

計算機序論2

2009/10/19

亀田能成

- 課題AはWWW参照

課題A

- スクリプト読み込みプログラムの完成
- 詳細はWWWのほうを参照

スクリプト読み込みプログラム

- 膨大な仕事を一気にこなすのは大変
- 仕事を分割して(=複数の関数)で処理

規模の大きいプログラム

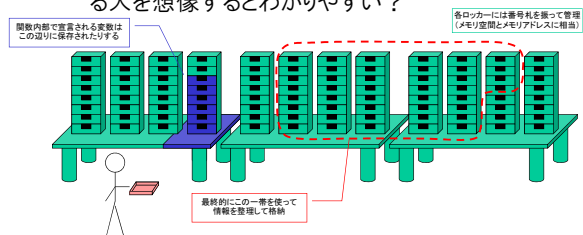
- 仕事だって大きくなれば一気ににはできない
 - 作業を分割する
 - 分割した作業を人に任せる
 - 作業内容はとやかく詮索せずに結果だけ受領
- C言語でも基本は同じ
 - アルゴリズムを分割する
 - 分割したところを別の関数に任せる
 - 関数から結果だけ受領

スクリプト読み込みプログラム 概要

- アルゴリズム概観
 - スクリプトファイルから1行ずつ書かれたデータを1つずつ読み込み、メモリ中のデータ構造に保管
 - できあがったデータ構造を標準出力に書き出し
- 詳細な仕様はWWWのほうを参照のこと

スクリプト読み込みプログラムが 働く様子(想像)

- データ構造はメモリ上に構成される
 - ロッカー(メモリに相当)みたいなもので作業している人を想像するとわかりやすい?



スクリプト読みプログラム チームプレイ(上層部)

- 社長
 - 「ファイルを読みませ結果を標準出力に出させる」作業を部下の『読み』課長にさせる
- 『読み』課長
 - ファイルを1行ずつ読み、読んだ行の中身に合わせて各「データ構造を構築してくれる係長」を呼び出す
 - 『物体』係長
 - 『線分』係長
 - 『パッチ』係長
 - 『アニメ』係長
 - 『光源』係長

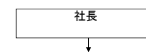
スクリプト読みプログラム チームプレイ(中層部)

- 『物体』係長
 - 新しい物体構造を1つ用意する(だけ)
- 『線分』係長
 - 現在の物体構造中の線分データ集合に線分を1個加える(だけ)
- 『パッチ』係長
 - 現在の物体構造中の三角形パッチ集合に三角形パッチを1個加える(だけ)
- 『アニメ』係長
 - 新しいアニメ構造を1つ用意し、その中身を設定
- 『光源』係長
 - 新しい光源構造を1つ用意し、その中身を設定

スクリプト読みプログラム チームプレイ(下っ端)

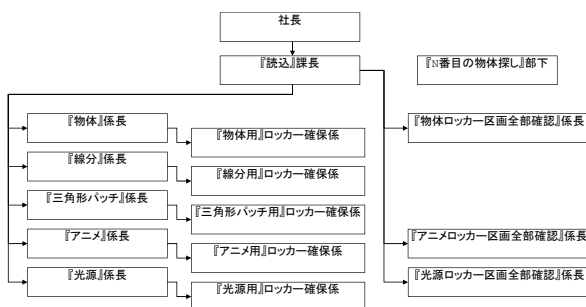
- 『物体』係長
 - 新しい物体構造を1つ用意する(だけ)
 - 『物体用』ロッカー確保アシスタント(係)
 - ロッカーに行って物体1つ分のロッカーを確保して係長にどこを確保したか伝える仕事をする
- 『線分』係長
 - 現在の物体構造中の線分データ集合に線分1個加える(だけ)
 - 『線分用』ロッカー確保アシスタント(係)
 - ロッカーに行って線分1つ分のロッカーを確保して係長にどこを確保したか伝える仕事をする
- (以下同様)

スクリプト読みプログラム 組織図

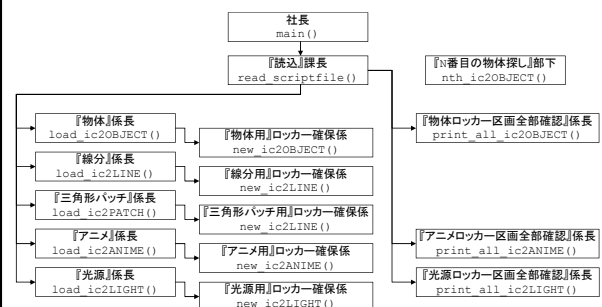


- 社長しかいないような会社(社長しか働かない会社)は大きくなれない

スクリプト読みプログラム 組織図



スクリプト読みプログラム 関数呼出関係図



構造体とポインタ (スクリプト読込プログラムにおける具体例)

目的
構造体の中にポインタ変数を含める
⇒Linked-Listを構築

構造体

- データをもとめて扱えるようにする
 - 変数を幾つでもまとめられる
 - 違う型の変数でもまとめて扱える
- 構造体はintやfloatやcharと同じく、変数を宣言するのに使われる

構造体/定義

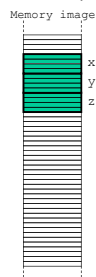
- 例: 3つの浮動小数点数をまとめて空間中の1点を規定

今後は"struct ic2POINT"という型となる

定義

```
struct ic2POINT {
    float x;
    float y;
    float z;
};
```

構造体中の各変数をメンバと呼ぶ。ここではメンバは3つ。



構造体/利用[単純な例]

- 例: 3つの浮動小数点数をまとめて空間中の1点を規定

"struct ic2POINT"という型の変数を使います、ということ。

利用する前にはプログラムの前方で宣言が必要。

宣言

```
struct ic2POINT chouten;
float d;
```

利用

```
d = chouten.x;
chouten.y = d * 2;
```

「ドット」で構造体変数とそのメンバを繋ぐと、メンバの値を参照できる。

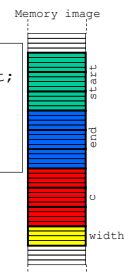
構造体の複雑な例

- 構造体自体も別の構造体に組み込める

```
struct ic2POINT {
    float x;
    float y;
    float z;
};
```

```
struct ic2LINEX {
    struct ic2POINT start;
    struct ic2POINT end;
    struct ic2COLOR c;
    float width;
};
```

```
struct ic2COLOR {
    float r;
    float g;
    float b;
};
```



構造体の複雑な例

- 構造体自体も別の構造体に組み込める

```
struct ic2LINEX {
    struct ic2POINT start;
    struct ic2POINT end;
    struct ic2COLOR c;
    float width;
};
```

```
struct ic2LINEX hasi;
float d;

d = hasi.w / 3.0;
hasi.start.y = d + 1.5;
```

構造体が入れ子になっている場合は、ドット演算子も続けて使う

構造体への演算

- 構造体は変数とはいえ、数値でもなければ文字列でもない(かもしれない)ので、可能な演算の種類は限られる

- 代入

```
struct ic2LINEX hasi;
struct ic2POINT aa, bb;
float d;
```

```
aa = hasi.start;
bb = aa;
bb.z = aa.z * 2.0;
```

- 四則演算はできない(そもそも想像できないでしょ?)

ポインタ・・・の前に

- 変数とメモリ空間との関係を正確に把握することが必要。見慣れたプログラムは、実際、どのように動いているのか？

```
int i;
int j;

i = 7;
j = i * 2;
```

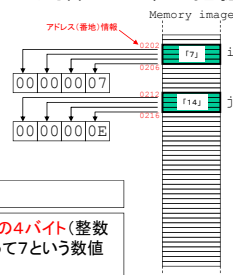
変数の実際

- 変数とメモリ空間との関係を正確に把握することが必要

```
int i;
int j;

i = 7;
j = i * 2;
```

人間: 変数 **i** に7を代入する
コンピュータ: 202番地からの4バイト(整数は4バイトを使うので)を使って7という数値を保存する



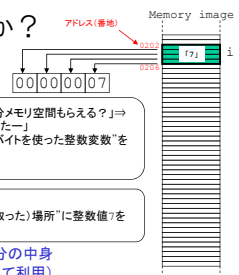
変数の宣言と利用

- 変数の宣言のときに何が起ったのか？
- 変数をどう使っているのか？

```
int i;
i = 7;
```

「これから整数変数を使うから4バイト分メモリ空間もらえる?」
⇒「はい、202番地から4バイト確保しましたー」
「面倒だから、その「202番地からの4バイトを使った整数変数」を呼ぶときには『i』って言うからね」
⇒「了解ですー」
「『i』に整数値7をセットしておいて」
「ええと、つまり「202番地からの4バイト分の(整数のために取った)場所」に整数値7をセットするんですね、了解」

整数変数 **i** = 202番地からの4バイト分の中身
(その中身を整数だと解釈して利用)

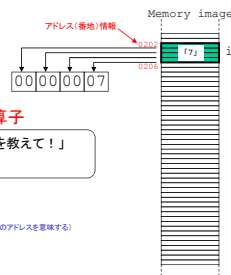


変数のアドレス(アドレス演算子&)

- もし万一、変数が格納されている場所を知りたいと、思ったら？

```
int i;
i = 7;
```

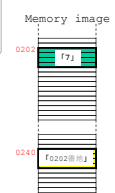
“&” アドレス演算子
「変数 **i** のアドレス(の先頭)を教えて!」
⇒「202番地ですね」
アドレス(番地)を表現している
(※アドレス、番地、住所、場所……この資料ではどれも同じことで、メモリ空間のアドレスを意味する)



ポインタ変数

- アドレスを扱うという概念(が先ほどのスライドで発生)
- 今後、変数として使いたくなる(かも)
- “&i”(例:「202番地」)というアドレス情報をどこかに保存したい

ポインタ変数の宣言
(住所を書きとめるための変数)
⇒240番地からの4バイトを確保
「整数変数 **i** (202番地からの4バイト分)に整数値7を書き込んでおいて」
⇒「了解!」
「どこで整数変数 **i** の住所教えてくれる?」
⇒「202番地(からの4バイト分)です」
「そっか、じゃあそれを書き留めておきたいから、(整数用)ポインタ変数 **p** (240番地からの4バイト分)に「202番地」って書き込んでおいて」
⇒「ラジャー!」



特別なアドレス値 NULL

- 「アドレスが存在しない」状態を表現する

```
int i;
int *p = NULL; ← ポインタ変数を確保して、その内容は「まだどのアドレスでもない」状態をNULLで表現

i = 7;
p = &i;

if (p != NULL)
    printf("num = %d\n", (*p));
```

このような確認は、例えば0/0の演算をする前に0がゼロでないことを確認するようなもので、健全なプログラム作成に必須

間接演算子 * (ポインタ演算子ともいう)

- ポインタ変数を使うようになると、「アドレスしか知らない」状態(=ポインタ変数に格納されたアドレス情報のみ)で、その中身を見たいことがある(かも)
 - 「で結局202番地には何が入ってるのよ?」

```
int i;
int *p;

i = 7;
p = &i;
j = *p;
```

「202番地から1バイト確保」
「202番地から1バイト確保」
「202番地から1バイト確保」
「整数変数i(202番地からの1バイト)に数値7を書き込んでおいて」
⇒「了解!」
「整数変数pの住所「202番地」を(数値用)ポインタ変数p(202番地からの1バイト)に書き込んでおいて」
⇒「了解!」
「どこでポインタ変数pについてもとと数値用だよな」「そうです」「じゃあそのpに書かれている「202番地」を訪ねて、そこに保存されているビット列を**数値値だと見なして**読んできて、整数変数j(212番地からの1バイト)に書き込んでおいてくれる?」
⇒「アイアイサー!」

Memory image
0200 [7]
0210 [7]
0220 [7]
0230 [7]
0240 [7]
0250 [7]
0260 [7]
0270 [7]
0280 [7]
0290 [7]
0300 [7]
0310 [7]
0320 [7]
0330 [7]
0340 [7]
0350 [7]
0360 [7]
0370 [7]
0380 [7]
0390 [7]
0400 [7]
0410 [7]
0420 [7]
0430 [7]
0440 [7]
0450 [7]
0460 [7]
0470 [7]
0480 [7]
0490 [7]
0500 [7]
0510 [7]
0520 [7]
0530 [7]
0540 [7]
0550 [7]
0560 [7]
0570 [7]
0580 [7]
0590 [7]
0600 [7]
0610 [7]
0620 [7]
0630 [7]
0640 [7]
0650 [7]
0660 [7]
0670 [7]
0680 [7]
0690 [7]
0700 [7]
0710 [7]
0720 [7]
0730 [7]
0740 [7]
0750 [7]
0760 [7]
0770 [7]
0780 [7]
0790 [7]
0800 [7]
0810 [7]
0820 [7]
0830 [7]
0840 [7]
0850 [7]
0860 [7]
0870 [7]
0880 [7]
0890 [7]
0900 [7]
0910 [7]
0920 [7]
0930 [7]
0940 [7]
0950 [7]
0960 [7]
0970 [7]
0980 [7]
0990 [7]
1000 [7]
1010 [7]
1020 [7]
1030 [7]
1040 [7]
1050 [7]
1060 [7]
1070 [7]
1080 [7]
1090 [7]
1100 [7]
1110 [7]
1120 [7]
1130 [7]
1140 [7]
1150 [7]
1160 [7]
1170 [7]
1180 [7]
1190 [7]
1200 [7]
1210 [7]
1220 [7]
1230 [7]
1240 [7]
1250 [7]
1260 [7]
1270 [7]
1280 [7]
1290 [7]
1300 [7]
1310 [7]
1320 [7]
1330 [7]
1340 [7]
1350 [7]
1360 [7]
1370 [7]
1380 [7]
1390 [7]
1400 [7]
1410 [7]
1420 [7]
1430 [7]
1440 [7]
1450 [7]
1460 [7]
1470 [7]
1480 [7]
1490 [7]
1500 [7]
1510 [7]
1520 [7]
1530 [7]
1540 [7]
1550 [7]
1560 [7]
1570 [7]
1580 [7]
1590 [7]
1600 [7]
1610 [7]
1620 [7]
1630 [7]
1640 [7]
1650 [7]
1660 [7]
1670 [7]
1680 [7]
1690 [7]
1700 [7]
1710 [7]
1720 [7]
1730 [7]
1740 [7]
1750 [7]
1760 [7]
1770 [7]
1780 [7]
1790 [7]
1800 [7]
1810 [7]
1820 [7]
1830 [7]
1840 [7]
1850 [7]
1860 [7]
1870 [7]
1880 [7]
1890 [7]
1900 [7]
1910 [7]
1920 [7]
1930 [7]
1940 [7]
1950 [7]
1960 [7]
1970 [7]
1980 [7]
1990 [7]
2000 [7]
2010 [7]
2020 [7]
2030 [7]
2040 [7]
2050 [7]
2060 [7]
2070 [7]
2080 [7]
2090 [7]
2100 [7]
2110 [7]
2120 [7]
2130 [7]
2140 [7]
2150 [7]
2160 [7]
2170 [7]
2180 [7]
2190 [7]
2200 [7]
2210 [7]
2220 [7]
2230 [7]
2240 [7]
2250 [7]
2260 [7]
2270 [7]
2280 [7]
2290 [7]
2300 [7]
2310 [7]
2320 [7]
2330 [7]
2340 [7]
2350 [7]
2360 [7]
2370 [7]
2380 [7]
2390 [7]
2400 [7]
2410 [7]
2420 [7]
2430 [7]
2440 [7]
2450 [7]
2460 [7]
2470 [7]
2480 [7]
2490 [7]
2500 [7]
2510 [7]
2520 [7]
2530 [7]
2540 [7]
2550 [7]
2560 [7]
2570 [7]
2580 [7]
2590 [7]
2600 [7]
2610 [7]
2620 [7]
2630 [7]
2640 [7]
2650 [7]
2660 [7]
2670 [7]
2680 [7]
2690 [7]
2700 [7]
2710 [7]
2720 [7]
2730 [7]
2740 [7]
2750 [7]
2760 [7]
2770 [7]
2780 [7]
2790 [7]
2800 [7]
2810 [7]
2820 [7]
2830 [7]
2840 [7]
2850 [7]
2860 [7]
2870 [7]
2880 [7]
2890 [7]
2900 [7]
2910 [7]
2920 [7]
2930 [7]
2940 [7]
2950 [7]
2960 [7]
2970 [7]
2980 [7]
2990 [7]
3000 [7]
3010 [7]
3020 [7]
3030 [7]
3040 [7]
3050 [7]
3060 [7]
3070 [7]
3080 [7]
3090 [7]
3100 [7]
3110 [7]
3120 [7]
3130 [7]
3140 [7]
3150 [7]
3160 [7]
3170 [7]
3180 [7]
3190 [7]
3200 [7]
3210 [7]
3220 [7]
3230 [7]
3240 [7]
3250 [7]
3260 [7]
3270 [7]
3280 [7]
3290 [7]
3300 [7]
3310 [7]
3320 [7]
3330 [7]
3340 [7]
3350 [7]
3360 [7]
3370 [7]
3380 [7]
3390 [7]
3400 [7]
3410 [7]
3420 [7]
3430 [7]
3440 [7]
3450 [7]
3460 [7]
3470 [7]
3480 [7]
3490 [7]
3500 [7]
3510 [7]
3520 [7]
3530 [7]
3540 [7]
3550 [7]
3560 [7]
3570 [7]
3580 [7]
3590 [7]
3600 [7]
3610 [7]
3620 [7]
3630 [7]
3640 [7]
3650 [7]
3660 [7]
3670 [7]
3680 [7]
3690 [7]
3700 [7]
3710 [7]
3720 [7]
3730 [7]
3740 [7]
3750 [7]
3760 [7]
3770 [7]
3780 [7]
3790 [7]
3800 [7]
3810 [7]
3820 [7]
3830 [7]
3840 [7]
3850 [7]
3860 [7]
3870 [7]
3880 [7]
3890 [7]
3900 [7]
3910 [7]
3920 [7]
3930 [7]
3940 [7]
3950 [7]
3960 [7]
3970 [7]
3980 [7]
3990 [7]
4000 [7]
4010 [7]
4020 [7]
4030 [7]
4040 [7]
4050 [7]
4060 [7]
4070 [7]
4080 [7]
4090 [7]
4100 [7]
4110 [7]
4120 [7]
4130 [7]
4140 [7]
4150 [7]
4160 [7]
4170 [7]
4180 [7]
4190 [7]
4200 [7]
4210 [7]
4220 [7]
4230 [7]
4240 [7]
4250 [7]
4260 [7]
4270 [7]
4280 [7]
4290 [7]
4300 [7]
4310 [7]
4320 [7]
4330 [7]
4340 [7]
4350 [7]
4360 [7]
4370 [7]
4380 [7]
4390 [7]
4400 [7]
4410 [7]
4420 [7]
4430 [7]
4440 [7]
4450 [7]
4460 [7]
4470 [7]
4480 [7]
4490 [7]
4500 [7]
4510 [7]
4520 [7]
4530 [7]
4540 [7]
4550 [7]
4560 [7]
4570 [7]
4580 [7]
4590 [7]
4600 [7]
4610 [7]
4620 [7]
4630 [7]
4640 [7]
4650 [7]
4660 [7]
4670 [7]
4680 [7]
4690 [7]
4700 [7]
4710 [7]
4720 [7]
4730 [7]
4740 [7]
4750 [7]
4760 [7]
4770 [7]
4780 [7]
4790 [7]
4800 [7]
4810 [7]
4820 [7]
4830 [7]
4840 [7]
4850 [7]
4860 [7]
4870 [7]
4880 [7]
4890 [7]
4900 [7]
4910 [7]
4920 [7]
4930 [7]
4940 [7]
4950 [7]
4960 [7]
4970 [7]
4980 [7]
4990 [7]
5000 [7]
5010 [7]
5020 [7]
5030 [7]
5040 [7]
5050 [7]
5060 [7]
5070 [7]
5080 [7]
5090 [7]
5100 [7]
5110 [7]
5120 [7]
5130 [7]
5140 [7]
5150 [7]
5160 [7]
5170 [7]
5180 [7]
5190 [7]
5200 [7]
5210 [7]
5220 [7]
5230 [7]
5240 [7]
5250 [7]
5260 [7]
5270 [7]
5280 [7]
5290 [7]
5300 [7]
5310 [7]
5320 [7]
5330 [7]
5340 [7]
5350 [7]
5360 [7]
5370 [7]
5380 [7]
5390 [7]
5400 [7]
5410 [7]
5420 [7]
5430 [7]
5440 [7]
5450 [7]
5460 [7]
5470 [7]
5480 [7]
5490 [7]
5500 [7]
5510 [7]
5520 [7]
5530 [7]
5540 [7]
5550 [7]
5560 [7]
5570 [7]
5580 [7]
5590 [7]
5600 [7]
5610 [7]
5620 [7]
5630 [7]
5640 [7]
5650 [7]
5660 [7]
5670 [7]
5680 [7]
5690 [7]
5700 [7]
5710 [7]
5720 [7]
5730 [7]
5740 [7]
5750 [7]
5760 [7]
5770 [7]
5780 [7]
5790 [7]
5800 [7]
5810 [7]
5820 [7]
5830 [7]
5840 [7]
5850 [7]
5860 [7]
5870 [7]
5880 [7]
5890 [7]
5900 [7]
5910 [7]
5920 [7]
5930 [7]
5940 [7]
5950 [7]
5960 [7]
5970 [7]
5980 [7]
5990 [7]
6000 [7]
6010 [7]
6020 [7]
6030 [7]
6040 [7]
6050 [7]
6060 [7]
6070 [7]
6080 [7]
6090 [7]
6100 [7]
6110 [7]
6120 [7]
6130 [7]
6140 [7]
6150 [7]
6160 [7]
6170 [7]
6180 [7]
6190 [7]
6200 [7]
6210 [7]
6220 [7]
6230 [7]
6240 [7]
6250 [7]
6260 [7]
6270 [7]
6280 [7]
6290 [7]
6300 [7]
6310 [7]
6320 [7]
6330 [7]
6340 [7]
6350 [7]
6360 [7]
6370 [7]
6380 [7]
6390 [7]
6400 [7]
6410 [7]
6420 [7]
6430 [7]
6440 [7]
6450 [7]
6460 [7]
6470 [7]
6480 [7]
6490 [7]
6500 [7]
6510 [7]
6520 [7]
6530 [7]
6540 [7]
6550 [7]
6560 [7]
6570 [7]
6580 [7]
6590 [7]
6600 [7]
6610 [7]
6620 [7]
6630 [7]
6640 [7]
6650 [7]
6660 [7]
6670 [7]
6680 [7]
6690 [7]
6700 [7]
6710 [7]
6720 [7]
6730 [7]
6740 [7]
6750 [7]
6760 [7]
6770 [7]
6780 [7]
6790 [7]
6800 [7]
6810 [7]
6820 [7]
6830 [7]
6840 [7]
6850 [7]
6860 [7]
6870 [7]
6880 [7]
6890 [7]
6900 [7]
6910 [7]
6920 [7]
6930 [7]
6940 [7]
6950 [7]
6960 [7]
6970 [7]
6980 [7]
6990 [7]
7000 [7]
7010 [7]
7020 [7]
7030 [7]
7040 [7]
7050 [7]
7060 [7]
7070 [7]
7080 [7]
7090 [7]
7100 [7]
7110 [7]
7120 [7]
7130 [7]
7140 [7]
7150 [7]
7160 [7]
7170 [7]
7180 [7]
7190 [7]
7200 [7]
7210 [7]
7220 [7]
7230 [7]
7240 [7]
7250 [7]
7260 [7]
7270 [7]
7280 [7]
7290 [7]
7300 [7]
7310 [7]
7320 [7]
7330 [7]
7340 [7]
7350 [7]
7360 [7]
7370 [7]
7380 [7]
7390 [7]
7400 [7]
7410 [7]
7420 [7]
7430 [7]
7440 [7]
7450 [7]
7460 [7]
7470 [7]
7480 [7]
7490 [7]
7500 [7]
7510 [7]
7520 [7]
7530 [7]
7540 [7]
7550 [7]
7560 [7]
7570 [7]
7580 [7]
7590 [7]
7600 [7]
7610 [7]
7620 [7]
7630 [7]
7640 [7]
7650 [7]
7660 [7]
7670 [7]
7680 [7]
7690 [7]
7700 [7]
7710 [7]
7720 [7]
7730 [7]
7740 [7]
7750 [7]
7760 [7]
7770 [7]
7780 [7]
7790 [7]
7800 [7]
7810 [7]
7820 [7]
7830 [7]
7840 [7]
7850 [7]
7860 [7]
7870 [7]
7880 [7]
7890 [7]
7900 [7]
7910 [7]
7920 [7]
7930 [7]
7940 [7]
7950 [7]
7960 [7]
7970 [7]
7980 [7]
7990 [7]
8000 [7]
8010 [7]
8020 [7]
8030 [7]
8040 [7]
8050 [7]
8060 [7]
8070 [7]
8080 [7]
8090 [7]
8100 [7]
8110 [7]
8120 [7]
8130 [7]
8140 [7]
8150 [7]
8160 [7]
8170 [7]
8180 [7]
8190 [7]
8200 [7]
8210 [7]
8220 [7]
8230 [7]
8240 [7]
8250 [7]
8260 [7]
8270 [7]
8280 [7]
8290 [7]
8300 [7]
8310 [7]
8320 [7]
8330 [7]
8340 [7]
8350 [7]
8360 [7]
8370 [7]
8380 [7]
8390 [7]
8400 [7]
8410 [7]
8420 [7]
8430 [7]
8440 [7]
8450 [7]
8460 [7]
8470 [7]
8480 [7]
8490 [7]
8500 [7]
8510 [7]
8520 [7]
8530 [7]
8540 [7]
8550 [7]
8560 [7]
8570 [7]
8580 [7]
8590 [7]
8600 [7]
8610 [7]
8620 [7]
8630 [7]
8640 [7]
8650 [7]
8660 [7]
8670 [7]
8680 [7]
8690 [7]
8700 [7]
8710 [7]
8720 [7]
8730 [7]
8740 [7]
8750 [7]
8760 [7]
8770 [7]
8780 [7]
8790 [7]
8800 [7]
8810 [7]
8820 [7]
8830 [7]
8840 [7]
8850 [7]
8860 [7]
8870 [7]
8880 [7]
8890 [7]
8900 [7]
8910 [7]
8920 [7]
8930 [7]
8940 [7]
8950 [7]
8960 [7]
8970 [7]
8980 [7]
8990 [7]
9000 [7]
9010 [7]
9020 [7]
9030 [7]
9040 [7]
9050 [7]
9060 [7]
9070 [7]
9080 [7]
9090 [7]
9100 [7]
9110 [7]
9120 [7]
9130 [7]
9140 [7]
9150 [7]
9160 [7]
9170 [7]
9180 [7]
9190 [7]
9200 [7]
9210 [7]
9220 [7]
9230 [7]
9240 [7]
9250 [7]
9260 [7]
9270 [7]
9280 [7]
9290 [7]
9300 [7]
9310 [7]
9320 [7]
9330 [7]
9340 [7]
9350 [7]
9360 [7]
9370 [7]
9380 [7]
9390 [7]
9400 [7]
9410 [7]
9420 [7]
9430 [7]
9440 [7]
9450 [7]
9460 [7]
9470 [7]
9480 [7]
9490 [7]
9500 [7]
9510 [7]
9520 [7]
9530 [7]
9540 [7]
9550 [7]
9560 [7]
9570 [7]
9580 [7]
9590 [7]
9600 [7]
9610 [7]
9620 [7]
9630 [7]
9640 [7]
9650 [7]
9660 [7]
9670 [7]
9680 [7]
9690 [7]
9700 [7]
9710 [7]
9720 [7]
9730 [7]
9740 [7]
9750 [7]
9760 [7]
9770 [7]
9780 [7]
9790 [7]
9800 [7]
9810 [7]
9820 [7]
9830 [7]
9840 [7]
9850 [7]
9860 [7]
9870 [7]
9880 [7]
9890 [7]
9900 [7]
9910 [7]
9920 [7]
9930 [7]
9940 [7]
9950 [7]
9960 [7]
9970 [7]
9980 [7]
9990 [7]
10000 [7]
10010 [7]
10020 [7]
10030 [7]
10040 [7]
10050 [7]
10060 [7]
10070 [7]
10080 [7]
10090 [7]
10100 [7]
10110 [7]
10120 [7]
10130 [7]
10140 [7]
10150 [7]
10160 [7]
10170 [7]
10180 [7]
10190 [7]
10200 [7]
10210 [7]
10220 [7]
10230 [7]
10240 [7]
10250 [7]
10260 [7]
10270 [7]
10280 [7]
10290 [7]
10300 [7]
10310 [7]
10320 [7]
10330 [7]
10340 [7]
10350 [7]
10360 [7]
10370 [7]
10380 [7]
10390 [7]
10400 [7]
10410 [7]
10420 [7]
10430 [7]
10440 [7]
10450 [7]
10460 [7]
10470 [7]
10480 [7]
10490 [7]
10500 [7]
10510 [7]
10520 [7]
10530 [7]
10540 [7]
10550 [7]
10560 [7]
10570 [7]
10580 [7]
10590 [7]
10600 [7]
10610 [7]
10620 [7]
10630 [7]
10640 [7]
10650 [7]
10660 [7]
10670 [7]
10680 [7]
10690 [7]
10700 [7]
10710 [7]
10720 [7]
10730 [7]
10740 [7]
10750 [7]
10760 [7]
10770 [7]
10780 [7]
10790 [7]
10800 [7]
10810 [7]
10820 [7]
10830 [7]
10840 [7]
10850 [7]
10860 [7]
10870 [7]
10880 [7]
10890 [7]
10900 [7]
10910 [7]
10920 [7]
10930 [7]
10940 [7]
10950 [7]
10960 [7]
10970 [7]
10980 [7]
10990 [7]
11000 [7]
11010 [7]
11020 [7]
11030 [7]
11040 [7]
11050 [7]
11060 [7]
11070 [7]
11080 [7]
11090 [7]
11100 [7]
11110 [7]
11120 [7]
11130 [7]
11140 [7]
11150 [7]
11160 [7]
11170 [7]
11180 [7]
11190 [7]
11200 [7]
11210 [7]
11220 [7]
11230 [7]
11240 [7]
11250 [7]
11260 [7]
11270 [7]
11280 [7]
11290 [7]
11300 [7]
11310 [7]
11320 [7]
11330 [7]
11340 [7]
11350 [7]
11360 [7]
11370 [7]
11380 [7]
11390 [7]
11400 [7]
11410 [7]
11420 [7]
11430 [7]
11440 [7]
11450 [7]
11460 [7]
11470 [7]
11480 [7]
11490 [7]
11500 [7]
11510 [7]
11520 [7]
11530 [7]
11540 [7]
11550 [7]
11560 [7]
11570 [7]
11580 [7]
11590 [7]
11600 [7]
11610 [7]
11620 [7]
11630 [7]
11640 [7]
11650 [7]
11660 [7]
11670 [7]
11680 [7]
11690 [7]
11700 [7]
11710 [7]
11720 [7]
11730 [7]
11740 [7]
11750 [7]
11760 [7]
11770 [7]
11780 [7]
11790 [7]
11800 [7]
11810 [7]
11820 [7]
11830 [7]
11840 [7]
11850 [7]
11860 [7]
11870 [7]
11880 [7]
11890 [7]
11900 [7]
11910 [7]
11920 [7]
11930 [7]
11940 [7]
11950 [7]
11960 [7]
11970 [7]
11980 [7]
11990 [7]
12000 [7]
12010 [7]
12020 [7]
12030 [7]
12040 [7]
12050 [7]
12060 [7]
12070 [7]
12080 [7]
12090 [7]
12100 [7]
12110 [7]
12120 [7]
12130 [7]
12140 [7]
12150 [7]
12160 [7]
12170 [7]
12180 [7]
12190 [7]
12200 [7]
12210 [7]
12220 [7]
12230 [7]
12240 [7]
12250 [7]
12260 [7]
12270 [7]
12280 [7]
12290 [7]
12300 [7]
12310 [7]
12320 [7]
12330 [7]
12340 [7]
12350 [7]
12360 [7]
12370 [7]
12380 [7]
12390 [7]
12400 [7]
12410 [7]
12420 [7]
12430 [7]
12440 [7]
12450 [7]
12460 [7]
12470 [7]
12480 [7]
12490 [7]
12500 [7]
12510 [7]
12520 [7]
12530 [7]
12540 [7]
12550 [7]
12560 [7]
12570 [7]
12580 [7]
12590 [7]
12600 [7]
12610 [7]
12620 [7]
12630 [7]
12640 [7]
12650 [7]
12660 [7]
12670 [7]
12680 [7]
12690 [7]
12700 [7]
12710 [7]
12720 [7]
12730 [7]
12740 [7]
12750 [7]

ちょっと変わった演算子 sizeof

- ある型があったとき、その型の宣言に必要なバイト数を教えてくれる
- ある変数があったとき、その変数を使用して
いるバイト数を教えてくれる

```
struct ic2LINEX x;
float f;
printf("size of float is %d\n", sizeof(float));
printf("size of ic2LINEX is %d\n", sizeof(struct ic2LINEX));
printf("size of x is %d\n", sizeof(x));
```

表示結果は、上から順に、4, 40, 40
sizeof演算子へは、伝統的に引数を()で括って渡す

Linked List (スクリプト読込プログラムを例として)

目的
事前に予想できない量のデータをメモリ上に
動的に確保できるようにする
ポインタと構造体を駆使
Linked Listの利用

さて準備は整った。

データ構造を使って表現したいもの

- 線分
- パッチ
- 物体

問題はスクリプトファイルを開いてみるまで、

- 1つの物体に
 - 何本線分があるかわからない
 - 何枚パッチがあるかわからない
- 物体が何個入っているかわからない
- どれだけアニメーションが続くかわからない

個数不明なデータに対する処理

Cプログラミング上の問題

- 配列では無理
 - C言語ではプログラム記述時に配列の要素数を指定
(配列の数はコンパイル時に固定しなくてはならない)

```
float ff[1000];
struct ic2LINE ll[100];
```

固定値-今回のような問題設定では固定値の指定不能

「十分に余裕をとった数値を設定」というのも1つの考え方だが、
今回はよろしくない(100万個データが来るかもしれない)

動的メモリ確保

- C言語における可変データ数に対する解決策
- データが増えるたびに必要なメモリをOSから貰って
くる
 - 貰ってくるための関数(malloc, calloc)を利用すること

```
#include <stdlib.h>
void *malloc(size_t NBYTES);
void *calloc(size_t N, size_t S);
```

calloc関数

calloc = cleared memory allocationという噂です

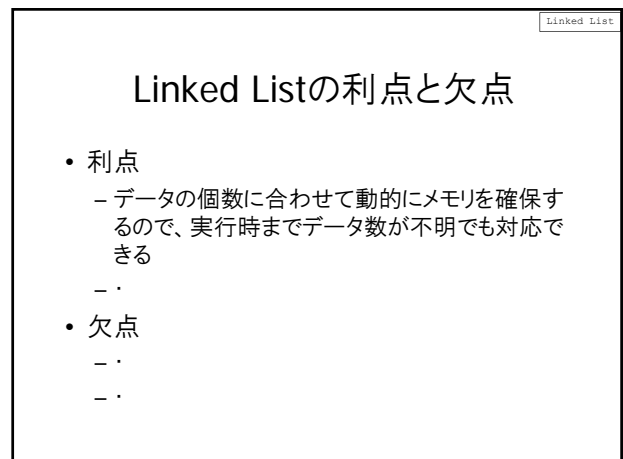
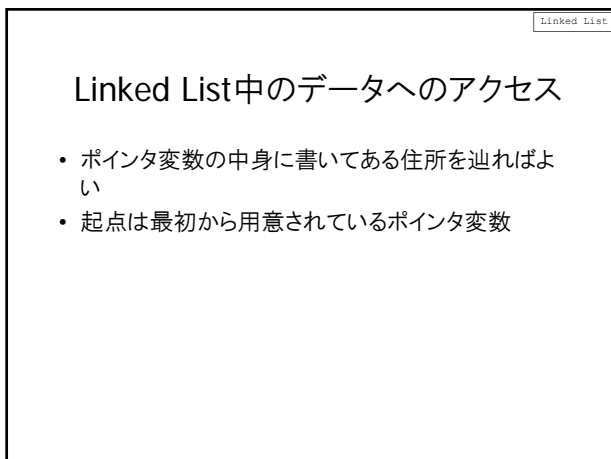
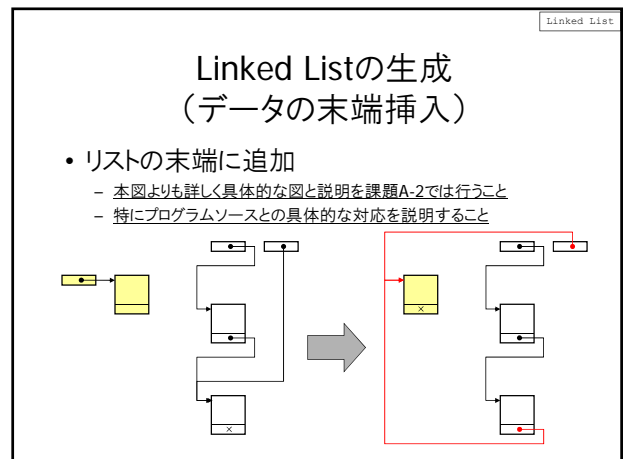
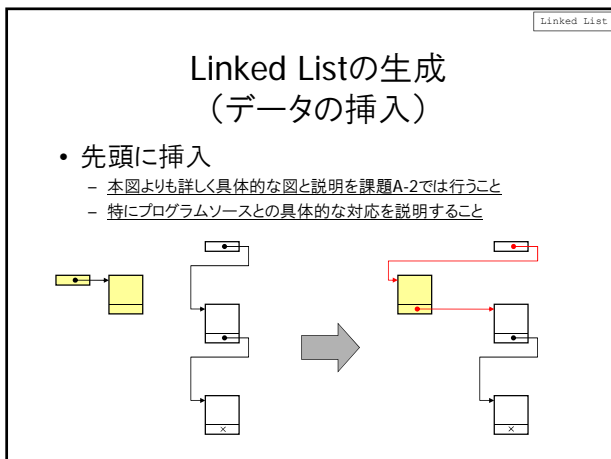
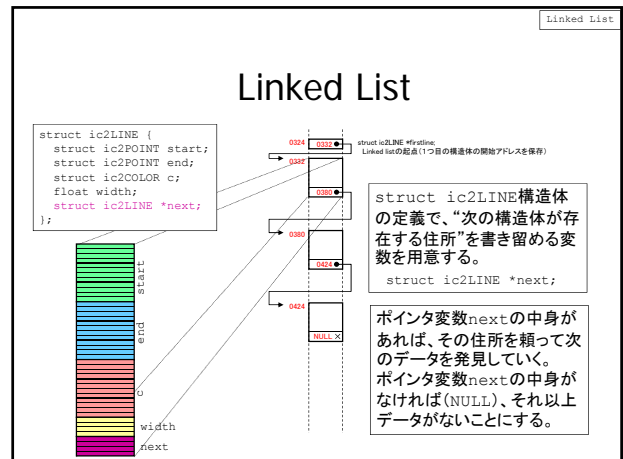
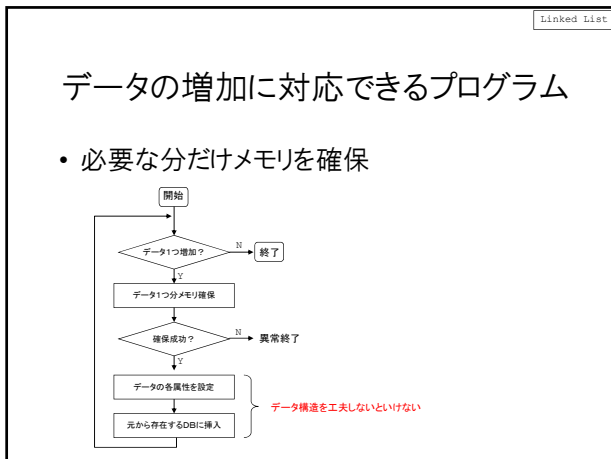
- Sバイトのメモリブロックを連続N個分確保
- 先頭のアドレスを返してくれる
 - 確保に失敗した場合はNULLを返してくる
- 確保されたメモリ空間は全て0で初期化済
 - malloc()では初期化してくれない

```
#include <stdlib.h>
```

```
struct ic2LINEX *ptr = NULL;
ptr = (struct ic2LINEX *)calloc(1, sizeof(struct ic2LINEX));
if (ptr == NULL) return; // メモリ確保に失敗 [1]
ptr->start.x = 2.0;
ptr->end.x = 4.0;
```

[1] struct ic2LINEXを1つ分(40バイトが1つで合計40バイト)確保してもらい、その先頭番地をcalloc()関数が返す。

[2] calloc()が返す番地がどういふ変数(struct ic2LINEX)のための番地なのかを明示するためのcast

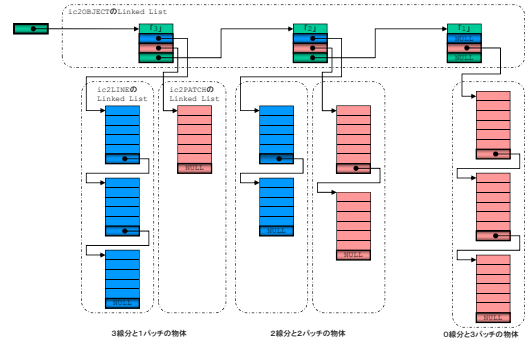


スクリプト読込プログラム

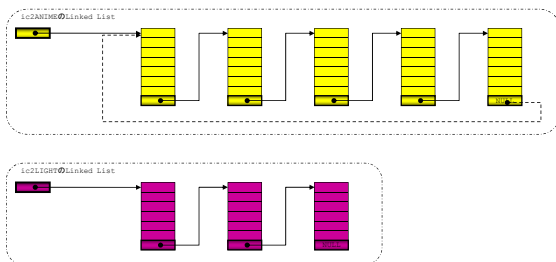
- スクリプトファイル中の全ての記述をメモリ上に保存
- データ構造の設計
 - Linked Listを多用(可変データ数に対応)
- 物体
 - 線分
 - 三角形パッチ
- アニメーション
- 光源

次のスライドからshortscript039a.txtのメモリイメージを見てみよう

データ構造の概観I(物体)



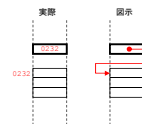
データ構造の概観II(他)



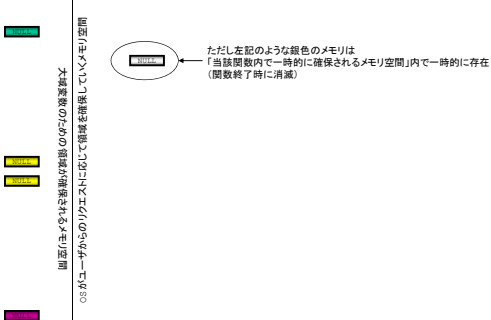
データ構造の構築の様子

- プログラムの実行の様子をメモリイメージで考えてみる

ポインタ変数(アドレスが格納される)



三種類のメモリ空間



データ構造の構築[01]

大域変数

load_ic2OBJECT(), 1回目

グローバル変数

静的変数

動的変数

動的変数

動的変数

動的変数

動的変数

動的変数

動的変数

動的変数

動的変数

動的変数

動的変数

動的変数

動的変数

動的変数

動的変数

動的変数

動的変数

動的変数

動的変数

