A6. Преобразуване

В зависимост от операндите си някои оператори могат да преобразуват стойността на операнда си от един тип в друг. В този раздел разглеждаме резултата, който може да се очаква при такова преобразуване. В §A6.5 е направено обобщение на преобразуването, нужно за основните оператори; както се очаква, то ще бъде придружено с преглед на всеки оператор.

A6.1 Интегрално преобразуване

Символ, късо цяло число или целочислено битово поле (независимо дали знакови или беззнакови), както и обект от тип за изброяване, могат да се използват във всеки израз, където може да се използва цяло число. Ако int може да представи всички стойности на първоначалния тип, тогава стойността се преобразува до int; в противен случай стойността се преобразува до unsigned int. Този процес се нарича интегрално преобразуване.

A6.2 Общи преобразувания

Всяко цяло число се преобразува до някакъв беззнаков тип, като предварително се намира най-малката неотрицателна стойност, съответстваща на цялото число, на степен с единица повече от най-голямата стойност, която може да се представи в съответния беззнаков тип. В двоично представяне това е еквивалентно на премахване на левия бит, ако битовият шаблон на беззнаковия тип е по-тесен, или на допълване с нулеви беззнакови стойности или знакови стойности за размножаване на знака, в случай че беззнаковият тип е по-широк.

Когато някакво цяло число се преобразува до знаков тип, стойността му остава непроменена, ако може да се представи по същия начин в новия тип; в противен случай стойността зависи от реализацията.

A6.3 Цели числа и числа с плаваща запетая

Когато стойност от тип с плаваща запетая се преобразува до интегрален тип, дробната част се пренебрегва; ако резултатната стойност не може да се представи в интегралния тип, поведението ѝ не е дефинирано. В частност резултатът от преобразуването на отрицателни стойности с плаваща запетая в беззнаков интегрален тип не е определен.

Когато стойност от интегрален тип се преобразува към тип с плаваща запетая и стойността се намира в областта на представяне, но не може да се представи съвсем точно, тогава резултатът може да бъде или следващата по-голяма стойност, или предходната по-малка. Ако резултатът не се намира в областта на представяне, поведението не е дефинирано.

A6.4 Типове с плаваща запетая

Когато стойност от тип с плаваща запетая и с по-малка точност се преобразува към стойност със същата или по-висока точност, тя остава непроменена. Когато стойност от тип с по-висока точност се преобразува към тип с по-малка точност и стойността се намира в областта на представяне, резултатът може да бъде следващата възможна по-висока стойност или предходната по-ниска. Ако резултатът излиза извън областта на представяне, поведението не е дефинирано.

A6.5 Аритметични преобразувания

Голяма част от операторите предизвикват преобразувания и определят типа на резултата по един и същ начин. Идеята е всички операнди да бъдат от един и същ тип, който ще бъде и типът на резултата. Този шаблон е известен като обикновени аритметични преобразувания (още наричан неявно преобразуване).

Първо, ако някой от двата операнда е long double, другият се преобразува до long double.

В противен случай, ако някой от двата операнда е double, другият се преобразува до double.

В противен случай, ако някой от двата операнда е float, другият се преобразува до

float.

В противен случай и към двата оператора се прилага интегрално преобразуване; после, ако някой от операторите е unsigned long int, другият се преобразувало unsigned long int.

В противен случай, ако единият операнд е long int, а другият - unsigned int, тогава резултатът се определя в зависимост от това, дали long int може да представи всички стойности на unsigned int; ако може, тогава unsigned int операндът се преобразува до long int; иначе и двата операнда се преобразуват до unsigned long int.

В противен случай, ако някой от операндите е long int, другият се преобразува до long int.

В противен случай, ако някой от операндите е unsigned int, другият се преобразува до unsigned int.

В противен случай и двата операнда са int.

Тук бяха направени две промени. Първо, аритметиката с float операнди може да се извършва в единична точност, а не само в двойна; в първото издание се казваше, че аритметиката със стойности с плаваща запетая трябва да се изпълнява единствено в двойна точност. Второ, когато по-късите беззнакови типове се комбинират с по-големи знакови типове, те не предават на резултатния тип беззнаковите свойства; в първото издание доминираща роля винаги имаше беззнаковияттип. Новите правила са малко по-сложни, но някак намаляват изненадите, които могат да се появят, когато едновременно трябва да работите със знакова и беззнакова стойност. Все още могат да се появят неочаквани резултати, когато сравнявате беззнаков израз със знаков израз с една и съща големина.

A6.6 Указатели и цели числа

Можете да прибавяте към указател или да изваждате от указател всеки израз от интегрален тип; в този случай интегралният израз се преобразува по начина, описан в §A7.7 за оператора за събиране.

Можете да изваждате два указателя от един и същ масив, сочещи към обекти от един и същ тип; резултатът се преобразува до цяло число, както е описано в дискусията за оператора за изваждане (§A7.7).

Постоянен интегрален израз със стойност 0 или явно преобразуван към тип void * може да се преобразува към указател от всякакъв тип посредством явно преобразуване, чрез присвояване или чрез сравнение. По този начин получавате нулев указател, равен на друг нулев указател от същия тип, но различен от указател към функция или обект.

Позволени са и други преобразувания, свързани с указатели, но някои от аспектите им зависят от реализацията. За да ги приложите, трябва да използвате някой оператор за преобразуване на типа или явно преобразуване (§A7.5 и §A8.8).

Всеки указател може да се преобразува до достатъчно голям целочислен тип, който може да го побере; необходимата големина зависи от реализацията. Функцията за разпределяне на големините също зависи от реализацията.

Всеки обект от даден интегрален тип може да се преобразува явно към указател. Разпределянето на големините винаги предоставя достатъчно голямо цяло число, преобразувано от указател отново в същия указател, но иначе зависи от реализацията.

Указател от един тип може да се преобразува към указател от друг тип. Резултатният указател може да не разпознае адреса, ако той не се отнася към правилно подравнен в паметта обект. Гарантира се, че указател към даден обект може да се преобразува до указател към обект, чийто тип изисква по-малко или точно същото подравняване в паметта, и обратното, без това да доведе до промени; "подравняването" зависи от реализацията, но обектите от char типовете навсякъде имат най-малки изисквания към подравняването. Както е описано в §A6.8, указател може да бъде преобразуван към тип void * и обратно, без това да предизвика промени.

Указател може да се преобразува към друг указател от същия тип, но с прибавено или премахнато определение (§A4.4, §A8.2) към обектния тип, към който се отнася указателя. Ако са добавени определения, новият указател е еквивалентен на стария, но се съобразява с ограниченията, наложени от новите определения. Ако са премахнати определения,

операциите върху основния обект остават подвластни на определенията в действителната му декларация.

Най-накрая, указател към функция може да се преобразува до указател към функция от друг тип. Извикването на функция от преобразуван указател зависи от реализацията; обаче, ако преобразуваният указател бъде преобразуван отново към първоначалния си тип, резултатът ще е идентичен на първоначалния указател.

A6.7 Void

(Несъществуващата) стойност на един void (празен неопределен тип) обект не може да се използва по всякакъв начин и не може да се прилага нито явно, нито косвено преобразуване към някакъв тип, различен от void. Тъй като void изразът обозначава несъществуваща стойност, такъв израз може да се използва само там, където самата стойност не е необходима, например като оператор-израз (§A9.2) или като ляв операнд от оператора запетая (§A7.18).

Всеки израз може да се преобразува към void тип посредством явно преобразуване. Например явното преобразуване към void обозначава пренебрегването на стойността от извикването на функция, използвано като оператор-израз.

void не беше въведен в първото издание на настоящата книга, но оттогава намери широко приложение.

A6.8 Указатели към void

Всеки указател към даден обект може да се преобразува до void *, без това да предизвика загуба на информация. Ако резултатът отново бъде преобразуван към първоначалния тип на указателя, първоначалният указател се възстановява. За разлика от преобразуването на указател към указател (§A6.6), което обикновено изисква явно преобразуване, указателите могат да се присвояват на указатели от тип void * и да се сравняват с тях.

Тази интерпретация на указателите от тип void * е нова; преди указателите от тип char * изпълняваха ролята на генерични указатели. ANSI стандартът благослови употребата на void * указатели с указатели към обекти, като позволи присвояването и сравняването им да се извършва без явно преобразуване (за разлика от всички други указатели).