・引数としてのポインター

ここでは、ある２つの値を入れ替えるswapの例を用いて、引数としてのポインターを説明する。プログラミングにおける、代表的なアルゴリズムである、ソートを行う時に、２つの配列の要素を入れ替えるという作業を行う必要がある。この時に、この作業を関数化せずにmainの中で行う場合には、tmpというもう一つの変数を用意し、

tmp = a[x];

a[x] = a[y]

a[y] = tmp;

とすれば良い。

しかし、この作業を引数としてa[x]とa[y]を与える様な関数にした場合、この関数を実行しても、main()の中では値が変化しない。

例

void swap(int a, int b)

{

int tmp;

tmp = a;

a = b;

b = tmp;

}

int main()

{

int a[2] = {10, 20};

swap(a[0],b[1]);

printf(“a[0] = %d”, a[0]);

printf(“a[1] = %d”, a[1]);

return 0;

}

これは、関数に引数として変数が与えられた時に、関数はその変数をコピーし、そのコピーに対して処理を加えるからである。

では、関数で行った処理をmain()内の変数に反映させる為にはどうすれば良いか？これはポインターを用いることで解決できる。ポインター(1)でポインターを用いることで、アドレスに格納されている値を、直接操作できることを説明した。よって、引数として変数のアドレスを与え、関数内部ではポインターを用いて処理を行うことで、関数で行った操作をmain()内の変数に反映させることができる。これを参照渡しという。これに対して、引数そのものを与えることを、値渡しという。

例

void swap(int\* a, int\* b)

{

int tmp;

tmp = \*a;

\*a = \*b;

\*b = tmp;

}

int main()

{

int a[2] = {10, 20};

swap(&a[0], &a[1]);

printf(“a[0] = %d”, a[0]);

printf(“a[1] = %d”, a[1]);

return 0;

}

・配列とポインター

配列とポインターには密接な関係がある。配列は宣言された時に、メモリに要素が連続して確保される。またa[10]という配列を宣言した場合、aは配列の先頭番地を表す。よってa+1は&a[1]を表し、a+9は&a[9]を表す。アドレスから値を操作するには\*という演算子を用いればよいので、\*(a+1)とすればa[1]の値を得られる。ポインターを介して、配列の要素にアクセスするためには、配列の先頭番地をポインターに代入し、配列の要素を[x]と指定する。

例

int main()

{

int a[2] = {10, 20};

int\* p = a;

printf(“%d”, p[1]);

return 0;

}

・ポインターと文字列

C言語には文字列型というものはなく、文字列を扱うためにはchar型の配列を用いるか、char型のポインターを用いる必要がある。

例

int main()

{

char\* s = “abcdef”;

printf(“%s\n”, s);

return 0;  
}

　このように文字列を扱った場合、文字列は配列の様にメモリに連続して格納される。そしてsにはその先頭番地が与えられる。すなわち、s = s+1などのような処理を行った後に、文字列を出力すると、出力される文字列が変化する。