

Basic Optimization Techniques

Technical P.I.

① Strings are stored: —

① Stack Area

② Heap Area

③ Data Area

Depend upon declaration.

`char *str = (char*) malloc(100);`

Stack

`char str[] = "KILT";`

Data/Shared area.

② Structure padding :-

`struct st`

`{`
`int a;`

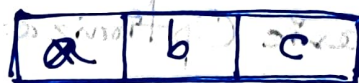
`float b;`

`char c;`

`char d;`

`};`

12 bytes



Undergo till 4 bytes are covered.

According

After 4 bytes completed one more 4 byte space is created.

`struct st`

`{`
`int x;`

`double y;`

`float z;`



12



12

4

1828027

[Signature]

③ Bit Field :-

Memory saving
member of
structure.

$$4 + 4 + \textcircled{4} + 4 \\ = 16$$

struct student

```
{  
    int roll_no:8; 4  
    int age:8; 4  
    char name[20]; 28  
};
```

int roll_no:8;

int:8

Anonymous

→ Bit field
without
any name.

Limitations :-

- * Can't use scanf to initialize.
- * Can't have pointer to bit field.

④ Diff b/w array/structure

⑤ " " st/union.

Basic Optimization Techniques

1. Avoid calculation inside loop.

- Inner loops have min. possible calculations
- Calculations dependent outside loop.

2. Avoid pointer de-reference in loop :-

- Creates lots of trouble in memory.

int temp = *p

for(i)

temp += i;

*p = temp;

1828027

Satpathy

3. Unroll small loops :—

$$\underline{\text{fact}[i] = \text{fact}[i-1] * i;}$$

4. Use register variable as counters :—

register int x;
register int y → CPU register
Little bit fast.

5. Operators Tring :—

— Minimize division

X int d = a/b/c;

✓ int d = a/b*c;

$\frac{a}{b/c} \neq \frac{a*b}{c}$

— Multiplication Bitwise shift.

$x = x/2, x = x >> 1;$
 $x = x*2, x = x << 1;$ } work on binary digit

— Pre-increment more than post increment
— In post increment lots of things happen
→ short circuit AND

(max < b) && (max = b)

If left false, right won't be evaluated.

— Prefer bitwise more than arithmetic :—

$a^1 = b^1 = a^1 = b$;

1828027

Satpathy

Satpathy