考虑 NaN 和 Inf, 该绝对误差的最大值是: 2^{103}

浮点数

- _____ 的类型转换既可能导致溢出, 又可能导致舍入.
  - A. int 到 float　　B. float 到 int
  - C. int 到 double　　D. float 到 double
- 给定一个实数, 会因为该实数表示为 float 而发生误差.
  - 不考虑 NaN 和 Inf, 该绝对误差的最大值是: 2^{103}
- 考虑一种符合 IEEE 规范的浮点数格式, 包含 1 个符号位, k 个指数位, n 个小数位.
  - 其中 $k \geq 2, n \geq 1$.
- 请问该浮点数最多能精确表示多少个连续的整数?
  - 使用含 k 和 n 的代数式表示:

浮点数

- _____ 的类型转换既可能导致溢出, 又可能导致舍入.
  - A. int 到 float　　B. float 到 int
  - C. int 到 double　　D. float 到 double
- 给定一个实数, 会因为该实数表示为 float 而发生误差.
  - 不考虑 NaN 和 Inf, 该绝对误差的最大值是: 2^{103}
- 考虑一种符合 IEEE 规范的浮点数格式, 包含 1 个符号位, k 个指数位, n 个小数位.
  - 其中 $k \geq 2, n \geq 1$.
- 请问该浮点数最多能精确表示多少个连续的整数?
  - 使用含 k 和 n 的代数式表示: $\min\left(2^{2^{k-1}+1}-1, 2^{n+2}+1\right)$

#thanks