

Princeton University

COS 217: Introduction to Programming Systems

C Unions

Problem: We need to define an array of 10 elements, some of which are of type `int` and some of which are of type `double`. What should the element type be?

Solution 1: The generic pointer (`void*`)

```
void *a[10];
int *pi;
double *pd;
...
pi = (int*)malloc(sizeof(int));
*pi = 5;
a[0] = pi;
pd = (double*)malloc(sizeof(double));
*pd = 5.5;
a[1] = pd;
...
printf("%d", *(int*)a[0]);
printf("%f", *(double*)a[1]);
...
free(a[0]);
free(a[1]);
...
```

Problem: Awkward

Problem: How do we know whether an element points to an `int` or a `double`?

Solution 2: A structure

```
struct MyStruct
{
    int i;
    double d;
};
...
struct MyStruct a[10];
...
a[0].i = 5;
a[1].d = 5.5;
...
printf("%d", a[0].i);
printf("%f", a[1].d);
...
```

Problem: Wastes space

Problem: How do we know which of an element's `i` or `d` fields is significant?

Solution 3: A union

```
union MyUnion
{
    int i;
    double d;
};
...
union MyUnion a[10];
...
a[0].i = 5;
a[1].d = 5.5;
...
printf("%d", a[0].i);
printf("%f", a[1].d);
...
```

Problem: How do we know which of an element's i or d fields is significant?

Solution 4: A structure containing a "tag" field and a union

```
enum Kind {KIND_INT, KIND_DOUBLE};

struct MyStruct
{
    enum Kind iKind; /* tag */
    union
    {
        int i;
        double d;
    } u;
};
...
struct MyStruct a[10];
...
a[0].iKind = KIND_INT;
a[0].u.i = 5;
a[1].iKind = KIND_DOUBLE;
a[1].u.d = 5.5;
...
for (i = 0; i < 10; i++)
    if (a[i].iKind == KIND_INT)
        printf("%d", a[i].u.i);
    else
        printf("%f", a[i].u.d);
```

Copyright © 2005 by Robert M. Dondero, Jr.