1　文字列処理関数　216 頁

1.1　学習のポイント

文字列のコピー、比較などを行なう文字列処理関数について学びます。

代表的な文字列処理関数として以下のものがあります。

関数	機能	引数の型	関数の型
strcpy(s,t)	配列 s に文字列 t をコピー	char *	char *
strcat(s,t)	配列 s に文字列 t を連結	char *	char *
strcmp(s,t)	文字列 s に文字列 t の大小比較	char *	int
strlen(s)	文字列 s の長さを返す。	char *	unsigned

これらの関数は、stdlib.h でプロトタイプ宣言されています。なお、char *型の引数としては具体的には次のようなものが書けます。
”Turbo”などの文字列定数
char s[20]; などと宣言されている配列名 s。
char *str; などと宣言されているポインタ str。

1.2　例題 48 reidai48.c

2 つの文字列のうち、小さい方を min に格納しなさい。

```
/*
  2 つの文字列のうち、小さい方を min に格納しなさい。
  reidai48.c
*/

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

int main()
{
  char s[20];
  char t[20];
```

```
  char min[20];

  printf("moji 1 ? "); scanf("%s", s);
  printf("moji 2 ? "); scanf("%s", t);

  if(strcmp(s, t) < 0){
    strcpy(min, s);
  }else{
    strcpy(min, t);
  }

  printf("%s\n", min);

  return 0;
}
```

1.3 練習問題 48 rensyu48.c

2 つの文字列のうち、小さい方を先頭にして buf に格納しなさい。

```
/*
  練習問題 48
  2 つの文字列のうち、小さい方を先頭にして buf に格納しなさい。
  rensyu48.c
*/

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

int main()
{
  char s[20];
  char t[20];
  char buf[50];

  printf("moji 1 ? "); scanf("%s", s);
  printf("moji 2 ? "); scanf("%s", t);

  if(strcmp(s , t) < 0){
```

```
    strcpy(buf, s);
    strcat(buf, " ");
    strcat(buf, t);
  }else{
    strcpy(buf, t);
    strcat(buf, " ");
    strcat(buf, s);
  }

  printf("%s\n", buf);

  return 0;
}
```

2　AI による大規模データ処理入門

2.1　言語処理アルゴリズム

自然言語で記述された大規模データから、その特徴を抽出する方法を考えます。人工知能の歴史においては、自然言語処理の研究は、機械翻訳や自動要約などへの応用を念頭に置いて、さまざまな側面から取り組まれてきました。

2.2　n-gram による特徴抽出

自然言語で表現された文書の表層的な特徴を抽出することで、文書の特徴を表現します。

記号列の特徴を調べるために、同じような部分記号列を n 個の記号の並びとして切り出し、同じ n 個の記号の並びが繰り返し出現しているかどうかを数え上げます。このとき、n 個の記号の並びを n-gram と呼びます。

例では、英文の入力に対して、n=5 として 5-gram を作成していきます。例のように、5 文字の記号の並びを先頭から作成していき、その頻度を調べます。

解析対象文

```
Alice was beginning to  (5 文字の記号の並び (5-gram) を
                         解析対象文の先頭から作成していく。)
Alice
```

```
 lice
 ice w
 ce wa
 e was
  was
```

2.3 n-gram を作成するプログラム ngram.c

```
/************************************/
/*     n-gram の作成                */
/*        ngram.c                   */
/************************************/

#include <stdio.h>
#define N 5 /*n-gram の長さ n*/

/*関数のプロトタイプの宣言 */
void initngram(char ngram[]) ;
            /*n-gram の初期化*/
void putngram(char chr,char ngram[]) ;
         /* n-gram の出力       */

/**********************/
/*     main() 関数     */
/**********************/
int main()
{
 char chr  ;/*入力文字*/
 char ngram[N+1];/*n-gram*/

 /*n-gram の初期化*/
 initngram(ngram) ;

 /*n-gram の出力*/
 while((chr=getchar())!=EOF){
  putngram(chr,ngram) ;/*出力*/
 }

 return 0 ;
```

```
}
/**********************/
/* putngram() 関数      */
/* n-gram の出力        */
/**********************/
void putngram(char chr,char ngram[])
{
 int i ;

 for(i=0;i<N-1;++i)
  ngram[i]=ngram[i+1] ;
 ngram[N-1]=chr ;

 printf("%s\n",ngram) ;
}

/**********************/
/* initngram() 関数     */
/* n-gram の初期化      */
/**********************/
void initngram(char ngram[])
{
 int i ;

 for(i=0;i<N;++i)
  ngram[i]=' ' ;

 ngram[N]='\0' ;/*文字列の終わり*/
}
```

n-gram.c プログラムを実行すると、入力テキストの文末に含まれる改行コードなども含めて n-gram を作成します。しかし、一般には、n-gram を作成する際には改行コードなどの制御記号は入力テキストから除いておく必要があります。

2.4 制御記号を削除するプログラム getch.c

```
/************************************/
/*  データ前処理プログラム          */
/*          getch.c                 */
```

```
/*  文字や数字の取り出し              */
/*  （制御記号の削除）                */
/************************************/

#include <stdio.h>

/**********************/
/*     main() 関数     */
/**********************/
int main()
{
 char chr  ;/*入力文字*/

 while((chr=getchar())!=EOF){
  /*制御記号以外なら出力*/
  if((' '<=chr) && (chr<='z'))
    putchar(chr);/*一文字出力*/
 }

 return 0 ;
}
```

2.5 ngram.exe の使い方

```
./getch < alice.txt | ./ngram | sort | uniq -c | sort -rg
```