Wskaźniki

Każda komórka w pamięci ma swoją nazwę, do której możemy się odwoływać przez adres, czyli wskaźnik.

Wskaźnik jest definiowany przy użyciu gwiazdki.

```
int a;
int * pa;
```

Wskaźnik jest rozmiaru słowa maszynowego (8).

`int * c, d;` - wskaźnik na c, ale d to zwykły int.

Przypisywanie adresu:

```
int *w, *p;
w = & a;
p - & a;
```

`debug(*w)` poda wartość zmiennej a.
`debug(w)` poda adres zmiennej a.

`debug(*(w+1) = 6)` - nie robić tak - stack smashing detected.

`*p = 12` - tak można, zmienia wartość zmiennej.

Powinno się inicjować wskaźniki:

`const int * pconst = nullptr;` lub `= 0;`

Jest to wyzerowany i „bezpieczny” wskaźnik. `nullptr` działa tylko dla wskaźników.

`int * const intpconst = & STALA;`

„Stały wskaźnik do int”, a nie „wskaźnik stałego int”.

`const int * const constintpconst = & STALA;`

„Stały wskaźnik do stałej”.

Tablice

Tablica jest umieszczona w pamięci element za elementem, w jednym kawałku.

Nazwa tablicy jest wskaźnikiem do jego zerowego elementu.

Funkcje

Funkcje również są gdzieś umiejscowione w pamięci i możemy odczytać ich adresy.
Przy każdym uruchomieniu programu adresy się zmieniają.

`debug((void *) nazwa funkcji);`

`void (*pf) (int [ ], const int);`

Wskaźnik pf wywołujemy jak funkcję.

pf(tab, N);

pf = inna funkcja; - zmieniamy wskaźnik na inną funkcję.

Tablica wskaźników na funkcję:

void (*tpf [ ]) (int [ ], const int) = {funkcja1, funkcja2, funkcja3, funkcja 4};

inicjacja tablicy funkcjami

Dzięki temu można wykonywać kilka operacji na tablicy.

Wektor wskaźników na funkcje

std::vector <void(*) (int [ ], const int) > nazwa = {wypelnij, kwadraty, negacja, odwroc};

Typedef - użycie jak deklaracja zmiennej - „inna nazwa zmiennej int”.

typedef int calkowita;
calkowita licznik = 5;

typedef int * pint;
pint wskaznik; // int * wskaznik
typedef int tint[M];

typedef void (*wsf) (int [ ], const int);

Wskaźniki dla niesformatowanych strumieni

Operacje wyjściowe:

write - funkcja do niesformatowanych napisów.

exit (0) - nie używać do zakończenia programu.

Operacje wejściowe:

Wczytujemy za pomocą get znaki z pliku. Sposób czytania „znak po znaku” nie jest najlepszy - jest kosztowny pod względem czasu. Używać tylko, gdy jest to konieczne.

Funkcja lustrzana to write - read. Skuteczniejsza metoda, ale wymaga zadeklarowania odpowiednio długiej tablicy.

Pliki binarne:

W pliku tekstowym bajty są interpretowane jako char. Plik binarny interpretuje binarnie.

Traktowanie tablicy intów jako tablicy charów:

plik.write((char *) & liczby, LEN * sizeof(int));

Trzeba użyć rzutowania.

Operacje dla plików binarnych:

seekg - ustaw się w dowolnym miejscu, np. plik.end (na końcu), plik.beg (na początku)
tellg - ile bajtów od początku
gcount - ile bajtów wczytano