C Language Overview

D.E ZEGOUR

École Supérieure d'Informatique

ESI

C Language Overview

Program Structure

Include Files
Declaration of global variables

Function 1
Function 2
*
*
Function n

Main program

The main program starts with main().

C Language Overview

Example of a C Program

```
#include <stdio.h>
#include <stdlib.h>
#include <Time.h>

/**  Implementation- **\: LIST Of INTEGERS**/

typedef int Typeelem_Li   ;
typedef struct Cell_Li * Pointer_Li ;

struct Cell_Li
 {
   Typeelem_Li  Val ;
   Pointer_Li Next ;
 } ;

Pointer_Li Allocate_cell_Li (Pointer_Li *P)
 {
   *P = (struct Cell_Li *) malloc( sizeof( struct Cell_Li)) ;
   (*P)->Next = NULL;
 }
```

```
 void Ass_val_Li(Pointer_Li P, Typeelem_Li Val)
  {
    P->Val = Val ;
  }
 void Ass_adr_Li( Pointer_Li P,  Pointer_Li Q)
  {
    P->Next = Q ;
  }
 Pointer_Li Next_Li(  Pointer_Li P)
  { return( P->Next ) ;  }
 Typeelem_Li Cell_value_Li( Pointer_Li P)
  { return( P->Val) ; }
 void Free_Li ( Pointer_Li P)
  { free (P);}
/** Variables of main program **/
 Pointer_Li L=NULL;

 /** Standard functions **/
 int Randnumber( int N )
  { return ( rand() % N ); }
```

C Language Overview

Example of a C Program

```
 /** Function prototypes **/
void Display (Pointer_Li *L);
void Create (Pointer_Li *L);

void Display (Pointer_Li *L)
 {
  /** Local variables **/
  Pointer_Li P=NULL;
 /** Body of function **/
  P  =  *L ;
  while( P != NULL) {
    printf ( " %d", Cell_value_Li(P) ) ;
    P  =  Next_Li ( P );
  }
 }
  void Create (Pointer_Li *L)
 {
  /** Local variables **/
  Pointer_Li P=NULL;
  Pointer_Li Q=NULL;
  int I;
  int N;
```

```
  /** Body of function **/
  printf ( " %s", "Number (N) of elements to put in the linked list:" ) ;
  scanf ( " %d", &N ) ;
  printf ( " %s", "N=" );
  printf ( " %d", N ) ;
  *L  =  NULL ;
  for( I  =  1 ;I <=  N ; ++I){
    Allocate_cell_Li (& P ) ;
    Ass_val_Li ( P , Randnumber(1000 ) ) ;
    if( *L == NULL) {
      *L  =  P; }
    else  {  Ass_adr_Li ( Q , P );     } ;
    Q  =  P ;
  } ;
  Ass_adr_Li ( P , NULL ) ;
 }
int main(int argc, char *argv[])
 {
  srand(time(NULL));     Create ( & L ) ;     Display ( & L );
  system("PAUSE");
  return 0;
}
```

C Language Overview

Simple Data Types

The C language deals with several types of variables.

The most commonly used ones are as follows, along with their equivalents in Pascal:

C	Pascal
char	char
int	integer
float	real

In C, there is no boolean type. The value 0 represents False, and any value different from 0 represents True.

C Language Overview

Definition of types

A type is defined as follows:

```
struct StructureName
{
  Field 1;
  Field 2;
  ...
  Field n;
}
```

Field 1, ... Field n are definitions of variables of any type.

Variables of this type are defined as:

StructureName N1, N2, ...;

Access is done using the dot notation.

N1.Field1

C Language Overview

Arrays

An array is declared as follows:

TypeName[size]

The index of the first element is 0.
The last element has an index of size - 1.

Example : int A(20); char P(10)

C Language Overview

Strings

A string is an array of characters. Every string must end with the null character '\0'.

Here are some string operations:

strcat(s1, s2): s1 receives the concatenation of s1 and s2.
strcmp(s1, s2): returns 0 if s1 = s2, a positive value if s1 > s2, and a negative value if s1 < s2.
strcpy(s1, s2): copies s2 into s1.
strlen(s): gives the number of characters in s, including '\0'.

C Language Overview

Expressions

Here is the correspondence between Pascal and C operators:

	C	PASCAL
Arithmétique	+, -, *, /, %	+, -, *, /, Mod
Logique	!, &&, ||	Not, And, OR
Relation	<, <=, >, >=, !=, ==	<, <=, >, >=, <>, =

C Language Overview

Statements

The main C instructions are as follows:

Block:	{Declarations; inst1; inst2; ... instn}
Assignment:	V = Expression
Composite Instruction:	{inst1; inst2; ... instn}
Conditional:	if (expression) inst
Alternative:	if (expression) inst else inst
While Loop:	while (expression) inst
For Loop:	for (initialization; condition; expression) inst

C Language Overview

Some shortcuts

i++	i = i + 1
X += Y	X = X + Y
A *= B + C	A = A * (B + C)

C Language Overview

Basic Input/Output Functions

Here are some basic input/output operations:

getchar(): reads a character from the screen.
putchar(): writes a character to the screen.
gets(): reads a string of characters from the screen.
puts(): writes a string of characters to the screen.
printf("format string", argument list):
reads data from the screen controlled by formats.

"format string" consists of two things:

a) the characters to print
b) the format of arguments starting with %

C Language Overview

Basic Formats

Format	Explanation
%c	character
%d, %i	decimal
%f	floating-point decimal
%s	string of characters
%o	octal
%x	hexadecimal
\n	newline
\f	form feed
\\	backslash

Formats can be preceded by a width. It is the number of characters on which the data will be written.

Example:
printf("This is an %s %d %c", example, 10, '.') gives ‘This is an example 10.’

Example :
scanf("%d %S3, &cpr, &add)
reads a decimal into cpr, then reads a string of characters into add.

C Language Overview

Other Input/Output Functions

fscanf(): similar to the scanf() function, except it reads data from a file.

fprintf(): similar to the printf function, but it outputs to a file.

fopen(): opens a file for use and returns a pointer that identifies the file.

fclose(): closes a previously opened file.

feof(): predicate equal to true if the end of the file is reached, false otherwise.

rewind(): positions the file pointer at the beginning of the file.

C Language Overview

Other Input/Output Functions

Opening a file is done as follows:

fp = fopen("physical file name", mode)

where fp is declared as File *fp.

The mode can take the following values:
"r": open the file in read mode
"w": create a file in write mode
"r+": open a file for read/write
"w+": create a file for read/write

To use all the functions we've described, we need to include the stdio.h file in the program's declaration section: #include <stdio.h>

C Language Overview

Procedures and functions

In C, we only talk about functions. However, you can simulate procedures using the void keyword.

The form of a procedure declaration is as follows:

```
type FunctionName(parameter declaration lists)
{
  Declaration of local variables
  Instructions
}
```

The return statement must exist and return the value of the function.

C Language Overview

Procedures and functions : Example

```
void Interchanger( int *X, int *Y);
{
  int Temp;
  Temp = *X;
  *Y = *X ;
  *Y = Temp
   }
main()
{
  X = 10;
  Y = 20;
  Interchanger(&X, &Y);
  printf("X = %d, Y = %d", X, Y);
}
```

C Language Overview

Buffered File Operations

The C language has a library of functions for performing file operations in buffered mode. These functions include:

fread(&buffer, sizeof(buffer), 1, fp): reads the current record into the Buffer area. sizeof() is a function that returns the size of a variable (in bytes).

fwrite(&buffer, sizeof(buffer), 1, fp): writes the Buffer area to the file at the current position.

fseek(fp, offset, origin): moves to a position in the file.

offset is the number of records from the origin.
origin can be 0 (beginning of the file), 1 (current position), or 2 (end of the file).

C Language Overview

Buffered File Operations

Buffer is a structure of a certain type.

The logical file fp is declared as FILE *fp.
The fopen and feof operations are used in this mode.

To use all the functions we've described, we need to include
the stdio.h file in the program's declaration section: #include <stdio.h>.

C Language Overview

Example : C Program

```
#include <stdio.h>
 struct  Typeblock
      {
        int Nb;
        int Tab[10];
      };

FILE *Fp ;
 struct Typeblock  Buffer;
 int I;

main()
    {
      Fp = fopen("F.c", "w+b");
      for ( I=0; I<10; I++)
         {
           Buffer.Nb = 10*I;
           Buffer.Tab[1] =50*I;
          fwrite(&Buffer, sizeof( Buffer), 1, Fp);
         }
```

```
   Buffer.Nb = 666;
   Buffer.Tab[1] = 999;
   I=6;
   fseek (Fp, (long) ( (I-1)* sizeof( struct Typeblock ) )  , 0 );
   fwrite(&Buffer, sizeof( Buffer), 1, Fp);

    rewind(Fp);
    printf( " Buffer Size= %d \n", sizeof( Buffer) );
    for ( I=0; I<10; I++)
       {
          fread(&Buffer, sizeof(Buffer), 1, Fp);
         printf(" Nb = %d, Tab(1) = %d\n " , Buffer.Nb, Buffer.Tab[1]);
       }

    I=7;
    printf("Reading the %d -th block \n",I);
    fseek (Fp, (long) ( (I-1)  * sizeof( struct Typeblock))  , 0 );
    fread(&Buffer, sizeof(Buffer), 1, Fp);
    printf(" Nb = %d, Tab(1) = %d\n " , Buffer.Nb, Buffer.Tab[1]);
}
```

C Language Overview

Example : Results

Buffer Size= 44
Nb = 0, Tab(1) = 0
Nb = 10, Tab(1) = 50
Nb = 20, Tab(1) = 100
Nb = 30, Tab(1) = 150
Nb = 40, Tab(1) = 200
Nb = 666, Tab(1) = 999
Nb = 60, Tab(1) = 300
Nb = 70, Tab(1) = 350
Nb = 80, Tab(1) = 400
Nb = 90, Tab(1) = 450
Reading the 7 -th block
Nb = 60, Tab(1) = 300