

情報工学演習III

ポインタと文字列の復習

(第2回)

渡辺宏太郎 伊達 央

補助 リュ・ビョンジュン 岡田 美里

コンピュータは何らかのデータをメモリ上に置くことで処理を行います。例えば、CASLでレジスタ内のデータをメモリ上に格納するときは

ST命令 レジスタ内のデータをメモリーへ書き込む命令

ラベル	ST	GRx,アドレス[,GRx]
-----	----	----------------

などとするのでした。覚えているかな？しかし、あつかうデータが大量になると1000番地とか2000番地といった生のアドレスを覚えることは不便となります。そこで、

—変数 aは1000番地のことを表すようにしよう—

というルールが考えられました。

アドレス(番地)		
a ←	1000	'w'
	1001	
	1002	

例えば
char a = 'w'; (char a = 119と同じ)
とすると、メモリ上のどこか(例えば1000番地)にchar型だから1バイトの領域が確保され、値119が格納されます。

この変数aの番地を知るには(上の例だったら1000番地)

&a

としました。

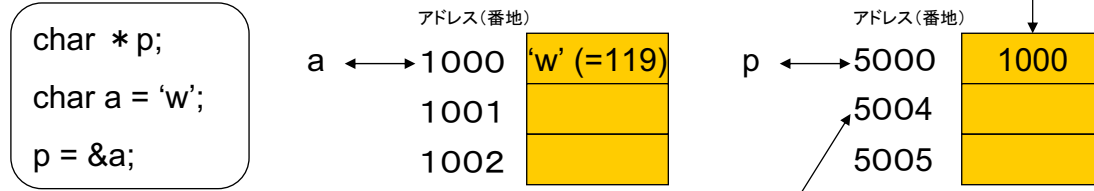
ポインタ

ポインタ(変数)もアドレスが与えられて、そこにデータが格納されるという意味では通常のchar aなどの変数と同じです。違う点は

データとしてアドレスが格納される

aのアドレス1000が格納されます。

という点です。



ポインタpに対して *pとするとpに格納されているアドレスの内容(この場合だったら119)となります。それだけです。

問19 適当にmain関数を補って以下のプログラムを実行してみよ。(検印はありませんがやってみて下さい)

A

```
int x, *y;
y = &x;
x = 10;
printf("contents of x, y = %d, %d\n", x, *y);
```

32ビットアドレッシングだとすると4飛ぶ

B

```
int x, *y;
y = &x;
*y = 10;
printf("contents of x, y = %d %d\n", x, *y);
```

C

```
int x, *y, *z;
y = &x;
z = y;
x = 10; (*y = 10; または *z = 10; でもよい)
printf("contents of x, y, z = %d %d %d\n", x, *y, *z);
printf("address of x, y, z = %p %p %p\n", &x, y, z); (yとzはyとzに入っているアドレスという意味)
```

20の確認印

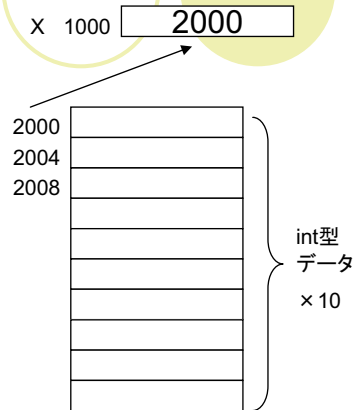
問20 Bでyに格納されている値とxのアドレスが等しいことを確かめよ。また、y自身のアドレスを表示せよ。

問21 以下のプログラムを実行してみよ。(検印はありませんがやってみて下さい)

```
int x, *y, **z;
z = &y;
y = &x;
x = 10; (または *y = 10; または **z = 10; どれでもよい)
printf("contents of x, y, z = %d, %d, %d\n", x, *y, **z);
```

問22 以下のプログラムを実行してみよ。結果はどうなるか？

- A
- ```
int X[10], i;
for(i = 0; i < 10; i++) X[i] = i;
for(i = 0; i < 10; i++) printf("%d", X[i]);
```
- B
- ```
int *X, i;
X = (int *)malloc(10 * sizeof(int));
for(i = 0; i < 10; i++) X[i] = i;
for(i = 0; i < 10; i++) printf("%d", X[i]);
```
- C
- ```
int X[10], i;
for(i = 0; i < 10; i++) *(X+i) = i;
for(i = 0; i < 10; i++) printf("%d", *(X+i));
```
- D
- ```
int *X, i;
X = (int *)malloc(10 * sizeof(int));
for(i = 0; i < 10; i++) *(X+i) = i;
for(i = 0; i < 10; i++) printf("%d", *(X+i));
```
- E
- ```
int *X, i;
X = (int *)malloc(10 * sizeof(int));
for(i = 0; i < 10; i++) printf("%p", X+i); (表示が4バイト飛ばしになっている意味がわかるか?)
```



void \*malloc(size\_t size)はsizeバイトのメモリを確保し、void型へのポインタを返す。わかりにくければ、確保したメモリ上の領域の先頭アドレスを返すと思ってもよい。

確認印

問23 以下のプログラムを実行してみよ。結果はどうなるか？その結果を参考にし  
てvc[i], i[vc], \*(p+i), \*(i+p)の値が全て一致することを確かめよ。

```
int X[10], i;
for(i = 0; i < 10; i++) i[X] = i;
for(i = 0; i < 10; i++) printf("%d", i[X]);
```

確認印

# 関数に配列を渡す方法

```
ex24.c - emacs@csimc13.cs.nda.ac.jp
File Edit Options Buffers Tools C Help

#include <stdio.h>

void f(int x);
int main(void){
 int x;
 x = 10;
 printf("address of x is %X\n",&x);

 f(x);
 printf("value of x is %d\n",x);
}

void f(int x){
 x = 15;
 printf("address of x is %X\n",&x);
}

-u:-- ex24.c (C Abbrev)--L15--A
Wrote /home/wata/HOME_DIR/ensyuu3/ex24.c
sc
```

問24 以下のプログラムを実行し、**main内でのxのアドレスと関数f内でのxのアドレスが異なる**

ことを確かめよ。

また、関数fを実行前と実行後でxの値が変わらないことを確かめよ。

(簡単なので確認印はありません)。

C言語の常識だが、4年になっても間違えて覚えている人が**必ずいる**。

漫然と打ち込むのではなく、**勉強しているという意識を高めよう**。

問25 問24で関数fをvoid f(int \*x)に変更し、main内でf(&x)としてxのアドレスを渡すように変更したとき、xの値が15に書き換わることを確かめよ。また、関数f内でポインタxの指すアドレスがmain内のxのアドレスと同じであることを確かめよ。さらに、関数f内でscanfによってポインタxの指すアドレスに15を代入するように変更してみよ。

```
ex25.c - emacs@csimc13.cs.nda.ac.jp
File Edit Options Buffers Tools C Help

#include <stdio.h>

void f(int *x);
int main(void){
 int x;
 printf("x=\n");
 scanf("%d",&x);
 printf("address of x is %X\n",&x);

 f(&x);
 printf("value of x is %d\n",x);
}

void f(int *x){
 *x = 15;
 /* ポインタの指しているアドレスが関数f内のxの
 アドレスと一致していることを確かめよ。
 printf("Pointing address is %X\n",&x);
 いいのか? */
}

-Aあ: ** ex25.c (C Abbrev)--L19--All--
Auto-saving...done
```

```
ex25.c - emacs@csimc13.cs.nda.ac.jp
File Edit Options Buffers Tools C Help

#include <stdio.h>

void f(int *x);
int main(void){
 int x;
 printf("x=\n");
 scanf("%d",&x);
 printf("address of x is %X\n",&x);

 f(&x);
 printf("value of x is %d\n",x);
}

void f(int *x){
 printf("x=\n");
 /* scanfでmain内のxに15を代入してみよ */
 /* ポインタの指しているアドレスが関数f内のxの
 アドレスと一致していることを確かめよ。
 printf("Pointing address is %X\n",&x);
 いいのか? */
}

-u: ** ex25.c (C Abbrev)
```

確認印

「関数に配列を渡すには配列の先頭アドレスを渡せばよい」ということを覚えていますか？例えば、fにint型の大きさ10の配列を渡すときf(&x[0])またはf(x)とすればアドレスが渡せます。一方、受け取る関数fは

→f(int x[10])

f(int x[])

f(int \*x)

のいずれの型で受け取るように記述してもOKです(配列の先頭アドレスが受け取ればよいので)。

配列の先頭アドレスさえ渡せばよいので配列の大きさには意味が無い

```
#include <stdio.h>

void f(int *x);
int main(void) {
 int i;
 int x[10];

 /* fに配列xを渡す */
 for(i=0; i<10; i++) {
 printf("x[%d]=%d\n", i, x[i]);
 }

 void f(int *x) {
 int i;
 /* 適当な処理を書く */
 }
}
```

問26 左のプログラムで関数f内でmain内の配列x[0]~x[9]に0~9を代入するようにしてみよ。

確認印

問27 2つのint型整数の値を交換する関数

void swap(int \*x, int \*y)

を作成し、それを利用して2つのint型整数を小さい順に並べる関数

void sort2(int \*n1, int \*n2)

を作成せよ。

```
ex27.c - emacs@localhost.localdomain
File Edit Options Buffers Tools C Help

#include <stdio.h>

void swap(int *x, int *y);
void sort2(int *n1, int *n2);

int main(void) {
 int na, nb;

 printf("2つの整数を入力\n");
 printf("整数A:");
 scanf("%d", &na);
 printf("整数B:");
 scanf("%d", &nb);

 printf("na = %d nb = %d\n", na, nb);
 sort2(&na, &nb);
 printf("na = %d nb = %d\n", na, nb);
}

void swap(int *x, int *y) {
 /* 適当に埋めよ */
}

void sort2(int *n1, int *n2) {
 /* swap(&n1, &n2); いいのか? */
}

--u:-- ex27.c (C Abbrev) --L30--A11-----
```

Swap(&n1,&n2)は間違っている。何故か？

確認印

問28 ファイルから1文字ずつ読み込んで数字文字の出現回数を受け取った配列に記憶する関数void count\_n(int \*cnt)を作成せよ. 数字の文字出現数は\*を並べたグラフで表示する. ファイルは「情報工学演習III」のページのdata.txtを利用すること. 結果が以下のようになればOKです.

```
#include <stdio.h>

void count_n(int *cnt);

int main(void) {
 int i, j;
 int cnt[10] = {0};

 count_n(&cnt[0]);

 for(i=0; i<10; i++) {
 printf("%d: ", i);
 for(j=0; j<cnt[i]; j++) {
 printf("*");
 }
 printf("\n");
 }
}

void count_n(int *cnt) {
 char ch;

 while(1) {
 ch = getchar();
 if(/*もしchがファイルエンドを示していたら*/){
 /*whileを抜けるようにする*/;
 }

 if(/*chが'0'と'9'の間にあったら*/){
 /* ch='0'だったらcnt[0]をインクリメント*/
 }
 }
}
```

ファイルから1文字読み取り  
chに代入します.

## 結果

```
[wata@csimc16 ensyuu3]$./a.out < data.txt
0: *****
1: ****
2: *****
3: *****
4: ****
5: **
6: **
7: **
8: *****
9: *****
[wata@csimc16 ensyuu3]$
```

a.out < data.txt

とするとdata.txtの内容を1文字ずつコンソールで打ち込んだことと同じことになります.

(Windowsではできません. LINUX系のみ)

確認印

問29 文字列を後ろから逆に表示する関数(ABCと受け取ったらCBAと表示する関数void r\_string(char \*str)を作成せよ. 文字列の長さは関数内で判定するものとする.

A, B, C 改行

と入力したら配列string[100]は

|     |     |     |   |  |
|-----|-----|-----|---|--|
| 'A' | 'B' | 'C' | 0 |  |
|-----|-----|-----|---|--|

となっています.

```
#include <stdio.h>

void rstring(char *str);

int main(void) {
 char string[100];

 printf("Input string:\n");
 scanf("%s", string);

 printf("Reverse is\n");
 rstring(&string[0]);
 return(0);
}

void rstring(char *str) {
 char *front;
 front = str;

 /* whileを使って文字列の終わりを確定 */
 /* 適当な処理を書く */

 printf("\n");
}
```

確認印

問30 文字列string内の全ての数字文字以外の文字を削除する関数  
void del\_asc(char \*str)を作成せよ.

```
#include <stdio.h>

void del_asc(char *str);

int main(void){
 char string[100];

 printf("Input string: ");
 scanf("%s", string);

 del_asc(&string[0]);
 printf("%s\n", string);
 return(0);
}

void del_asc(char *str){
 char *ptr;
 ptr = str;

 /* *strが0でない間whileでループ
 *strが0〜9だったら*ptrに*strの値を代入
 */
 /* *ptrの最後は0にしておく */
}
```

確認印

問31 ~wata/Banner.pl>data2.txtを実行し, 適当な文字列を打ち込んでみよ.  
(~wata/Banner.plの実行結果がdata2.txtに書き込まれる. Windowsではできません)次ページのように空白の連続した個数, \*の連続した個数, 改行を数える関数  
void arttoa(char \*str, int \*arr)を完成せよ.

```
#include <stdio.h>

void arttoa(char *str, int *arr);

int main(void){
 char st[1000];
 int a[1000]={0};
 char ch;
 int count=0, i=0;

 while(1){
 ch = getchar();
 count++;
 if(ch == EOF || count>1000){
 st[i] = EOF;
 break;
 }
 st[i] = ch;
 i++;
 }

 i=0;
 arttoa(&st[0], &a[0]);
 i = 0;
 while(a[i] != -2){
 printf("%d ", a[i]);
 i++;
 }
}
```

~wata/Banner.plの実行例

```
[wata@csimc16 ensyuu3]$./Banner.pl > data2.txt
watanabe
[wata@csimc16 ensyuu3]$ more data2.txt

* * * * *
* * * * *
* * * * *
* * * * *
* * * * *

[wata@csimc16 ensyuu3]$
```

プログラムの実行は a.out < data2.txt  
で問28と同じです.

```

void arttoa(char *str, int *arr){
 int len=0,i=0;
 char state = ' ';
 while(*str!=EOF){
 if(*str == '\n'){ /* 改行が来たときの処理 */
 arr[i] = len; /* 改行が来る直前の文字列の長さを格納しておかないと */
 arr[i+1] = -1; /* 改行記号の書き込み */
 len = 0; /* 連続した文字列の長さをリセット */
 str++;
 i=i+2;
 state = ' '; /* 改行の次は空白から始まると仮定. *が来てもOK */
 }else{
 if(*str == state){
 /* 適当な処理を書け */
 /* ヒント lenとstrをどうにかする */
 }else{
 /* 次に来る文字が直前の文字と異なる場合だからarr[i]に書き込み */
 len /*をどうするのか */
 i /*をどうするのか */
 if(state==' '){
 /* stateをどう変えるのか */
 }else{
 /* stateをどう変えるのか */
 }
 }
 }
 }
 arr[i] = -2; /* EOFは-2とする */
}
(データの末端)

```

実行例

```

[wata@csimc16 ensyuu3]$ more data2.txt
* *
* *
* *
* *
** **
** **
* *

[wata@csimc16 ensyuu3]$./a.out < data2.txt
8 -1 1 1 4 1 1 -1 1 1 4 1 1 -1 1 1 4 1 1 -1 1 1 2 1 1 1 -1 1 2 2 2 1 -1 1 1 4
1 1 -1 1 -1 [wata@csimc16 ensyuu3]$

```

関数arttoaは情報理論でやったランレングス符号化に似た符号化です。

思い出せただろうか？

確認印

## レポート問題

- 問32 2つのint型整数の値を交換する関数  
void swap(int \*x, int \*y)  
を作成し、それを利用して3個のint型整数を小さい順に並べる関数  
void sort3(int \*n1, int \*n2, int \*n3)  
を作成せよ。
- 問33 2つのint型整数の値を交換する関数  
void swap(int \*x, int \*y)  
を作成し、それを利用してN個のint型整数を小さい順に並べる関数  
void sortN(int \*n)  
を作成せよ。Nはマクロで定義するものとする。
- 問34 ファイルから1文字ずつ読み込んで英大文字または英小文字の出現回数を受け取った配列に記憶する関数void countc\_n(int \*cnt)を作成せよ。ただし大文字のAと小文字のaなどの区別はしない。文字出現数は\*を並べたグラフで表示する。ファイルは「情報工学演習Ⅲ」のページのdata.txtを利用すること。
- 問35 文字列string内の全ての数字文字を削除する関数  
void del\_digit(char \*str)を作成せよ。
- 問36 文字列を整数列に変換する関数  
int \*atoi(char \*str)を作成せよ。



# レポート問題

- 問37 整数の配列intgrsを文字列に変換する関数 `char *itoa(int *intgrs)`を作成せよ.
- 問38 問31で符号化した文字列を復号化する関数 `void atoart(char *str, int *arr)`を作成せよ.

