



## Inleiding

In deze cursus zal een rondleiding plaats hebben in de wereld van GEM. GEM staat voor Graphics Environment Manager en is een middel om een grafische Man Machine Interface (MMI) onafhankelijk te maken van de gebruikte computer. Dit is voordelig omdat dan een applicatie gemaakt kan worden voor verschillende machines. De GEM specificatie is ontwikkeld door een team van Digital research. De invloed van deze specificatie is nog duidelijk te zien in de nu veel gebruikte X omgevingen, welke veel bij Signaal gebruikt worden (Onder andere MOTIF en Open Windows applicaties).

Hoe is de cursus opgebouwd :

Het eerste deel van de cursus behandelt de verschillende onderdelen (programma's) welke nodig zijn om een applicatie in GEM te maken.

In het tweede deel zal aandacht geschonken worden aan de meest gebruikte C statements. Dit is noodzakelijk om de meegeleverde programmatuur te kunnen begrijpen.

In het derde deel zal ingegaan worden in de GEM omgeving met als zwaartepunt de Application Environment Services (AES).

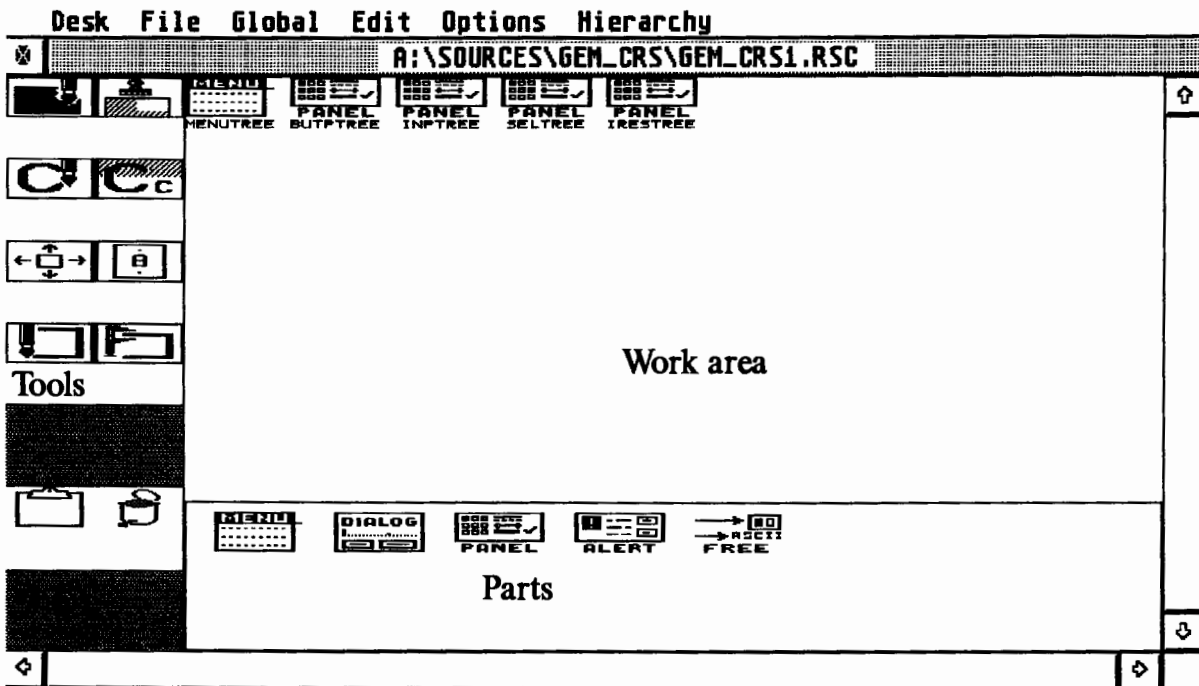
In het vierde deel zal het meegeleverde programma volledig behandeld worden.

Het is bijzonder moeilijk in te schatten hoe groot de belangstelling is en wat de aanvangs kennis is van de belangstellenden. Door improvisatie zal zo goed mogelijk op de situatie ingesprongen worden.

Hier zal de cursus stoppen en zal gekeken worden of eventuele voortzetting van de cursus noodzakelijk is. Een en ander is afhankelijk van de respons, die van de belangstellenden komt.

## 1 De ondersteunende programma's of omgevingen :

In de cursus zal gebruik gemaakt worden van de Resource Construction Set, welke meegeleverd wordt bij het PURE-C ontwikkelings pakket. Verder wordt er gebruik gemaakt van de PURE-C programmeer omgeving voor het maken en aanpassen van C programma's.



Figuur 1 :  
De resource editor.

In het nu volgende wordt een korte beschrijving gegeven van de mogelijkheden binnen de resource construction set en de PURE-C ontwikkel omgeving.

### 1.1 De Resource Construction Set :

Dit programma is gemaakt door Digital research en heeft de versie 2.1. Door middel van dit programma kunnen XXXX.RSC files aangemaakt worden welke gebruikt kunnen worden in de GEM applicatie. Het programma heeft 6 menu ingangen namelijk

Desk	File	Global	Edit	Options	Hierarchy
About GEM	New	Output	Cut	Info	Sort Children
	Open	Protection	Copy	Name	Unhide Children
	Merge	Save Pref.	Paste	Type	Remove parent
	Close	Hide Parts	Delete	Load	
	Save	Hide Tools			
	Save as				
	Abandon				
	Quit				

### 1.1.1 Desk menu :

Als deze ingang geselecteerd wordt dan komt een informatie dialog op, waarin de ontwerpers en het versie nummer vermeld staan.

### 1.1.2 File menu :

In dit menu worden alle file bewerkingen afgehandeld; tevens is er de menu functie Quit om het programma te verlaten.

-- New :

Met New kan men de werk window schoon maken; alle voorgaande bewerkingen worden weg gegooid.

- Open :

Met deze functie kan een reeds eerder gemaakte resource opgehaald worden.

- Merge :

Met deze functie kan een eerder gemaakte resource bijgeladen worden.

- Close :

Hiermee kan een geopende resourceboom gesloten worden. Het geopende object wordt in een icoon veranderd.

- Save :

Een aangepaste resource wordt onder dezelfde naam weer weggeschreven.

- Save as :

De resource wordt onder een nieuwe naam weggeschreven.

- Abandon :

Ongeveer als "New" de resource wordt weggegooid.

- Quit :

Het programma wordt verlaten.

### 1.1.3 Global menu :

- Output :

Selectie voor de generatie van definitie files (voor ons (\*H)). Selectie voor wel of niet uitvoer van een resource build file.

- Protection :

Selectie van mogelijkheden om te voorkomen dat met foutieve handelingen objecten corrupt worden.



**- Save Preferences :**

De geselecteerde instellingen worden weggeschreven op schijf.

**- Hide Parts :**

Alle mogelijke objecten worden wel of niet vertoond.

**- Hide Tools :**

De instellings mogelijkheden van de object eigenschappen (properties) worden wel of niet vertoond.

### 1.1.4 Edit menu :

**- Cut :**

Een geselecteert object of objectboom wordt weggehaald en op het prikbord gezet.

**- Copy :**

Een geselecteert object of objectboom wordt gekopieerd en op het prikbord gezet.

**- Paste :**

Het object of objectboom welke op het prikbord staat wordt naar het edit venster gedirigeerd.

**- Delete :**

Een geselecteert object of objectboom wordt weggehaald.

### 1.1.5 Options menu :

**- Info :**

Geeft informatie over aantal gegenereerde objecten en geheugen gebruik.

**- Name :**

Met deze functie kan de naam van het object in bewerking veranderd worden.

**- Type :**

Met deze functie kan het object type aangepast worden.

**- Load :**

Met deze fuktie kunnen images voor icoon- en image objecten ingelezen worden (Welk formaat er toegevoerd moet worden is mij onbekend).

## 1.1.6 Hierarchy menu :

### - Sort Children :

Met behulp van een dialog kan de volgorde van de kinderen van een object gekozen worden.

### - Unhide Children :

Alle kinderen die de property hide hebben opstaan worden weer vertoond.

### - Remove parent :

Een parent wordt verwijderd uit een object boom; de kinderen worden aan een bovenliggend parent gekoppeld.

## 1.2 PURE C ontwikkel omgeving

Het PURE-C pakket vindt zijn oorsprong in de firma BORLAND, welke een grote reputatie heeft opgebouwd bij het ontwerpen van compilers. De firma heeft echter de Atari tak, toen onder de naam TURBO-C, afgestoten aan het software house Application Systems Heidelberg en heeft zich voornamelijk gericht op de PC wereld. Vanaf dat moment heet het pakket PURE-C en bestaat uit een geïntegreerde assembleer, compileer, link, debug en edit omgeving. De menu opbouw is als volgt :

Pure C	File	Edit	Search	Compile	Project	Options	Help
About	Open .C	UndoFind	Compile	Select	Shell	Menu	
	Open .H	Cut	Find same	Compile ""	Debug	Compiler	Editor
	Open .S	Copy	Find sel.	Assemble	Run	Assembler	C Language
	Print	Paste	Replace	Assemble ""	Make	Linker	Libraries
	Print sel.	Select all	Replace same	Execute	Make all	LoadOptions	
	Close	Shift left	Check braces	Link	Save ""	Assembler	
	Abandon	Shift right	Find error	Index			
	Save	Cycle wind.	Goto line				
	Save as	User screen					
	TOS shell						
	Quit						



### 1.2.1 Pure C menu :

Informatie over het programma (Makers en revisie).

### 1.2.2 File menu :

- Open .C; - Open .H; - Open .S; - Open .PRJ :

Een file selection box wordt geopend, waarin een file geselecteerd kan worden.

- Print; - Print selection :

De inhoud van een window of een gedeelte daarvan wordt afgedrukt op een aangesloten printer.

- Close :

De actieve editor window wordt gesloten.

- Abandon :

De aangebrachte wijzigingen in het actieve window worden ongedaan gemaakt.

- Save :

De inhoud van de actieve window wordt weggeschreven op schijf; de oude file wordt overschreven.

- Save as :

De inhoud van de actieve window wordt weggeschreven onder een nieuwe naam, via een selectie in de file selection box.

- TOS shell :

Geeft toegang tot de TOS shell (als deze gebruikt wordt).

- Quit :

Verlaat het programma.

### 1.2.3 Edit menu :

- Undo :

Herstelt de laatste edit actie.

- Cut :

De geselecteerde tekst wordt weggehaald en in de cut/paste buffer gezet.

- Copy :

De geselecteerde tekst wordt gekopieerd in de cut/paste buffer.

- Paste :

De tekst in de cut/paste buffer wordt op de cursorpositie ingevoegd.

- Select all :

Alle tekst van de actieve window wordt geselecteerd.

- Shift left; - Shift right :

Verschuift de geselecteerde tekst een tab positie (In mijn ogen een onbenullige functie).

- Cycle windows :

De onderliggende window wordt naar boven gehaald en actief gemaakt.

- User screen :

Het scherm schakelt over naar het scherm van de voorheen opgestarte applicatie of weer terug.

### 1.2.4 Search menu :

- Find :

De find dialog komt op, waarin een zoek string opgegeven kan worden. Deze string wordt vanaf de cursor positie in de opgegeven richting gezocht. Als de string in de actieve window gevonden wordt dan wordt de tekst geselecteerd.

- Find same :

De zoekfunctie zoekt vanaf de cursorpositie naar de voorgaand geselecteerde string geactiveerd met "Find".

- Find selection :

Deze functie zoekt naar een string, welke aangegeven wordt met een selectie in het actieve window. De zoek richting is die van de "Find" functie. De string wordt overgenomen in de "Find" dialog.

- Replace :

De replace dialog komt op waarin de zoek en replace string opgegeven kunnen worden. Tevens kan opgegeven worden welke zoek richting en of alle tekst vanaf de cursor positie vervangen moet worden.

- Replace same :

De ingestelde waardes van de "Replace" dialog worden opnieuw geactiveerd.

- Check braces :

Deze functie doorzoekt de C code op de compound karakters ')' en '{' en controleert of elke open zijn sluit tegen hanger heeft.

- Find error :

Deze functie zoekt via een geselecteerde regel in het messages window naar de positie van de geselecteerde regel in de code.

- Goto line :

Deze functie selecteert de aangegeven regel van het actieve window.



### 1.2.5 Compile menu :

- Compile :

Compileert een geselecteerde file welke in de opgekomen file selection box geselecteerd kan worden.

- Compile "" :

Compileert de kode in de actieve window.

- Assemble :

Assembleert een geselecteerde file welke in de opgekomen file selection box geselecteerd kan worden.

- Assemble "" :

Assembleert de kode in de actieve window.

- Execute :

Een extern programma kan opgestart worden, bijvoorbeeld de resource construction set, terwijl de C omgeving in het geheugen van de computer blijft.

### 1.2.6 Project menu :

- Select :

Een file selection box komt op waarin een project file gekozen kan worden.

- Debug :

Als aanwezig, wordt de debugger opgestart.

- Run :

Het programma aangegeven in de project file (XXXX.PRJ) wordt getest op generatie tijd en modificatie tijd, waarna er eventueel gecompileerd en gelinkt wordt, waarna het programma opgestart wordt. De C omgeving blijft resident in het geheugen.

- Make :

Het programma aangegeven in de project file (XXXX.PRJ) wordt getest op generatie tijd en modificatie tijd, waarna er eventueel gecompileerd en gelinkt wordt.

- Make all :

Het programma aangegeven in de project file (XXXX.PRJ) (alle files) wordt gecompileerd en gelinkt.

- Link :

Het programma aangegeven in de project file (XXXX.PRJ) (alle files) wordt gelinkt.



### 1.2.7 Options menu :

- Shell; - Compiler; - Assembler; - Linker :

Instellingen voor de omgeving, compiler, assembler en linker.

- Load :

Een file selection box komt op, waarmee een omgevings definitie file geselecteerd kan worden.

Deze file zorgt ervoor dat er een nieuwe omgeving opgebouwd wordt compleet met het opnieuw laden van tekst editors copiler linker settings en selectie van een project file.

- Save "" :

De huidige omgevings definitie wordt weggeschreven op schijf.

### 1.2.8 Help menu :

- Menu; - Editor; - C Language; - Libraries; - Options; - Assembler; - Index :

Als de help file aanwezig is kunnen verschillende help entries geselecteerd worden.

### 1.2.9De project file :

PURE-C kent een projectfile, welke enigzins te vergelijken is met een makefile in de UNIX wereld. Met deze file is het mogelijk om meerdere files tegelijk te compileren en linken. De XXXX.PRJ file bevat een opsomming van files en libraries, waaruit een programma is opgebouwd. Afhankelijkheden van files kunnen hier ook vastgelegd worden.

Voorbeeld van een project file :

```
; GEM-CRS1.PRJ for use with single module programs
; -----
01 gem-crs1.prg ; name of executable program
02 .C [-IA:\include -IA:\sources\gem-crs\rsc -IA:\sources\gem-crs]
03 = ; list of modules follows...
04 A:\lib\pcstart.o ; startup code
05 A:\sources\gem-crs\gem-crs1 ; main
06 A:\lib\pcstdlib.lib ; standard lib
07 A:\lib\pcextlib.lib ; extended lib
08 A:\lib\pctoslib.lib ; TOS lib
09 A:\lib\pcgemlib.lib ; AES and VDI lib
```



; remove unused libraries to speed up linking!

Voor het "=" teken op regel 03 wordt de programma naam gespecificeerd (regel 01) en de compiler en linker opties (regel 02). Compiler opties worden als volgt opgegeven : C [optie 1] <optie 2> ... <optie n>]. De linker optie wordt opgegeven als volgt : L [optie 1] <optie 2> ... <optie m>].

Het systeem controleert automatisch op modificatie en generatie tijd van afhankelijke files.

Voorbeeld :

In regel 05 staat de file gem-crs1; de extentie is ".C" bij deze file hoort een object file met extentie ".O".

Als de "Make" functie geactiveerd is en de datum\tijd van de objectfile is ouder dan die van de c-file dan start de ontwikkel omgeving de compiler op om de c-file te compileren. Is gem-crs1 een assemblerfile met extentie ".S" dan wordt automatisch de assembler opgestart. Wil men deze afhankelijkheid niet dan is het mogelijk door een extentie van ".O" toe te voegen om compileren of assembleren te voorkomen (zie regel 04). Ook kunnen er in de projekt file libraries opgegeven worden (zie regel 06 ... 09).

## 2 De taal C

### Historie :

De taal C is eerste instantie ontstaan door de behoefte om een systeem of programma duidelijker en doorzichtiger te maken dan mogelijk is met behulp van een assembler taal. In het begin van de computer techniek in 1972 werd de taal C voor het eerst ontwikkeld in het AT&T Bell laboratorium door D.M. Ritchie voor de ontwikkeling van operating systemen zoals MSDOS, VAX VMS en UNIX. De taal bleek niet alleen heel goed te voldoen om een operating systeem te ontwerpen maar het bleek ook een ideale manier te zijn om de code onafhankelijk te maken van de gebruikte machine. Het overzetten van de code van de ene machine naar de andere heet porting.

In 1977 schreven Ritchie en Kernigham hun specificatie van de taal C in hun boek "The C Programming Language" in de volksmond K&R standaard. Later bleek de standaard, voornamelijk bij het porteren van software niet geheel te voldoen. Daarom is de specificatie later nog aangepast door de American National Standards Institute (ANSI). Daarom wordt er tegenwoordig alleen nog gewerkt met deze ANSI standaard.

## C contra andere talen :

De taal C is niet de enigste programmeertaal maar heeft wel in zeer veel systemen gebruikt. Andere talen zijn onder andere BASIC, PASCAL, MODULA, ADA en LISP. Elk van die talen hebben over het algemeen een eigen toepassings gebied.

Doordat C de programmeur een grote vrijheid laat is het toepassings gebied zeer breed. Hierdoor heeft de taal de naam gekregen "Rommelig", "Onduidelijk" en "Cryptisch" te zijn. Als je de code van sommige programmeurs ziet dan lijkt het meer geheim schrift dan programmatuur. Data typering en data conversie is dan ook lang zo strak niet gehandhaaft dan bij talen als PASCAL en ADA.

## Programma ontwikkeling :

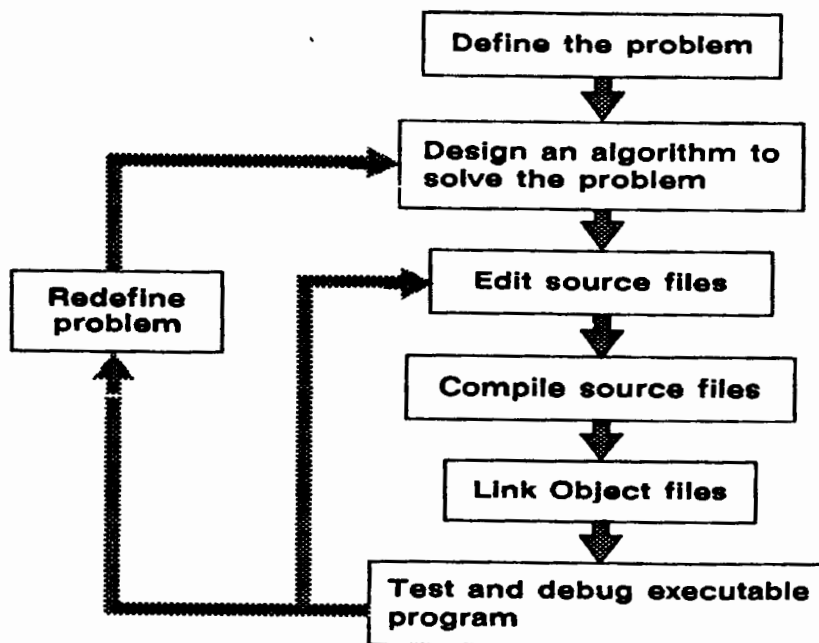
De ontwikkeling van een programma verloopt meestal in een aantal fasen meestal de lifecycle van het programma genoemd :

- Definitie van het programma :

In deze fase wordt bepaald welke functies het programma moet uitvoeren en hoe een operator met het programma moet werken.

- Ontwerp fase :

In deze fase worden de algorithmes vastgelegd, welke nodig zijn voor het realiseren van de gewenste functies.



Figuur 2 :  
Het software ontwikkel proces.

- Coderings fase :

In deze fase worden modules gemaakt, welke in de ontwerpfase vastgelegd zijn; tevens wordt in deze fase ook de gehele programmeer omgeving gemaakt, nodig voor compilatie en linken van de verschillende modules. Er worden ook simpele testen uitgevoerd om primitieve algorithmes te testen (alpha testing).

- Test fase :

In deze fase wordt het programma getest of alle functies, welke in de definitie fase gewenst zijn, aanwezig zijn en werken.

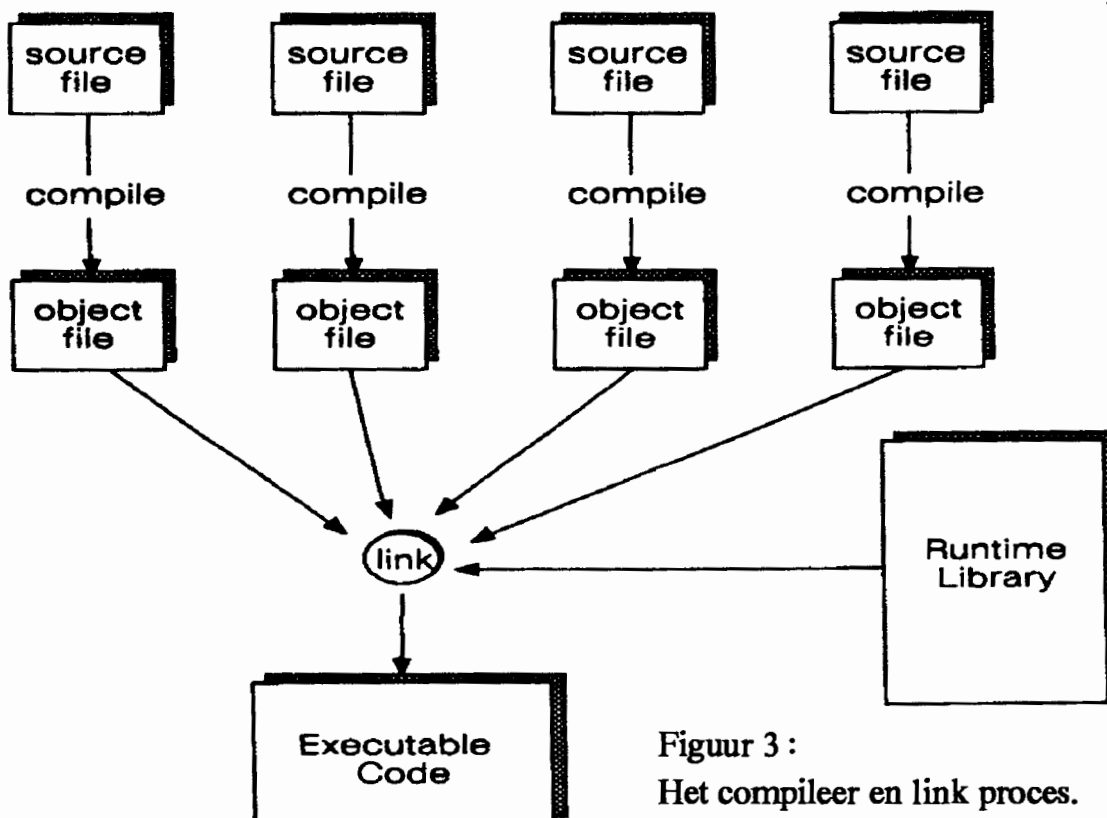
- Ongehouds fase :

Als het programma in gebruik genomen wordt dan heeft de gebruiker meestal nog wensen m.b.t. het gebruik van het programma. De aanpassingen zijn allemaal wijzigingen op de specificatie gemaakt in de definitie fase.

De ontwikkel omgeving :

Voornamelijk in de coderings - en test fase wordt er gebruik gemaakt van een ontwikkelings omgeving (In ons geval de PURE-C omgeving). Dit is nodig om zonder al te veel ophef een programma te kunnen genereren (zie ook figuur 2).

In de omgeving is het mogelijk om alle files achter elkaar te compileren en te linken (zie figuur 3).



Figuur 3 :  
Het compileer en link proces.

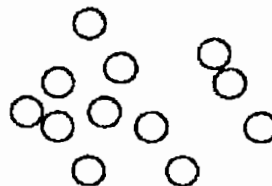
## 2.1 De C fundamentelementen :

Het fundamentele element van de taal C is de functie. Je kunt wel zeggen dat alle elementen van de code opgebouwd worden door middel van alleen maar functies. Het begin van een programma ligt dan ook altijd vast bij de beginfunctie

main. Vanuit deze functie worden andere subfuncties aangeroepen. Deze subfuncties hoeven niet alleen in de file te staan, waarin ook main gedeclareerd staat. Functies kunnen ook geïmporteerd worden vanuit andere files.

Uit alles blijkt dat er een sterke hiërarchische opbouw is van zowel de omgeving als in de C files zelf (Zie figuur 4).

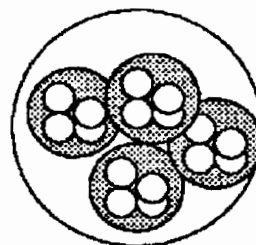
**Machine Instructions:** At the lowest level, every program consists of primitive machine instructions.



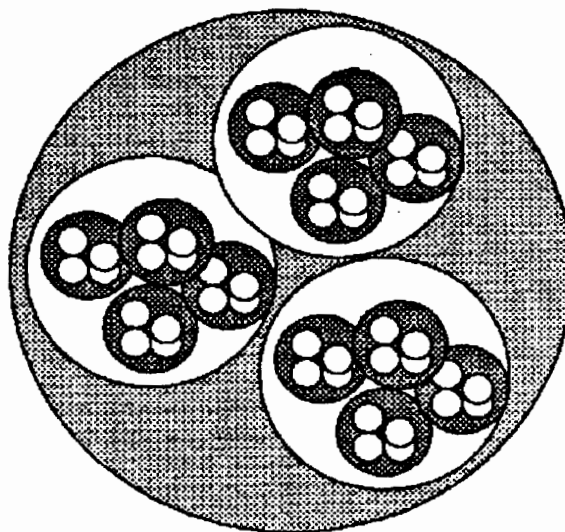
**Language Statements:** High-level languages consist of statements that perform one or more machine instructions.



**Functions:** Functions consist of groups of language statements.



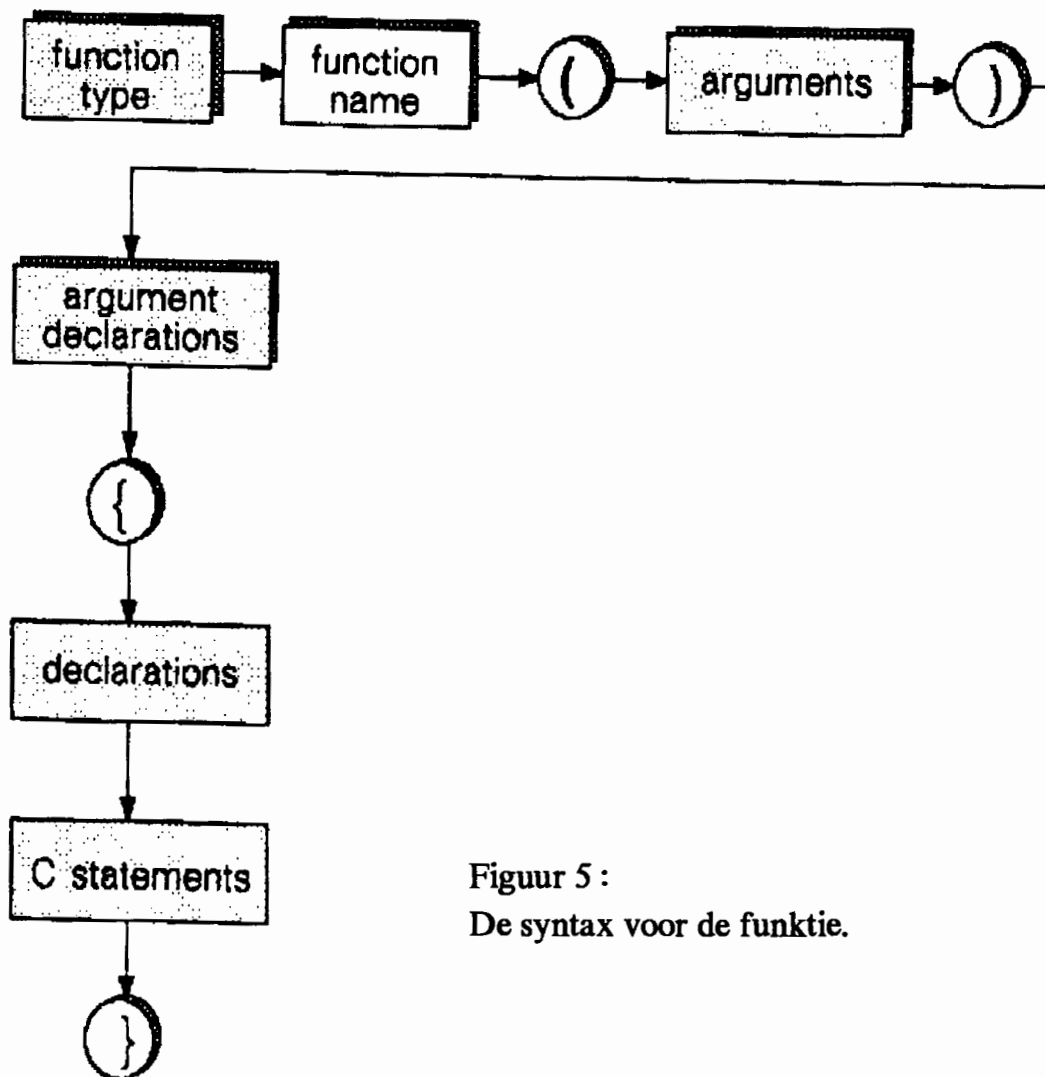
**Programs:** Programs consist of groups of functions.



**Figuur 4 :**  
De hiërarchische opbouw van de verschillende onderdelen.

### 2.1.1 De funktie :

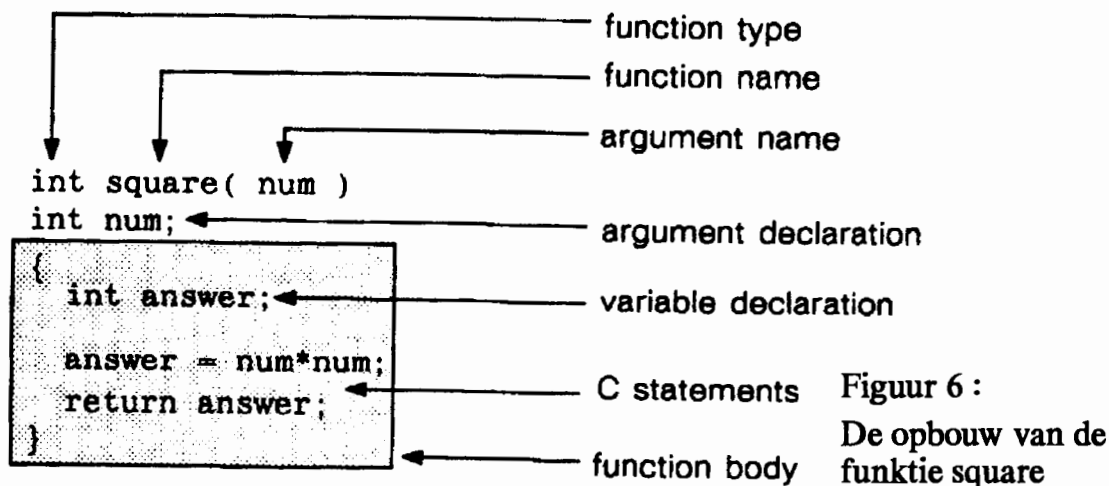
De funktie is opgebouwd volgens figuur 5.



Figuur 5 :  
De syntax voor de funktie.

Hoe ziet dit er in de praktijk uit :

In figuur 6 staat een funktie declaratie, welke het kwadraat van een variabele berekend. De declaratie bestaat uit 3 delen. Het eerste deel "int" is een reserved keyword wat integer betekend. Het tweede deel is de funktie naam en het derde deel is een lijst van parameter variabelen.



## 2.2 Variabelen en constanten :

Variabelen en constanten hebben beide te maken met getallen. Het verschil tussen de twee is dat de waarde van de constante bepaald wordt tijdens de compileer actie, terwijl de waarde van de variabele bepaald wordt gedurende de executie van het programma. Het statement  $j = 5 + 1$  ziet er runtime uit als  $j = 6$ ; decompiler rekent de expressie gewoon uit en zet het resultaat in de executiecode. Anders gaat het met het statement  $j = j - 2$ ; hier moet de computer de variabele ophalen bewerking uitvoeren en de variabele weer terug zetten.

### 2.2.1 Variabele naam :

De taal C is case sensitive, wat betekend dat voor de variabele naam zowel hoofdletters als kleine letters gebruikt kunnen worden. Enkele voorwaarden van een variabelenaam zijn :

- Een naam mag niet beginnen met een digit.
- Een naam mag geen "\$" bevatten.
- Een naam mag geen speciale karakters bevatten zoals "%", "#", ".", maar wel een underscore "-".
- Een naam mag niet gelijk zijn aan een reserved word.

Reserved words :

auto	double	int	struct	break
else	long	switch	case	enum
register	typedef	char	extern	return
union	const	float	short	unsigned
continue	for	signed	void	default
goto	sizeof	volatile	do	if
static	while			



### 2.2.2 Expressies :

Een expressie is een aanduiding voor rekenkundige bewerking op een of meerdere variabelen. De bewerking op een variabele wordt aangegeven door een operator. Als een expressie meerdere operatoren bevatten dan geldt er tussen de operatoren een bepaalde bewerkings volgorde. Dit is een uitbreiding op de volgorde welke we kennen bij de rekenkundige bewerkingen.

Voorbeeld  $6 * 13 / 3 + 5$  wordt uitgerekend als :

1 -  $x = 13 / 3$ : Van rechts beginnen met de hoogste prioriteit.

2 -  $x = x * 6$

3 -  $x = x + 5$ : Als laatste de laagste prioriteit bewerking uitvoeren.

In figuur 7 staan de verschillende prioriteiten (precedence) van de operatoren.

Class of operator	Operators in that class	Associativity	Precedence	
primary	() [] -> .	Left-to-Right		
unary	cast operator sizeof & (address of) * (dereference) - + - ++ -- !	Right-to-Left		
multiplicative	* / %	Left-to-Right		
additive	+ -	Left-to-Right		
shift	<< >>	Left-to-Right		
relational	< <= > >=	Left-to-Right		
equality	== !=	Left-to-Right		
bitwise AND	&	Left-to-Right		
bitwise exclusive OR	^	Left-to-Right		
bitwise inclusive OR		Left-to-Right		
logical AND	&&	Left-to-Right		
logical OR		Left-to-Right		
conditional	? :	Right-to-Left		
assignment	= += -= *= /= %= >>= <<= &= ^=	Right-to-Left		
comma	,	Left-to-Right		LOWEST

Figuur 7 :  
Volgorde van de operatoren.



### 2.2.2.1 Veel gebruikte bewerkingen :

<code>a = b + c;</code>	: a krijgt de waarde van <code>b + c</code> (statement wordt afgesloten met een <code>”;</code> ).
<code>a += 4;</code>	: Is gelijk aan <code>a = a + 4;</code>
<code>a = b % c;</code>	: a krijgt de restwaarde van de integer deling <code>b / c</code>
<code>a++;</code>	: Is gelijk aan <code>a = a + 1;</code>
<code>a--;</code>	: Is gelijk aan <code>a = a - 1;</code>
<code>a &amp; 0xff;</code> <code>hex.</code>	: De bits van de integer a worden bitwise 'ge' AND met ff
<code>a   0xff;</code>	: Als boven nu met de OR functie.
<code>a &lt;&lt; 1;</code>	: De inhoud van a wordt 1 plaats naar links geschoven.
<code>a &gt;&gt; 1;</code>	: De inhoud van a wordt 1 plaats naar rechts geschoven.
<code>a = ~b;</code>	: a krijgt de inverse waarde van b (Ones complement).

### 2.2.2.2 Logische expressies :

In veel talen kent men de logische waarde "TRUE" of "FALSE". De taal C kent deze in

principe niet. Een logische variabele is "FALSE" als de waarde 0 is. Een logische variabele is "TRUE" als de waard ongelijk 0 is.

Veel gebruikte logische bewerkingen :

<code>a = b &amp;&amp; c;</code>	: De waarde van a is "TRUE" als b en c "TRUE" zijn.
<code>a = b    c;</code>	: De waarde van a is "TRUE" als b of c "TRUE" zijn.
<code>a = !b;</code>	: a krijgt de inverse waarde van b.
<code>a = b &lt; c;</code>	: a is "TRUE" als b kleiner of gelijk is aan c.

### 2.2.3 Data types :

De taal c kent de volgende basis types :

- char	: Woord van 1 byte (8 bits) lengte.
- short	: Woord van 2 bytes lang.
- int	: Woord waarvan de lengte bepaald wordt door degebruikte machine; voor de Atari computer 2 bytes lang.



- long : Woord van 4 bytes lengte.
- signed : Integer met teken bit.
- unsigned : Integer zonder tekenbit.
- enum : Integer gebruikt bij enumeraties.
- float : Een waarde welke opgedeeld is in een exponent en een mantissa; 4 bits lang.
- double : Als een float waarde maar dan 8 bytes lang.

### 2.2.3.1 Constante notaties :

Enkele veel gebruikte integer notaties zijn :

Decimaal	Octaal	Hexadecimaal	Opmerking
3	003	0x3-	
8	010	0x8-	
15	017	0xF-	
16L	020L	0x10	Long type
-87	-0127	-0x57	-
255U	0377	U0xFFU	Unsigned.

Enkele veel gebruikte float notaties zijn :

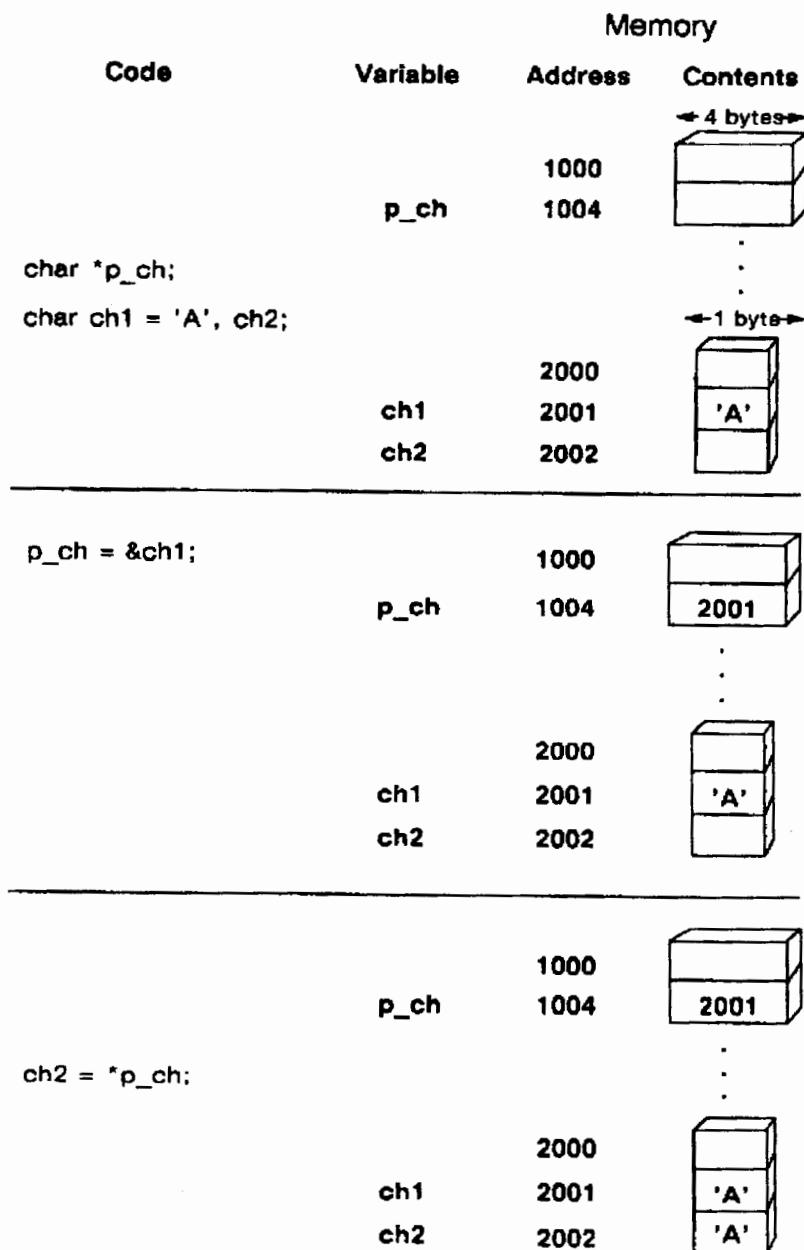
3.1415	: PI
.333333	: 0.333333
3e2	: 300
5E-2	: 0.05

Enkele string declaraties :

- "aap\n" : 'a','a','p','\n','\0' Waarbij "\n" new line voorstelt en "\0" de character waarde 000 voorstelt.
- "\a\b\f\n\r\t\v" : <alert><backspace><formfeed><newline><carriage return><hor. tab><vert. tab>

### 2.2.3.2 Pointers :

In de taal C nemen de pointers een speciale plaats in. Een heleboel benaderingen van een variabele gebeurt via een pointer. Onder andere de benadering van een array gaat via een pointer. Voor het mechaniek van de pointer afhandeling zie figuur 8.



Figuur 8 :  
Pointer mechaniek.



### 2.2.3.2.1 Bewerkingen op pointers :

De kracht van de taal C zit hem hoofdzakelijk in de uitgebreide mogelijkheden om bewerkingen te doen met pointers. Stel er is de volgende situatie :

```
int      i[10];      /* Integer array van 10 integers */
int      *ptr;       /* Integer pointer */

ptr = i;            /* Als een array aangesproken wordt dan mag bij een
                    /* assignment van een pointer geen adres teken "&"
                    /* gebruikt worden. */

ptr += 5;          /* De pointer wordt 5 plaatsen opgehoogd; als een integer
                    /* bestaat uit 2 bytes dan zal de pointer 10 bytes opgehoogd
                    /* worden, */

*ptr = 123;        /* Het element i[5] krijgt de waarde 123 & */
```

Als er een integer bij een pointer opgeteld / afgetrokken wordt, dan wordt de pointer net zoveel element groottes opgeschoven.

### 2.2.3.3 Structures :

Met een structure kun je een datablok samenstellen van verschillende types. Je kunt bijvoorbeeld een aantal integers, floats, characters enz. samenstellen tot een nieuw data type met behulp van de structure.

#### Voorbeeld :

Definitie :

```
struct adres {
    char    straat[40];
    int     huisnummer;
    char    postcode[6];
    char    woonplaats[40];
};
```



Declaratie :

```
struct adres          persoon = {"Kalverstraat", 321, "5678, "Amsterdam"};
struct adres          *adr_ptr;
```

Code :

```
adr_ptr = &persoon;

printf ("%s %d %s\n", adr_ptr->straat, persoon.huisnummer,
adr_ptr->woonplaats);
```

Uitvoer :

Kalverstraat 321 Amsterdam

### 2.2.3.4 Unions :

De union kan ook data elementen samenstellen maar dan op een heel bijzondere manier. De union bepaalt de ruimte, die nodig is om het grootste data te herbergen. Deze ruimte wordt dan gereserveerd. De union kan nu verschillende types herbergen. Het wordt ook wel eens het variant type genoemd.

Voorbeeld :

```
union adres {
    char    straat[40];
    int     huisnummer;
    char    postcode[6];
    char    woonplaats[40];
};

union adres    persoon;
union adres    *adr_ptr;
```



Code :

```
adr_ptr = &persoon;  
  
strcpy (persoon.straat, "Kalverstraat");  
printf ("%s\n", adr_ptr->straat);
```

Uitvoer :

Kalverstraat

### 2.2.3.5 Typedefs :

Met het keyword typedef is het mogelijk om een eigen datatype te maken.  
Voorbeeld als boven :

```
typedef struct {  
    char    straat[40];  
    int     huisnummer;  
    char    postcode[6];  
    char    woonplaats[40];  
} ADRES-TYPE;
```

Declaratie :

```
ADRES-TYPE    persoon = {"Kalverstraat", 321, "4567AB", "Amsterdam"};  
ADRES-TYPE    *adr_ptr;
```

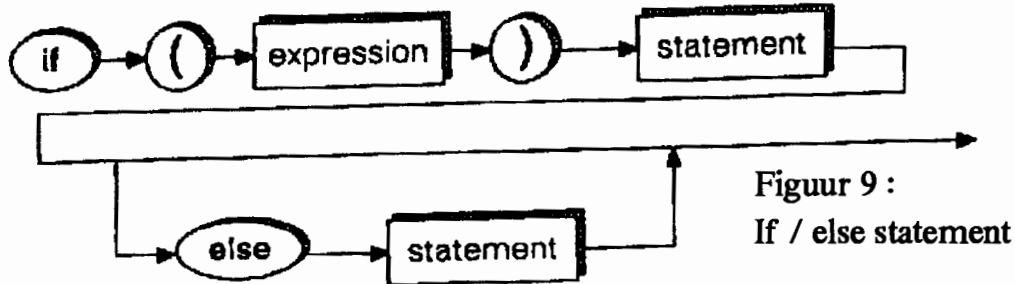
voor de rest is het gelijk aan het bovenstaande voorbeeld.

## 2.3 Flow control :

Het is mogelijk om de programma doorloop conditioneel te veranderen, hiervoor worden de "if", "else" en "switch" statements gebruikt.



### 2.3.1.1 Het if / else statement :



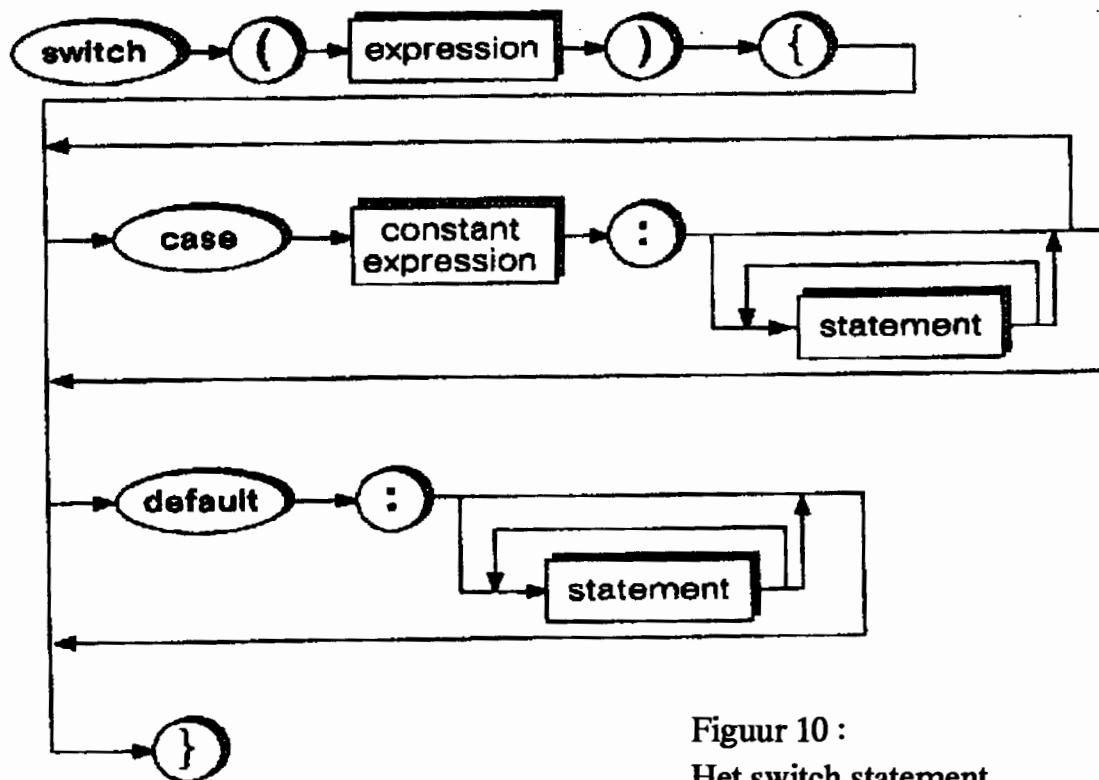
Figuur 9 :  
If / else statement

Voorbeeld if / else :

```
if (x)
    statement1;      /* Wordt uitgevoerd als x ongelijk 0 is. */
else
    statement2;     /* Wordt uitgevoerd als x gelijk 0 is. */
```

Voor statement1 en statement2 kunnen ook meerdere statements gebruikt worden, gegroepeerd d.m.v. de "compound" tekens "{" en "}".

### 2.3.2 Het switch / case / default statement :



Figuur 10 :  
Het switch statement.



Voorbeeld switch / case /default :

```
switch (x) {  
  case 0:  
  case 1:  
      statement1;      /* Wordt uitgevoerd als x of 0 of 1 is.  
      statement2;  
      break;           /* Hier stopt de ingang. */  
  case 2:  
      statement3;      /* Wordt uitgevoerd als x gelijk 3 is. */  
      break;  
  default :  
      statement4;      /* Wordt uitgevoerd als x ongelijk is aan de  
                       voorgaande case afvragingen. */  
}
```

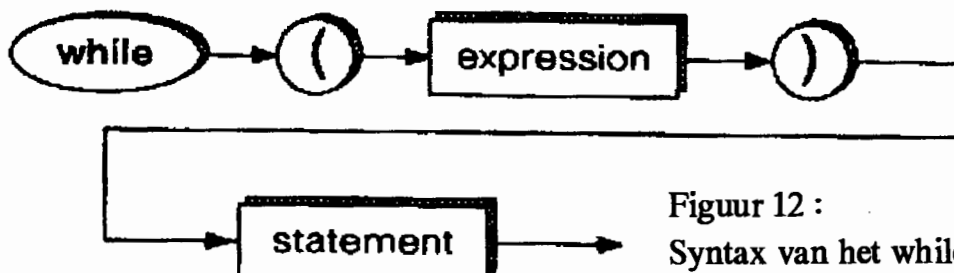
## 2.4 Statements voor programma lussen :

Met programma lussen of iteratie worden stukken programma meerdere keren doorlopen tot er aan een bepaalde voorwaarde is voldaan. De taal C kent drie statements voor lussen :

- Het "while" statement.
- Het "do...while" statement.
- Het "for" statement.

### 2.4.1 De while lus :

In figuur 11 staat de syntax van he while statement.



Figuur 12 :  
Syntax van het while statement.



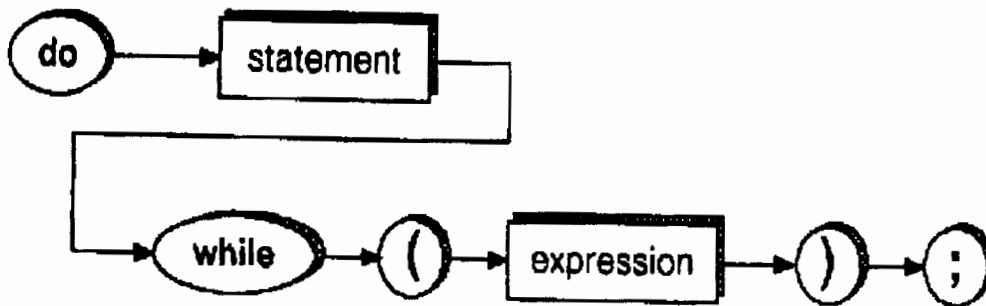
Voorbeeld van het while statement :

```
x = -1;          /* Preconditie van de lus */
while (x != 3) { /* Postconditie x == 3 */
    x++;
    printf ("x = %d\n", x);
}
```

Output :

```
x = 0
x = 1
x = 2
x = 3
```

### 2.4.2 De do...while lus :



figuur 13 :

Syntax van de do ... while lus.

Voorbeeld van de do...while lus :

```
x = -1;          /* Preconditie van de lus */
do {
    x++;
    printf ("x = %d\n", x);
} while (x != 3); /* Postconditie x == 3 */
```

Output :

```
x = 0
x = 1
x = 2
```

### 2.4.3 De for lus :

Voorbeeld van de for lus :

```
for (x = 0; x < 3; x++) {
    printf ("x = %d\n", x);
}
```

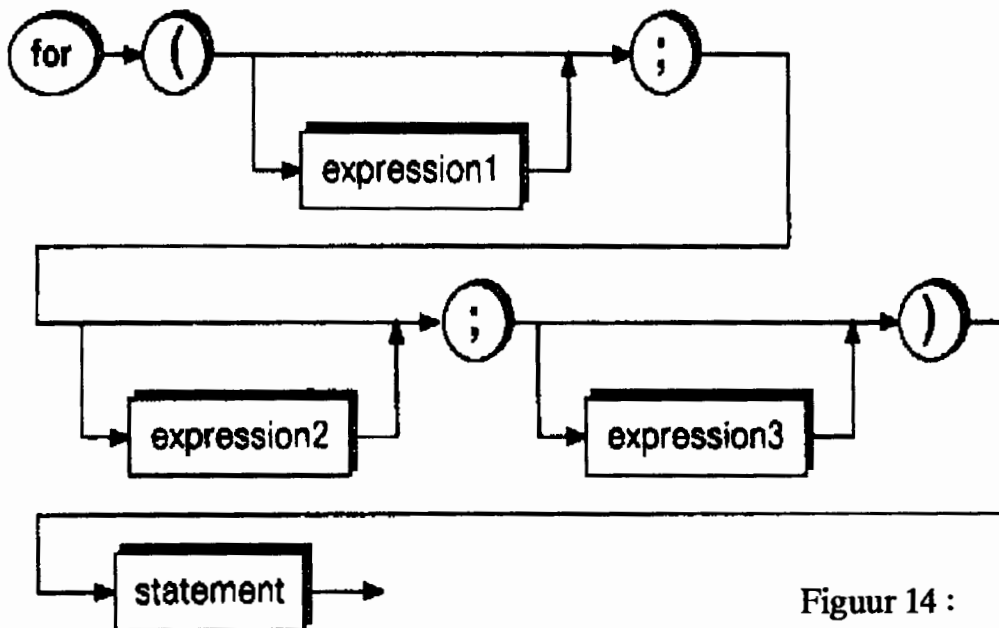


is gelijk aan :

```
x = 0;
while (x < 3) {
    printf ("x = %d\n", x);
    x++;
}
```

Output :

```
x = 0
x = 1
x = 2
```



Figuur 14 :  
Syntax van het for statement.

## 2.5 De preprocessor :

De taal C kent zoals enkele andere talen een preprocessor. Met behulp van deze processor kunnen we de source code textueel aanpassen. Letwel de text wordt voordat deze gecompileerd wordt aangepast volgens zeer strikte regels. De functies van de preprocessor zijn :

- Macro processing.
- Include (meenemen) van sourcefiles (meestal header files).
- Conditionele compilatie van de C code.

### 2.5.1 Macro processing :

Met behulp van de preprocessor kan er in de sourcecode text aangepast worden door middel van substitutie van macros. Een preprocessor directive wordt aangegeven door de eerste karakter van een regel te laten beginnen met een "#". De macro wordt aangegeven door de directive "define" :



```
#define STRING-LEN 10
```

```
static char string[STRING-LEN];
```

Voor STRING-LEN vult nu de preprocessor 10 in.

```
#define TO-LOWER(c) ((c) - ('a' - 'A'))
```

```
lower = TO-LOWER('X');
```

De preprocessor vervangt TO-LOWER door de expressie ((`'X'`) - (`'a'` - `'A'`)); de parameter `c` wordt ingevuld.

Ingebouwdw macro's :

De taal C kent een aantal voorgedefinieerde macro's :

- LINE-- :      Expandeert het regelnummer van de gebruikte regel in de sourcecode.
- FILE-- :      Expandeert de filenaam van de gebruikte sourcecode.
- TIME-- :      Expandeert de tijd van compileren.
- DATE-- :      Expandeert de datum van compileren.
- STDC-- :      Is 1 als de compiler een ANSI C standaard compileerd.

## 2.5.2 De include directive :

Met behulp van de include directive is het mogelijk om text tussen te voegen. Als er in een code regel staat `#include <gem-crs.h>` dan wordt er op die plek de inhoud van de file `gem-crs.h` tussen gevoegd.

De file wordt gezocht in de zoekpaden, welke je bij de compiler parameters kunt instellen.

## 2.5.3 Conditionele compilatie :

Door middel van conditionele compilatie is het mogelijk om delen van de code op bepaalde voorwaarden weg te laten. Dit is handig voor b.v. het maken van een testversie van het programma of het vereenvoudigen van porten van het programma



van de ene machine naar de andere.

De volgende directives zijn te gebruiken :

`#if; #else; #elif; #endif; #ifdef; #ifndef`

Voorbeeld :

```
#define TEST 2
```

```
#ifdef TEST
```

```
#if TEST == 1
```

```
    printf ("First\n");
```

```
#elif TEST == 2
```

```
    printf ("Second\n");
```

```
#else
```

```
    printf ("Last\n");
```

```
#endif /* #if TEST == 1 */
```

```
#endif /* #ifdef TEST */
```

Het symbol `test` is gedefinieerd en heeft de waarde 2. De preprocessor zal dus de regel `#elif TEST == 2` uitvoeren.

## 2.6 De runtime library :

Voor de ANSI C omgeving is ook een library gedefinieerd, waaruit standaard functies geput kunnen worden. De functies en datatypes in de library worden door middel van headerfiles bekend gemaakt aan de buitenwereld. Tijdens de linkerslag van de programma generatie worden de gebruikte functies door de linker uit de library gehaald. In de PURE-C omgeving gaat het om de `PCSTDLIB.LIB`, `PCFLTLIB.LIB` en de `PCEXTLIB.LIB`. Deze libraries worden geacht altijd aanwezig te zijn. Enkele belangrijke functies:

`math.h` : `sqrt()`, `cos()`, `sin()`, `tan()`, `acos( double x )`,  
`asin()`, `atan()`, `atan2()`.

`stdio.h` : `fclose()`, `feof()`, `fflush()`, `fgetc()`, `getchar()`, `fopen()`,  
`fprintf()`, `fputc()`, `putc()`, `fread()`, `fscanf()`, `fwrite()`, `scanf()`,  
`sprintf()`, `sscanf()`.

`string.h` : `strcat()`, `strncat()`, `strcmp()`, `strncmp()`, `strcpy()`,  
`strncpy()`, `strlen()`, `strchr()`, `strrchr()`.