・リストとは

リストとは、自身の要素に自身の型のポインターを持つ構造体を用いたデータ構造である。この、要素に自身の型のポインターを持つ構造体を自己参照構造体という。

例：自己参照構造体

struct list {

int data;

struct list\* next:

}

リスト構造は、複数のノードを持ち、それらはポインターによって連結されている。具体的なリスト構造の実装方法は、要素のポインターに、自身の次のノード（節）の構造体のアドレスを代入することである。こうすることに依って、複数のノードをポインターによって連結することができる。

・ポインターの利点

今までは、複数のデータをまとめて扱うときには、配列を用いていた。しかし、配列にデータの追加や削除といった操作を行う場合、各要素を1個づつずらしていくといった手間が発生する。これは配列が大きくなればなるほど、効率が悪くなることを意味している。しかし、リスト構造はポインターによって連結されているだけであるので、データの追加や削除が、ポインターの付け替えだけで実現できるので、大変効率が良い。これがリスト構造の利点である。

・ポインターの欠点

ポインターの欠点は、各ノードは自身の次にあるノードが何か、という情報しか持っていないので、何番目のデータを取り出すといった操作をする場合には、先頭ノードから辿っていく必要がある。これは大変効率が悪く、これがリスト構造の欠点である。

・リストへのノードの挿入

リストへのノードの挿入はポインターの付け替えによって行われる。具体的には、挿入したい場所の一つ前のノードのポインターに、挿入するノードのアドレスを代入し、挿入するノードのポインターには、挿入したい場所の一つ後のノードのアドレスを代入することで実現できる。

・リストのノードの削除

　リストのノードの削除は、挿入と同様にポインターの付け替えによって行われる。具体的には、削除したいノードの一つ前のノードのポインターに、削除したいノードの一つ後のノードのアドレスを代入することで実現できる。