Universität des Saarlandes
FR 6.2 - Informatik
Prof. Dr. W.J. Paul
Dipl.-Ing. Dominik Rester

C0 Grammatik

Startsymbol: <programm>, Nonterminale stehen in eckigen Klammern

<Zi>	→	0 | ... | 9	Ziffer
<ZiF>	→	<Zi> | <ZiF><Zi>	Ziffernfolge
<Bu>	→	a | ... | z | A | ... | Z	Buchstabe
<BuZi>	→	<Bu> | <Zi>	alphanumerisches Zeichen
<BuZiF>	→	<BuZi> | <BuZiF><BuZi>	alphanumerische Zeichenfolge
<Na>	→	<Bu> | <Bu><BuZiF>	Name
<C>	→	<ZiF>	Konstante
<id>	→	<Na> | <C> | <id>.<Na> |	Identifier (für Var. u. Konst.)
		<id>[<A>] | <id>* | &<id>	
<F>	→	<id> | $-_1$<F> | (<A>)	Faktor
<T>	→	<F> | <T> * <F> | <T> / <F>	Term
<A>	→	<T> | <A> + <T> | <A> $-_2$ <T>	(algebr.) Ausdruck
<Atom>	→	<A> > <A> | <A> ≥ <A> |	
		<A> < <A> | <A> ≤ <A> |	
		<A> == <A> | <A> ≠ <A> |	
		0 | 1	"Boole'sche Variable"
<BF>	→	<Atom> | ∼<BF> | (<BA>)	Boole'scher Faktor
<BT>	→	<BF> | <BT> ∧ <BF>	Boole'scher Term
<BA>	→	<BT> | <BA> ∨ <BT>	Boole'scher Ausdruck
<An>	→	<id> = <A> | <id> = <BA> |	Zuweisung
		if <BA> then {<AnF>} else {<AnF>} |	bedingte Anweisung
		if <BA> then {<AnF>} |	
		while <BA> do {<AnF>} |	Schleife
		<id> = <Na>(PF) |	Funktionsaufruf
		<id> = <Na>() |	Funktionsaufruf
		<Na> = new <Typ>*	Speicher anfordern
<PF>	→	<A> | <PF>,<A> |	Parameterfolge
<AnF>	→	<An> | <AnF>;<An>	Anweisungsfolge
<program>	→	<DF>; <AnF>	C_0-Programm
<DF>	→	<Dekl> | <DF>; <Dekl>	Deklarationsfolge
<Dekl>	→	<VaD> | <FuD> | <TypD>	Deklaration
<VaD>	→	<Typ> <Na>	Variablen-Deklaration
<VaDF>	→	<VaD> | <VaDF>; <VaD>	Variablen-Deklarationsfolge
<Typ>	→	<elTyp> | elTyp[<ZiF>] | elTyp*|	el. Typen, Array-Typ, Pointer
		<Na> | <Na>[<ZiF>] | <Na>* |	selbstdefinierte Typen
		struct {<KompDF>}	struct-Typ
<elTyp>	→	int | bool | char | unsigned	elementare Typen
<KompDF>	→	<KompD> | <KompDF>,<KompD>	Komponentendeklarationsfolge
<KompD>	→	<Na>:elTyp | <Na>:<Na>	Komponentendeklaration
<FuD>	→	<Typ> <Na>(<ParDF>){<VaDF>;<rumpf>} |	Funktionsdeklaration
		<Typ> <Na>(<ParDF>){<rumpf>} |	ohne lokalen Variablen
		<Typ> <Na>(){<VaDF>; <rumpf>} |	ohne Parameter
		<Typ> <Na>(){<rumpf>}	ohne lokalen Var. und P.
<ParDF>	→	<ParD> | <ParDF>,<ParD>	Parameterdeklarationsfolge
<ParD>	→	<Typ> <Na>	Parameterdeklaration
<rumpf>	→	<AnF>; return <A>| return <A>	Funktionsrumpf
<TypD>	→	typedef <Typ> <Na>	Typdeklaration