

Pointers

Examples

```
int i;
int *ptr1;
int *ptr2;

i = 2;
ptr1 = &i;
*ptr1 = 3;
printf("%d, i); /* what gets printed? */
ptr1 = ptr2; /* this is bad - do you know why? */
ptr1 = 2; /* probable error, setting ptr1 to point to memory address 2 */
```

Pass by Value vs Pass by Reference

Pass by Reference

```
mySampleFunction (Mint *i)
```

- Able to modify value of i within mySampleFunction.
- Faster - memory not copied. Preferred method when passing large structures even if structure doesn't need to be modified.

Pass by Value

```
mySampleFunction (Mint i)
```

- NOT able to modify value of i in function.
- Internal copy of i created for use in function - discarded on leaving function.
- Slower.

"Types" vs "Records" convention

"Types" usually mean pointers to "records". "Records" usually mean structures.

Example:

```
typedef struct fpointrecord *fpointtype;

typedef struct fpointrecord {
    Mint flag;
    char visible, selected, identifier;
    Mpoint3 location;
    fpointtype next, previous;
    fnodetype node;
    coveragetype coverage;
} fpointrecord;
```

Examples of use within a function:

```

{
fpointrecord mypoint;
fpointtype newpoint;

mypoint.flag = 1; /* notice we access flag using the dot "." */

feNewFeaturePoint(&coverage, &newpoint, MAP_NODE); /* allocate memory for a fpointrecord */
newpoint->flag = 1; /* notice we access flag using the arrow "->" */
free(newpoint);
}

```

Example of allocating memory:

```

void feNewFeaturePoint (coveragetype *cover, fpointtype *fpoint,
char identifier)

{

*fpoint = (fpointtype) malloc(sizeof(fpointrecord));
...
...
...
}

```

Knowing When To Use . and ->

Example:

```
g_activecoverage->arclist.list
```

Follow the flow:

```

extern coveragetype g_activecoverage; /* g_activecoverage is a coveragetype */

typedef struct coveragerecord *coveragetype; /* coveragetype is a ptr to coverage record */

/* coveragerecord is a structure. arclist is of type farclisttype */
typedef struct coveragerecord {
char visible;
Mint beginlayer;
Mint endlayer;
Mfloat z; /* default z value for new objects (in GMS) */
Mchar name[64];
modelcoveragetype modelcoverage;
mfmtmpcoveragetype mfmtmpmodelcoverage;
fpointlisttype nodelist; /* arc nodes and vertices are stored here */
fpointlisttype pointlist; /* only points not attached to arcs */
farclisttype arclist;
farcgrouplisttype arcgroup;
fpolylisttype polylist;
observationrecord *observe;
fluxrecord flux;
coveragetype next;
} coveragerecord;

```

```
/* farclisttype is a structure. list is an farctype */
typedef struct farclisttype {
    farctype list, last;
    Mint numfarc;
    char openID;
} farclisttype;

typedef struct farcrecord *farctype; /* farctype is a ptr to farcrecord */

/* etc. */
/* etc. */
```

Precedence

1. [] . ->
2. & *

Example: (*labelpointerarray)[2]->value; /* need parentheses */

xdb debugger

When printing a pointer, two values appear. The first is the address of the pointer, the second is the address the pointer is pointing to.

Example:

p templabel

0x34567 0x76543