1　関数プロトタイプ

1.1　学習のポイント

　関数の型、引数の型を明確にするための関数プロトタイプについて学びます。
関数の型は、特に宣言しなければ、int 型とみなされます。実数型や、ポインタ型などの int 型以外の値を返す関数は、その関数を使用する (呼び出す) 前に型宣言しなければなりません。
また、ANSI C ではその際に関数の引数の型についても宣言するように勧めています。
このように、関数の使用に先立って、関数の型と引数の型を宣言したものを関数プロトタイプと呼びます。

```
double abs(double); /*関数プロトタイプ宣言*/

void main(void)
{
    .
    .
    a = abs(10);
    .
    .
}

double abs(double x)
{
    .
    .
}
```

関数プロトタイプ宣言がなされている場合は、もし関数呼び出し時に引数の型に誤りがあるとコンパイラがエラーメッセージを出してくれます。
C 言語」(河西朝雄著　ナツメ社)116 頁

1.2 例題 28

```
/*
  reidai28.c
  与えられた文字列を反転し、それへのポインタを返す関数 reverse() を作りなさい。
  C 言語 118 頁
*/

#include <stdio.h>
#include <string.h>

char *reverse(char a[])
{
  static char buff[80];
  int         i;
  int         n;

  n = strlen(a);
  for(i = 0; i < n; i++){
    buff[i] = a[n - i - 1];
  }
  buff[n] = '\n';

  return(buff);
}

int main()
{
  char str[80];

  printf("moji ? ");
  scanf("%s", str);
  printf("%s\n", reverse(str));

  return 0;
}
```

「C 言語」(河西朝雄著　ナツメ社)118 頁

2　AI による大規模データ処理入門

2.1　力ずくの探索手法

　探索技術の基礎となるのは、探索対象となる状態空間をもれなく順に探索していく力ずくの探索手法です。ここでは、探索とは何かを概観した上で、力ずくの探索手法である横型探索と縦型探索のアルゴリズムを紹介します。
AI による大規模データ処理入門　24 頁　小高　知宏著　オーム社

2.2　縦型探索プログラム

```
/*
  縦型探索プログラム
  depth.c
*/

#include <stdio.h>

#define LIMITL 256
#define TRUE 1
#define START 1
#define GOAL 999

struct node{
  int nodeid;
  int parentid;
};

struct node openlist[LIMITL];
int         openlistep = 0;
struct node closedlist[LIMITL];
int         closedlistep = 0;

void initlist();
```

```
void printlist();
void expand(int id);
void removefirst();
int  check(int id);
void printroute(int id);
void checkopenlist(int id);
void addopenlist();

int main()
{
  struct node currentnode;

  initlist();
  printlist();

  while(TRUE){
    if(openlistep == 0){
      printf("ゴールは見つかりませんでした\n");
      break;
    }

    if(openlist[0].nodeid == GOAL){
      printf("\n ゴールを見つけました\n");
      printf("%d[%d]", openlist[0].nodeid,
                       openlist[0].parentid);
      printroute(openlist[0].parentid);
      break;
    }

    currentnode = openlist[0];
    removefirst();

    expand(currentnode.nodeid);

    closedlist[closedlistep++] = currentnode;

    printlist();
  }

  return 0;
```

```
}

void printroute(int id)
{
  int i;

  for(i = 0; i < closedlistep; ++i){
    if(closedlist[i].nodeid == id){
      printf("<-%d[%d]", closedlist[i].nodeid,
                         closedlist[i].parentid);
      break;
    }
  }

  if(closedlist[i].parentid != 0){
    printroute(closedlist[i].parentid);
  }

  printf("\n");
}

int check(int id)
{
  int i;
  int res = 0;

  for(i = 0; i < openlistep; ++i){
    if(openlist[i].nodeid == id){
      res = TRUE;
    }
  }

  for(i = 0; i < closedlistep; ++i){
    if(closedlist[i].nodeid == id){
      res = TRUE;
    }
  }

  return res;
}
```

```
void checkopenlist(int id)
{
  int i,j;

  for(i = 1; i < openlistep; ++i){
    if(openlist[i].nodeid == id){
      for(j = i; j < openlistep; ++j){
        openlist[j] = openlist[j + 1];
      }
      --openlistep;
      --i;
    }
  }
}

void removefirst()
{
  int i;
  for(i = 0; i < openlistep; ++i){
    openlist[i] = openlist[i + 1];
  }
  --openlistep;
}

void expand(int id)
{
  int tree[][5] = {
    {1, 2, 3, 5},
    {2, 4},
    {3, 6, 7},
    {5, 4, 8},
    {6, 5, 7},
    {8, 7, 999},
    {0}
  };

  int i = 0;
  int j;
```

```
  while(tree[i][0] != 0){
    if(tree[i][0] == id){
      for(j = 1; tree[i][j] != 0; ++j){

        if(check(tree[i][j]) != TRUE){
          addopenlist();
          openlist[0].nodeid = tree[i][j];
          openlist[0].parentid = id;
        }
        checkopenlist(tree[i][j]);
      }
      break;
    }
    ++i;
  }
}

void addopenlist()
{
  int i;

  for(i = openlistep; i > 0; --i){
    openlist[i] = openlist[i - 1];
  }
  ++openlistep;
}

void initlist()
{
  openlist[0].nodeid = START;
  openlist[0].parentid = 0;
  ++openlistep;
}

void printlist()
{
  int i;

  printf("\n オープンリスト    ");
  for(i = 0; i < openlistep; ++i){
```

```
    printf("%d[%d],", openlist[i].nodeid,
                      openlist[i].parentid);
  }
  printf("\n");

  printf("クロースドリスト  ");
  for(i = 0; i < closedlistep; ++i){
    printf("%d[%d],", closedlist[i].nodeid,
                      closedlist[i].parentid);
  }
  printf("\n");
}
```

2.3 使い方

```
./depth
```