

# 185.190 Effiziente Programme

## Aufgabe: Hash-Tabelle

Berger G., Hotz-Behofsits C., Reisinger M., Schmidleithner T.

WS12/13

# Ausgangssituation

# Schritt 1

## Vermeidung von Speicherfragmentierung

```
struct hashnode {  
    char *keyaddr;  
    size_t keylen;  
    int value;  
} __attribute__((__packed__));
```

Geringe Verbesserung  $\Rightarrow$  **beibehalten**.

# Schritt 2

## Math-Library

```
sumsq += count*count;
```

ersetzt durch

```
sumsq += pow(count, 2);
```

Verwendung der math-Library (math.h), geringe Verbesserung  $\Rightarrow$   
beibehalten.

# Schritt 3

## Inline Funktionen

```
inline unsigned long hash(char *addr, size_t len)
```

inlining bei den Funktionen. Keine Verbesserung  $\Rightarrow$  entfernt.

# Schritt 4

## Entfernen der next-Pointer

```
struct hashnode *next; /* link ext. chaining */
```

Hinzugabe von **linearem Sondieren**:

```
int position = hash(keyaddr, keylen) & (HASHSIZE-1);
struct hashnode *l; l = ht[position];
while (l != NULL) {
    if (keylen == l->keylen &&
        memcmp(keyaddr, l->keyaddr, keylen) == 0)
        return l->value;
    if (position <= HASHSIZE)
        l = ht[++position];
    else
        break;
}
```

next-Pointer wurden entfernt, stattdessen wurde lineares Sondieren implementiert. Verschlechterung  $\Rightarrow$  zurücksetzen.